Studies in Computational Intelligence 807

Sarah Vluymans

Dealing with Imbalanced and Weakly Labelled Data in Machine Learning using Fuzzy and Rough Set Methods

Springer

Studies in Computational Intelligence

Volume 807

Series editor

Janusz Kacprzyk, Polish Academy of Sciences, Warsaw, Poland
e-mail: kacprzyk@ibspan.waw.pl

The series "Studies in Computational Intelligence" (SCI) publishes new developments and advances in the various areas of computational intelligence—quickly and with a high quality. The intent is to cover the theory, applications, and design methods of computational intelligence, as embedded in the fields of engineering, computer science, physics and life sciences, as well as the methodologies behind them. The series contains monographs, lecture notes and edited volumes in computational intelligence spanning the areas of neural networks, connectionist systems, genetic algorithms, evolutionary computation, artificial intelligence, cellular automata, self-organizing systems, soft computing, fuzzy systems, and hybrid intelligent systems. Of particular value to both the contributors and the readership are the short publication timeframe and the world-wide distribution, which enable both wide and rapid dissemination of research output.

The books of this series are submitted to indexing to Web of Science, EI-Compendex, DBLP, SCOPUS, Google Scholar and Springerlink.

More information about this series at http://www.springer.com/series/7092

Sarah Vluymans

Dealing with Imbalanced and Weakly Labelled Data in Machine Learning using Fuzzy and Rough Set Methods

Springer

Sarah Vluymans
Department of Applied Mathematics,
Computer Science and Statistics
Ghent University
Gent, Belgium

ISSN 1860-949X ISSN 1860-9503 (electronic)
Studies in Computational Intelligence
ISBN 978-3-030-04662-0 ISBN 978-3-030-04663-7 (eBook)
https://doi.org/10.1007/978-3-030-04663-7

Library of Congress Control Number: 2018961733

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

To my family

Foreword

The number of application domains in which large quantities of data need to be processed has increased manifold over the past decades. Typical examples include text mining, bioinformatics, image processing, knowledge management, etc. This evolution has triggered a growing need for intelligent techniques in machine learning in general and in classification in particular. In this book, hybrid models called *fuzzy rough sets*, designed to handle uncertainty in data by combining both gradual properties (fuzziness) and indiscernibility (roughness), are invoked to this purpose.

Rough set theory, proposed by Zdzisław Pawlak in 1982, has proven its merits as an elegant and powerful framework for data analysis. Based on the approximation of decision classes, it allows to infer data dependencies that are useful for feature selection and decision model construction. A core notion in rough set theory is that of discernibility: the ability to distinguish between objects, based on their attribute values. The observation that in many cases objects resemble each other to a certain extent motivated the hybridization of rough sets with fuzzy sets, introduced by Lotfi Zadeh in 1965. Indeed, using fuzzy relations that model a gradual notion of discernibility between objects allows to apply the rough set methodology to real-valued data without the need for discretization.

Fuzzy rough sets were shown to perform remarkably well in lazy learning approaches, serving either as a preprocessing tool for nearest neighbour classifiers, or effectively replacing the latter by assessing test objects' membership to the lower and/or upper approximation of decision classes. They also demonstrated their use for imbalanced classification, i.e. involving data where one or several classes are under-represented.

At the same time, different noise-tolerant fuzzy rough set models were proposed to make classification more robust. In particular, the ordered weighted average (OWA) approach was identified as a promising one from both theoretical and practical angle. In order to determine if an instance belongs possibly or certainly to a concept, this approach carries out a weighted evaluation of the instance's neighbours in the training data. A key asset of the model is its association of weights with ordered positions of elements rather than with elements themselves,

which allows to express, e.g. that an object certainly belongs to a concept if most of its neighbours do, where 'most' is modelled by an appropriate OWA weight vector.

In this book, Sarah Vluymans effectively takes the research on OWA fuzzy rough sets to the next level, at once providing clear and effective guidelines on how to use them in practice and expanding its application radius to a wide range of challenging classification problems, including imbalanced, semi-supervised, multi-instance and multi-label classification.

In Chap. 3, she first focuses on the general OWA model and demonstrates that a thoughtful selection of weight vectors taking a number of simple dataset characteristics in mind can significantly improve classification. Moreover, she shows that the provided guidelines also carry over to other applications of OWA fuzzy rough sets like prototype selection, making them a sound base for machine learning practitioners to work with.

Chapter 4 addresses the difficult problem of multi-class imbalanced classification. By a skilful combination of the binary (two-class) fuzzy rough classifier IFROWANN and the one-versus-one (OVO) decomposition scheme, again involving the adaptive selection of OWA weight vectors, the author manages to outperform state-of-the-art approaches in terms of both balanced accuracy and mean AUC.

Chapter 5 evaluates the OWA model in a semi-supervised learning (SSL) context, where labels are known for a minority of training samples and unlabelled instances are used for improving generalization. As a remarkable conclusion, the author reveals that the popular self-labelling technique does not improve the traditional OWA model which employs only labelled data and that the latter even outperforms existing SSL approaches based on self-labelling.

In multi-instance learning (MIL), a data sample is described by a bag of feature vectors called instances, where the class labels of instances are not known, only those of the bags and the goal is to predict the label of new bags. In Chap. 6, fuzzy and fuzzy rough classifiers are assembled to handle MIL data that are competitive with and, in the case of imbalanced data, even superior to state-of-the-art approaches.

Finally, Chap. 7 deals with multi-label learning (MLL), a setting orthogonal to MIL where more than one label can be associated with a single data sample. Using a customized label set similarity relation within an OWA fuzzy rough nearest-neighbour based consensus approach, the author once more manages to come up with an efficient and competitive proposal.

Summarizing, through the many contributions of this book, Sarah Vluymans has managed not only to considerably widen the application scope of OWA fuzzy rough sets but also to make a convincing case for their practical use and appeal. Special praise is warranted for the meticulous and comprehensive experimental evaluation that accompanies each chapter and which serves as a shining light of best practice in machine learning.

We hope that her work will inspire other researchers to continue efforts along these lines and to further foster the paradigm of OWA fuzzy rough sets as a useful tool for machine learning.

Gent, Belgium
September 2018

Chris Cornelis
Yvan Saeys
Ghent University

Preface

This book is based on my Ph.D. dissertation completed at Ghent University (Belgium) and the University of Granada (Spain) in June 2018. It focuses on *classification*. The goal is to predict the class label of elements (that is, assign them to a category) based on a previously provided dataset of known observations. Traditionally, a number of features are measured for all observations, such that they can be described by a feature vector (collecting the values for all features) and an associated outcome, if the latter is known. In the classic *iris* dataset, for example, each observation corresponds to an iris plant and is described by its values for four features representing biological properties of the flower. The associated class label is the specific family of irises the sample belongs to and the prediction task is to categorize a plant to the correct family based on its feature values. A classification algorithm does so based on its training set of labelled instances, that is, a provided set of iris flowers for which both the features values and class labels are known. One of the most intuitive classifiers is the nearest neighbour algorithm. To classify a new element, this method locates the most similar training instance (the nearest neighbour) and assigns the target to the class to which this neighbour belongs. Other methods build an explicit classification model from the training set, for example, in the format of a decision tree.

Few real-world datasets are perfect, that is, it is usually not possible to make entirely accurate class predictions based on the training set. One natural issue is the uncertainty present in any dataset. Mathematics provides us with frameworks to model such data imperfections and this from different viewpoints. *Fuzzy set theory* extends traditional set theory by allowing a partial membership of elements to a set in the form of a real-valued membership degree between zero and one. In this way, vague and subjective concepts as well as graded relationships between observations can be represented. Data incompleteness or indiscernibility is another common obstacle and refers to the situation where the measured features are insufficient to provide a precise definition for or sharply delineate a concept. *Rough set theory* resolves this by approximating the concept with a lower (conservative) and upper (liberal) approximation. Fuzzy and rough set theory have been combined into *fuzzy rough set theory*. A graded similarity between observations is incorporated and the

fuzzy rough lower and upper approximations of a fuzzy concept are themselves fuzzy sets. Fuzzy rough sets have been used in several machine learning algorithms, both in the preprocessing and learning phases. In the classification context, the fuzzy rough approximations are determined for the different classes. In this book, we develop fuzzy rough set based classification algorithms for imbalanced and weakly labelled data, challenging types of datasets extending the traditional format presented above.

Chapters 1 and 2 together form the introduction to this work. The former describes the types of datasets under study and provides the definitions of fuzzy set theory, rough set theory and fuzzy rough set theory together with intuitive examples. The latter introduces the classification domain. It discusses the classification task in general, including common challenges such as the bias-variance trade-off and the curse of dimensionality. A substantial part of Chap. 2 is dedicated to the description of such prevalent classification approaches as nearest neighbour classification, decision tree algorithms and support vector machines. Finally, we specify how an experimental evaluation of classification algorithms can be conducted in a correct manner.

In Chap. 3, we study the OWA based fuzzy rough set model, a robust generalization of traditional fuzzy rough sets. This noise-tolerant extension uses OWA aggregations to compute the membership degrees of observations to the fuzzy rough lower and upper approximations. An OWA aggregation of a set of values is based on a vector of weights used to weigh the contribution of each individual value. A so-called weighting scheme defines these weight vectors. To preserve the intuition behind the fuzzy rough lower and upper approximations, vectors of increasing and decreasing weights are, respectively, used in their definitions. To date, there are no clear further instructions on which weighting scheme should be used in machine learning algorithms relying on OWA based fuzzy rough approximation operators. As our experiments show that the preference for a particular weighting scheme varies between datasets, we remedy this shortcoming in Chap. 3. We develop an OWA weighting scheme selection strategy for both the lower and upper approximation based on easy-to-understand and simple-to-compute dataset characteristics like the overall size or the number of classes. By providing these guidelines, we remove the need for the user to choose between one of the available weight definitions themselves. Apart from solving this issue, our detailed study also explains the behaviour and internal characteristics of the OWA based fuzzy rough approximations. The building blocks provided in this chapter greatly improve the ease-of-use and understanding of this fuzzy rough set model and render it a more accessible tool for use in future machine learning techniques.

Chapter 4 tackles the first challenge in classification data, namely, the presence of *class imbalance*. Most research on imbalanced data has been performed for two-class or binary data with one majority and one minority class. When the elements of the former considerably outnumber those of the latter, classification algorithms can be hindered in their prediction performance. In particular, they tend to predict the majority class label too often and result in many minority class misclassifications. To amend this issue, specialized algorithms have been developed

to deal with the class imbalance in the training set in either the preprocessing or learning phase. In Chap. 4, we follow the more recent trend in the research community and propose an algorithm for multi-class imbalanced datasets, in which the number of classes exceeds two and the distribution of training instances among them is (severely) skewed. Our classifier is called FROVOCO and is based on the one-versus-one decomposition scheme that divides the multi-class problem into several binary sub-tasks, one for each pair of classes. To classify a test instance, each binary classifier trained on a sub-task is fired and computes class confidence degrees for the element to both classes under its consideration. All values are grouped in a score-matrix and afterwards aggregated to one class prediction for the target. Within FROVOCO, we use an adaptive version of the IFROWANN classifier, a fuzzy rough set based classifier for binary imbalanced data. The proposed adaptive aspect lies with the dynamic choice of OWA weights depending on the class imbalance of the binary problem at hand. The second novel component of FROVOCO is our newly proposed WV-FROST aggregation procedure used to extract a class prediction from the score-matrix. We complement the local information grouped in the score-matrix with a global class assessment by means of two fuzzy rough set based affinity terms. We extensively evaluate our FROVOCO method on multi-class imbalanced datasets, validating our two proposed novel components and showing the dominance of our complete method over existing proposals.

In Chap. 5, we consider semi-supervised data, a context in which class information is available for only part of the training instances. Consequently, the training set consists of both labelled and unlabelled elements. A classification algorithm trained on a semi-supervised dataset can (in principle) use the information available in both the labelled and unlabelled elements. This chapter studies the application of our fuzzy rough set based classifiers from Chap. 3 on semi-supervised classification data. In particular, we assess whether our algorithms benefit from a *self-labelling* step, in which the labelled part of the training set is extended by deriving class predictions for a portion of the unlabelled instances. Our experimental study indicates that (i) our methods perform strongly in spite of the semi-supervised characteristic of their training set, (ii) they do not benefit from a prior self-labelling step and are instead able to extract sufficient information from the limited amount of labelled training elements and (iii) they constitute a powerful and computationally more efficient alternative to semi-supervised classification algorithms that do rely on self-labelling.

Chapter 6 considers the classification of *multi-instance data*, where each observation is represented by a group (a bag) of feature vectors. No class label is available for the individual instances, only their class assignment as a full bag is known. Image classification is an example setting that can be modelled with multi-instance data. An observation (bag) corresponds to a whole image, which can be divided into several image regions or segments (instances). The prediction task is to decide which complete scene the image represents. Several general approaches to multi-instance classification can be followed depending on whether the information discerning between classes is extracted at the level of instances, the level of bags or

in an induced feature space mapping whole bags to single feature vectors. We propose two frameworks of multi-instance classification algorithms based on fuzzy set theory and fuzzy rough set theory, respectively. The former consists of general multi-instance classifiers, while the latter group of algorithms is specifically developed for class imbalanced multi-instance data. Both groups can be further divided into two families of instance-based and bag-based methods. We consequently develop four categories of algorithms: (i) fuzzy instance-based methods, (ii) fuzzy bag-based methods, (iii) fuzzy rough instance-based methods and (iv) fuzzy rough bag-based methods. Their category defines the general flow of their calculations and prediction procedure, but we propose several possible settings of the internal parameters of our methods, that is, several ways to concretely perform all computations. We present a comprehensive experimental evaluation of these parameter choices and explain why certain options can be favoured over others. Based on our conclusions, we compare our proposed methods with their most suitable settings to existing multi-instance classifiers on both balanced and imbalanced datasets and show their strong prediction performance overall. For class imbalanced multi-instance datasets in particular, we are able to conclude the distinct dominance of our fuzzy rough set based methods over existing proposals.

We consider another extension of the traditional dataset format in Chap. 7, namely, that of *multi-label data*. In multi-label datasets, observations are labelled with several classes at once and the classification task is to predict the entire labelset for a target rather than a single class label. One possible way to do so is to adopt a nearest neighbour based approach, wherein the labelset prediction is derived from class information in the vicinity of the target instance, that is, by aggregating the labelsets of neighbouring training elements in a specific way. We propose a new method for the latter step. Our FRONEC algorithm uses OWA based fuzzy rough set theory to derive an appropriate consensus prediction from the labelsets encountered in the neighbourhood of the instance to classify. In an experimental evaluation on both synthetic and real-world datasets, we show that our FRONEC proposal is highly competitive with (and often outperforms) existing nearest neighbour based multi-label classifiers.

Finally, Chap. 8 concludes the book by summarizing our proposals, results and conclusions and outlining several possible directions of future research. Note that previously published work related to the material presented here includes [217, 361, 421, 422, 423, 425, 426].

Ghent, Belgium
September 2018

Sarah Vluymans

Contents

Chapter 1
Introduction

Generally put, this book is on fuzzy rough set based methods for machine learning. We develop classification algorithms based on fuzzy rough set theory for several types of data relevant to real-world applications. Before going into detail on our models, a short introduction to these main components is required. In Sect. 1.1, we first present the problem of imbalanced data and several challenging other data types, which can be grouped under the umbrella term of weakly labelled data, for which we develop fuzzy rough set based classification methods in later chapters. Section 1.2 provides a brief initiation into fuzzy rough set theory. This mathematical framework for modelling data uncertainty is discussed sufficiently detailed, but without going into too much of the mathematical particulars. The latter aspect is reserved for later chapters. We include a general discussion on machine learning with a particular focus on how fuzzy rough set theory has already been used in this domain in Sect. 1.3. Finally, to conclude this introductory chapter, we provide an overview of the book in Sect. 1.4.

1.1 Imbalanced and Weakly Labelled Data

Machine learning is a field of study concerned with computer algorithms that enhance their knowledge of or performance in some task through experience [155, 323, 328]. The need for explicit programming is reduced. In this work, the concept of experience refers to available information provided in the form of a dataset containing (supposedly) correctly labelled observations. We focus on the task of *classification* (Chap. 2), which requires a method to construct a prediction model or mechanism based on a collected set of labelled elements (the *training set*).

In standard *supervised* learning, the learner is presented with a fully labelled training set, that is, every instance is associated with a known outcome. This outcome

S. Vluymans, *Dealing with Imbalanced and Weakly Labelled Data in Machine Learning using Fuzzy and Rough Set Methods*, Studies in Computational Intelligence 807, https://doi.org/10.1007/978-3-030-04663-7_1

Table 1.1 A portion of the *iris* dataset

Sepal length	Sepal width	Petal length	Petal width	Class
5.1	3.5	1.4	0.2	Iris Setosa
4.9	3.0	1.4	0.2	Iris Setosa
4.6	3.1	1.5	0.2	Iris Setosa
7.0	3.2	4.7	1.4	Iris Versicolor
6.4	3.2	4.5	1.5	Iris Versicolor
6.9	3.1	4.9	1.5	Iris Versicolor
6.3	3.3	6.0	2.5	Iris Virginica
5.8	2.7	5.1	1.9	Iris Virginica
7.1	3.0	5.9	2.1	Iris Virginica

can be categorical, in which case the prediction task corresponds to *classification*, or continuous, when we consider *regression*. An instance or observation x can be represented by a feature vector and the ith position in this vector contains the value of x for the ith descriptive feature. A standard supervised training dataset can consequently be represented in a flat table of n rows and $d + 1$ columns (with n the number of instances and d the number of features). The additional column (commonly the last one) contains the outcome. Table 1.1 contains an example classification dataset, namely a portion of the widely popular *iris* dataset created by Robert Fisher [153]. The observations are described by four features measuring plant properties. The class label represents the type of iris (setosa, versicolor or virginica). The learning task associated with a supervised training dataset is to derive a prediction model to predict the outcome of newly presented instances of which only the feature values are known.

Several properties of a dataset can render it inherently more challenging to use in a classification setting. One often encountered issue is an uneven division of observations across the classes, implying that relatively more information is available for some classes than others. The topic of *imbalanced data* is briefly introduced in Sect. 1.1.1. Aside from the challenge of a skewed class distribution, the standard dataset format presented in Table 1.1 can be generalized in several ways. First, we can consider *semi-supervised data* ([72, 514], Sect. 1.1.2), which can be placed on the midpoint between supervised (Table 1.1) and unsupervised data (Table 1.1 without the final column). The training set contains both labelled and unlabelled instances and the aim is to combine the information in both to predict the outcome of the unlabelled elements in the training set as well as any newly presented instances. Secondly, we report *multi-instance learning* ([217], Sect. 1.1.3). In a multi-instance dataset, each observation is represented by several feature vectors. Together, they form one bag. Only the bag as a whole has an associated outcome, while its constituent instances do not. A third generalization is represented by *multi-label learning* ([216], Sect. 1.1.4), where each observation can be associated with more than one class label. Every instance may have a different number of outcomes and relationships between the

possible labels can exist. The term multi-label learning is usually reserved for classification tasks, while *multi-target learning* is used for regression datasets. The three associated dataset formats can be grouped on the common denominator of *weakly labelled data*. A taxonomy for weakly labelled data has recently been proposed in [215] based on three axes: (i) the instance-label relationship, (ii) the supervision model in the learning phase and (iii) the supervision model in the prediction stage. The multi-instance and multi-label paradigms relate to the first axis, while the semi-supervised setting relates to the supervision model in the learning stage.

1.1.1 Imbalanced Data

Table 1.2 depicts an imbalanced version of the *iris* dataset. Only two classes are present, setosa and versicolor. The former contains two observations, while the latter consists of six samples. The fact that there are three times as many elements of one class than of the other in the training set can inhibit a learning algorithm to construct a strong-performing prediction model, that is, a classifier that would perform well on both classes. Traditional classification methods tend to predict the majority class too often and lead to relatively too many classification errors on the minority class, which is often the class of interest [57, 211, 391]. Class imbalance presents itself in many research areas, including medical diagnosis and bioinformatics.

In a two-class imbalanced dataset, the *imbalance ratio* (IR) is defined as the ratio of the size of the majority class (often called 'negative', N) to the size of the minority class (often called 'positive', P). In particular, $\text{IR} = \frac{|N|}{|P|}$. The higher this value, the more imbalance exists between the two classes. When more than two classes are present, this metric can for instance be generalized to

$$\text{IR} = \frac{\max\limits_{C \in \mathcal{C}} |C|}{\min\limits_{C \in \mathcal{C}} |C|}, \tag{1.1}$$

Table 1.2 An imbalanced version of the *iris* dataset

Sepal length	Sepal width	Petal length	Petal width	Class
5.1	3.5	1.4	0.2	Iris Setosa
4.9	3.0	1.4	0.2	Iris Setosa
7.0	3.2	4.7	1.4	Iris Versicolor
6.4	3.2	4.5	1.5	Iris Versicolor
6.9	3.1	4.9	1.5	Iris Versicolor
6.6	3.0	4.4	1.4	Iris Versicolor
6.7	3.0	5.0	1.7	Iris Versicolor
5.7	2.6	3.5	1.0	Iris Versicolor

Table 1.3 A portion of the *iris* dataset in semi-supervised classification

Sepa length	Sepal width	Petal length	Petal width	Class
5.1	3.5	1.4	0.2	Iris Setosa
4.9	3.0	1.4	0.2	?
4.6	3.1	1.5	0.2	?
7.0	3.2	4.7	1.4	Iris Versicolor
6.4	3.2	4.5	1.5	Iris Versicolor
6.9	3.1	4.9	1.5	?
6.3	3.3	6.0	2.5	?
5.8	2.7	5.1	1.9	Iris Virginica
7.1	3.0	5.9	2.1	?

where $\mathcal{C}$ is the set of all classes. Class imbalance, in particular the general setting of an imbalanced dataset with more than two classes, is studied in Chap. 4.

1.1.2 Semi-supervised Data

Table 1.3 presents an example semi-supervised dataset. This is the same portion of the *iris* dataset as in Table 1.1, but not all class labels are known. For some training instances, the outcome is hidden. This type of datasets is commonly encountered in applications where the label assignment is costly or difficult to obtain [514]. As a result, only a (very) small portion of the training instances is labelled, complemented with a (large) number of unlabelled elements. Application areas in which training sets are often only partially labelled include natural language processing, bioinformatics and image recognition. The common characteristic of these domains is that data is often abundantly available or relatively easy to obtain, but challenging or expensive to annotate.

In *semi-supervised classification*, the goal is to use the information in both the labelled and unlabelled parts to construct a classification model. In *semi-supervised clustering* on the other hand, the labelling information is transformed to a set of constraints to which the clustering method should adhere. Semi-supervised learning is further explored in Chap. 5.

1.1.3 Multi-instance Data

Multi-instance learning was first proposed in [127] to deal with representation ambiguity of training samples. In particular, this study focused on the classification task where several alternative feature vectors represent the same entity. The collection of

Table 1.4 Two observations from the multi-instance *Musk1* dataset

BagID	f_1	f_2	…	f_{166}	f_167	Class
MUSK-jf59	52	−110	…	−60	−29	Positive
	49	−98	…	−13	−12	
	23	−113	…	−9	90	
	47	−110	…	−5	−8	
	9	−114	…	−28	112	
NON-MUSK-334	7	−197	…	34	55	Negative
	25	−198	…	20	−8	

instances (feature vectors) that make up one observation is called a *bag*. Each bag in a multi-instance dataset can contain a different number of instances, but all instances are described by the same set of features. Two observations contained in the traditional *Musk1* dataset, used in the original multi-instance proposal and the bulk of later experimental studies, can be found in Table 1.4. Each observation corresponds to a chemical molecule. One and the same molecule can have different conformations or shapes, each corresponding to an instance. This dataset was used in drug activity prediction, where the task is to predict whether a molecule can bind to a given target, making it a so-called good drug molecule. The classification outcome can be 'positive' or 'negative'. A positive class label means that at least one (but not necessarily all) of the conformations of the molecule lead to a target binding. However, as this label is only assigned to the bag as a whole, there is no direct indication which instance has the binding property. The hidden relation between instances and classes renders multi-instance classification a more general and challenging prediction task than traditional single-instance classification.

A recent and thorough review of the multi-instance learning domain can be found in [217]. Application domains can be divided into three general groups. The first group, which includes the drug prediction task described above, consists of fields with different alternative views, representations or descriptions for the same object. A second branch of multi-instance data sources study compound objects. Each instance corresponds to a particular part of the object. A classic example is the image recognition task, where the image object is divided into several smaller segments. Finally, evolving objects, sampled at different time points, can be modelled as multi-instance data as well. Multi-instance learning forms the focus of Chap. 6.

1.1.4 Multi-label Data

The multi-label learning domain has been reviewed in e.g. [195, 216, 488]. As opposed to the traditional classification task associated with the data format in Table 1.1, the goal in multi-label classification is to predict the presence of multiple

Table 1.5 Three observations from the multi-label *Birds* dataset

f_1	f_2	…	f_{260}	Labels
0.054606	0.161667	…	8	Swainson Thrush
0.027401	0.015898	…	13	Varied Thrush, Hermit Warbler
0.060964	0.187454	…	15	Pacific-slope Flycatcher, Varied Thrush,
				Golden Crowned Kinglet

labels at the same time. Every element can belong to more than one category at once and the number of classes can be different for each instance. Example multi-label classification applications are image processing and text categorization. An image or text can naturally belong to multiple classes at the same time, when it depicts several concepts (e.g. a park with playing children) or discusses several topics (e.g. a review of a political play).

As an example, Table 1.5 groups some elements of the *Birds* dataset [61]. Each instance corresponds to an audio track, labelled with the birds whose song is present in the track. There are 19 possible bird species and several birds can be heard together in some tracks. For example, the third sample contains sounds from three different birds. Chapter 7 discusses the domain of multi-label learning in more detail.

1.2 Fuzzy Rough Set Theory

In this book, we use fuzzy rough set theory in classification algorithms to deal with the challenging data types discussed above. Fuzzy rough sets were proposed in the seminal contribution of [132] as a fusion of two existing mathematical paradigms: fuzzy set theory [477] and rough set theory [342]. In particular, rough sets were generalized to fuzzy rough sets by allowing (or rather, introducing) fuzziness in several strategic places. As such, more flexibility was obtained and several real-world problems could be handled in a more appropriate manner.

Both fuzzy and rough sets, discussed in Sects. 1.2.1 and 1.2.2 respectively, aim to capture uncertainty in data. The former does so by modelling *vagueness*, while the latter focuses on *incompleteness* or *indiscernibility*. The combination of the two into fuzzy rough sets implies the ability to model both (complementary) types of data uncertainty. The traditional fuzzy rough set model is recalled in Sect. 1.2.3.

1.2.1 Fuzzy Sets

Fuzzy sets were introduced in [477] to model intrinsically vague or subjective notions. In realistic problems, it is not always possible to provide a crisp definition of a con-

cept or a black-and-white division of data into groups. A simple example is the question how to define *expensive property* in the housing market. Contemplating this problem should immediately demonstrate that it is difficult to find one threshold above which any property should be considered expensive. This threshold may be context-dependent, since one could for example expect different cut-offs to apply for expensive apartments or expensive mansions. The problem is also naturally subjective, because a wealthy person may employ a higher threshold than a less-privileged one. Another issue is the artificial division this threshold-based approach implies. Suppose the threshold of €400000 has been selected. A property of €401000 would be considered expensive, while one of €399000 would not be so. The difference in these prices would be negligible for most buyers and yet they lead to a different assignation of 'expensive' and 'not expensive'. Finally, one can also oppose the lack of gradation present in the crisp definition of expensive property. Two houses costing half a million and five million euros respectively would both be considered expensive without making a further distinction between them.

These issues can be resolved by allowing a graded membership of elements to a set. In traditional set theory, an element either belongs to a set or it does not. Fuzzy sets allow elements to belong to them to a certain degree. The membership function $A(\cdot)$ associated with a fuzzy set takes on values in the unit interval $[0, 1]$. At the lower end of the spectrum, $A(x) = 0$ indicates that element x does not belong to A at all. The other extreme $A(x) = 1$ means that x fully belongs to A. Any value between 0 and 1 corresponds to a partial membership of the element to the set. Figure 1.1 presents an example membership function for the fuzzy set of expensive property. As common-sense dictates, this is an increasing function. The left dashed line shows that a house costing €250000 has a membership degree of about 0.3 to this set, meaning that it is not considered all that expensive. On the other hand, a house price of €550000 (represented by the right dashed line) is considered sufficiently steep and has a membership degree of about 0.9. Fuzzy set theory clearly provides us with a flexible tool to address the issues listed above. Aside from set membership, the same ideas can be incorporated in fuzzy relations and fuzzy logic, discussed in the following paragraphs.

In traditional set theory, a relation between elements can be represented as a set of pairs. If the pair (x_1, x_2) belongs to this set, the two elements are related. This results in a similar crisp delineation between instances that are related to each other and instances that are not. It does not allow one to express the degree to which two instances are related. This issue is dealt with by introducing a *fuzzy relation*, defined as a mapping from the set of all possible pairs of elements to the unit interval. For a fuzzy relation R, value $R(x_1, x_2)$ expresses the degree of relatedness of x_1 and x_2. The closer this value is to one, the stronger the relation between the two instances.

Fuzzy logic extends traditional Boolean logic to the fuzzy setting [262]. Fuzzy logic operators have been defined to generalize the principles of conjunction, disjunction, implication etc. A *triangular norm (t-norm)* $\mathcal{T} : [0, 1]^2 \to [0, 1]$ is a commutative and associative operator that is increasing in both arguments and upholds the boundary condition $(\forall a \in [0, 1])(\mathcal{T}(a, 1) = a)$. It is related to the Boolean conjunction, as is evident when the domain is restricted to {0,1}. A fuzzifica-

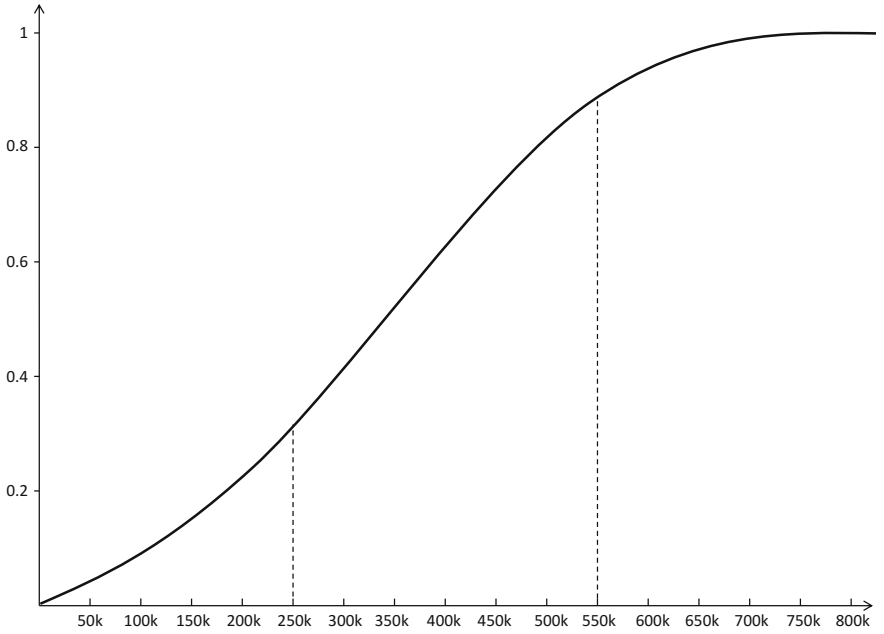


Fig. 1.1 Example membership curve for the fuzzy set of expensive property

tion of the Boolean disjunction is represented by a *triangular conorm (t-conorm)* $\mathcal{S} : [0, 1]^2 \to [0, 1]$, which is also commutative, associative and increasing in its two arguments, but has 0 as neutral element, that is, $(\forall a \in [0, 1])(\mathcal{S}(a, 0) = a)$. A third group of operators consist of so-called *implicators* $\mathcal{I} : [0, 1]^2 \to [0, 1]$, that are decreasing in their first argument, increasing in the second and satisfy the boundary conditions $\mathcal{I}(0, 0) = \mathcal{I}(0, 1) = \mathcal{I}(1, 1) = 1$ and $\mathcal{I}(1, 0) = 0$. They form the fuzzy extension of traditional implication, as is evident from the boundary conditions. Finally, the Boolean negation is extended to a *negator* $\mathcal{N} : [0, 1] \to [0, 1]$, which is a decreasing operator that satisfies $\mathcal{N}(1) = 0$ and $\mathcal{N}(0) = 1$. A wide variety of choices for operators $\mathcal{T}, \mathcal{S}, \mathcal{I}$ and $\mathcal{N}$ exist in the literature, examples of which can be found in Table 1.6.

1.2.2 Rough Sets

Rough set theory was proposed in [342]. A rough set consists of two sets which together approximate a given incomplete concept. Incompleteness should be understood as the inability of the measured features to discern the concept. Consider the example of course evaluation data provided by university students. At the end of a teaching term, it is common practice to ask students to fill in a questionnaire on

Table 1.6 Example fuzzy logic operators: t-norms, t-conorms, implicators and a negator

T-norm		T-conorm	
Name	Definition	Name	Definition
Minimum	$\mathcal{T}(a,b) = \min(a,b)$	Maximum	$\mathcal{S}(a,b) = \max(a,b)$
Product	$\mathcal{T}(a,b) = a \cdot b$	Probabilistic sum	$\mathcal{S}(a,b) = a + b - a \cdot b$
Łukasiewicz	$\mathcal{T}(a,b) = \max(a+b-1,0)$	Bounded sum	$\mathcal{S}(a,b) = \min(1,a+b)$
Implicator		Negator	
Name	Definition	Name	Definition
Kleene-Dienes	$\mathcal{I}(a,b) = \max(1-a,b)$	Standard	$\mathcal{N}(a) = 1-a$
Łukasiewicz	$\mathcal{I}(a,b) = \min(1-a+b,1)$		
Reichenbach	$\mathcal{I}(a,b) = 1-a+a \cdot b$		

courses they have taken. Assume that there are three statements that the student needs to reply to, namely *'The course met my expectations.'*, *'The pace of teaching was appropriate.'* and *'The exercises were a fitting addition to the theory classes.'*. For each of these, the students can select one of the options 'Agree', 'Somewhat agree', 'Neutral', 'Somewhat disagree' and 'Disagree' as their response. They are also required to give an overall evaluation of 'Good', 'Intermediate' or 'Bad'. A small example dataset based on such an inquiry is depicted in Table 1.7.

The task could be to derive a definition of a good course based on the replies of the students to the three statements. However, based on the data collected in Table 1.7, an unambiguous delineation is not possible. The second and fifth student have each replied 'Somewhat agree' to each of the statements, based on which they assign the course a 'Good' and 'Intermediate' overall evaluation respectively. This shows that the evaluated feature set is incomplete: two elements of different classes coincide in all feature values.

Rough set theory approximates the class of good courses in two ways. The *lower approximation* is the set of elements (i) which belong to this class and (ii) for which all elements with exactly the same feature values also belong to the class. Based on Table 1.7, the lower approximation of 'Good' consists of students s_1 and s_3. It can be interpreted as the set of elements that *certainly* belong to this concept, as there is no evidence against their membership. It is a conservative approximation. The *upper approximation* on the other hand derives a liberal approximation of the 'Good' class. This set contains all instances for which there is at least one data point with exactly the same feature values and the 'Good' evaluation. In this case, the upper approximation contains students s_1, s_2, s_3 and s_5.

It should be clear from this simple example that the definitions of the lower and upper approximation entirely rely on the observed feature values. The full feature or attribute set $\mathcal{A}$ defines an equivalence relation on the observations, resulting in

Table 1.7 Example data collected from student evaluations on a particular university course. Each row corresponds to the answers of one student

	Statement 1	Statement 2	Statement 3	Evaluation
s_1	Agree	Agree	Neutral	Good
s_2	Somewhat agree	Somewhat agree	Somewhat agree	Good
s_3	Somewhat agree	Neutral	Agree	Good
s_4	Somewhat agree	Somewhat agree	Neutral	Intermediate
s_5	Somewhat agree	Somewhat agree	Somewhat agree	Intermediate
s_6	Agree	Neutral	Somewhat disagree	Intermediate
s_7	Agree	Neutral	Disagree	Intermediate
s_8	Somewhat disagree	Neutral	Somewhat disagree	Bad
s_9	Disagree	Agree	Somewhat disagree	Bad
s_{10}	Disagree	Agree	Disagree	Bad

a partition of the elements into equivalence classes. The equivalence class $[x]$ of element x is defined as

$$[x] = \{y \mid (\forall a \in \mathcal{A})(a(x) = a(y))\}. \tag{1.2}$$

In this expression, $a(x)$ and $a(y)$ correspond to the values of feature a for elements x and y respectively. It should be clear that the equivalence class of x consists of all instances y that have exactly the same feature values as x. Based on this equivalence relation, the lower and upper approximation of a concept C in dataset T can be defined as

$$\underline{C} = \{x \mid x \in T, [x] \subseteq C\}$$

and

$$\overline{C} = \{x \mid x \in T, [x] \cap C \neq \emptyset\}$$

respectively. The former consists of elements for which the equivalence class is entirely contained in C, while the latter groups elements for which the equivalence class has a non-empty intersection with C. An alternative (but equivalent) formulation of these sets is

$$x \in \underline{C} \Leftrightarrow (\forall y \in T)(y \in [x] \rightarrow y \in C) \Leftrightarrow (\forall y \in T)((x, y) \in R \rightarrow y \in C) \tag{1.3}$$

and

$$x \in \overline{C} \Leftrightarrow (\exists y \in T)(y \in [x] \wedge y \in C) \Leftrightarrow (\exists y \in T)((x, y) \in R \wedge y \in C), \tag{1.4}$$

where R is the crisp equivalence relation defining the equivalence classes. As briefly noted in Sect. 1.2.1, any traditional relation can be represented as a set of pairs of related elements.

Rough set theory forms an ideal tool to approximate indiscernible concepts in datasets of the format represented in Table 1.7. However, an important limitation lies with its use of the equivalence classes (1.2). The case of categorical data, where each feature can take on a finite (and limited) amount of possible values, is handled appropriately by using (1.2). In the presence of continuous numerical data, the interpretability of this definition is reduced. It is unlikely for instances to exactly coincide in a numerical feature, even though their values may be close together. The equivalence classes will consequently often be singletons, such that the core intuition behind the lower and upper approximations is lost. A solution would be to divide the range of numerical features into several intervals by means of discretization [189]. These intervals can be interpreted as categories, but any discretization process will result in information loss, which may therefore not be the ideal fix. A second limitation is that, similar to the vagueness issue discussed in Sect. 1.2.1, the definitions of $\underline{C}$ and $\overline{C}$ assume that C is a traditional crisp set.

1.2.3 Fuzzy Rough Sets

To deal with the restraints associated with rough sets in general datasets, fuzzy rough set theory has been proposed in [132]. The authors introduced fuzziness into the rough set operators. In particular, both the approximated concept and similarity between elements are allowed to be fuzzy, the former modelled by a fuzzy set and the latter by a fuzzy relation.

One of its crucial components is the fuzzy relation R measuring similarity between elements. For two instances x and y, $R(x, y) \in [0, 1]$ represents how strongly they are related or, more precisely, how similar we consider them to be. This fuzzy relation replaces its crisp relative appearing in (1.3)–(1.4). We can fuzzify each component within these expressions to derive the definitions of the fuzzy rough lower and upper approximations. These are fuzzy sets, to which every instance in T has a membership degree between zero and one. The fuzzy rough lower approximation of a (possibly fuzzy) concept C is defined as

$$\underline{C}(x) = \min_{y \in T} [\mathcal{I}(R(x, y), C(y))]. \tag{1.5}$$

Aside from the use of the fuzzy relation R, the universal quantifier in (1.3) has been replaced by a minimum operator, the implication has been generalized to an implicator (see Sect. 1.2.1) and the membership of y to the fuzzy set C is represented by its membership degree. By using the minimum instead of infimum operator, we have assumed that T is finite. This will be the case in both this work and real-world

applications. In a similar way, the fuzzy rough upper approximation of C is derived as

$$\overline{C}(x) = \max_{y \in T}[\mathcal{T}(R(x, y), C(y))], \tag{1.6}$$

where the maximum operator has replaced the existential quantifier and a t-norm is used instead of the crisp conjunction. Expressions (1.5)–(1.6) correspond to the general implicator/t-norm fuzzy rough set formulation proposed in [358].

As stated above, fuzzy relation $R : T^2 \rightarrow [0, 1]$ expresses the degree of similarity between two elements in T. Usually, one assumes that this relation is at least reflexive $((\forall x \in T)(R(x, x) = 1))$ and symmetric $((\forall x, y \in T)(R(x, y) = R(y, x)))$. When R has both these properties, it is called a fuzzy tolerance relation. If a fuzzy tolerance relation is also $\mathcal{T}$-transitive with respect to some t-norm $\mathcal{T}$ $((\forall x, y, z \in T)(\mathcal{T}(R(x, y), R(y, z)) \leq R(x, z)))$, it is called a fuzzy $\mathcal{T}$-similarity relation.

A consequence of the use of the minimum and maximum operators in (1.5)–(1.6) is that the $\underline{C}(x)$ and $\overline{C}(x)$ membership degrees are highly sensitive to noise and outliers present in dataset T. Indeed, the addition of one element y to T can have a large effect on these definitions, since it can lead to a new minimum value in (1.5) or a new maximum value in (1.6). This poor noise robustness of traditional fuzzy rough sets has been pointed out in several places in the literature and a variety of noise-tolerant fuzzy rough set models have afterwards been proposed to address this issue [118, 229]. These include β-precision fuzzy rough sets [150], variable precision fuzzy rough sets [324, 325], vaguely quantified fuzzy rough sets [106], fuzzy variable precision rough sets [497], soft fuzzy rough sets [224, 225], ordered weighted average based fuzzy rough sets [108], variable precision fuzzy rough sets based on fuzzy granules [465] and a data distribution aware fuzzy rough set model [15]. Among them, ordered weighted average based fuzzy rough sets can be considered a preferred alternative, both theoretically and empirically [118]. This model is studied in detail in Chap. 3.

1.3 Fuzzy Rough Algorithms in Machine Learning

As stated above, our focus is on the classification task, in which a prediction model is derived from a set of labelled observations. Nevertheless, the field of machine learning far exceeds the classification domain. In this section, we provide a brief overview of popular settings and, at the same time, indicate how fuzzy rough set theory has been used in these domains in the literature [422].

Two fundamental stages of the knowledge discovery process are data preprocessing and data mining [102]. The recent review [189] divides the former into data preparation and data reduction methods. A data preparation step converts raw data to an appropriate format by, for example, cleaning or transforming the data. Data reduction methods include such popular techniques as feature selection (reduction of the number of descriptive features, [297]), instance selection (reduction of the number of observations, [296]), feature extraction (creation of new features, [295]), instance

generation (creation of new instances, [404]) and discretization (transformation of the continuous features to categorical features, [190]). We discuss the use of fuzzy rough set theory for feature and instance selection in Sect. 1.3.1.

Once the data has been shaped into the correct format and possibly improved by a data reduction technique, the aim is to extract the hidden information it contains. Concretely, this can be the construction of a prediction model based on the available information or the discovery of patterns or structures in the observed data. Section 1.3.2 provides more detail on this step and the different subdomains where fuzzy rough set theory has been used.

General (but thorough) introductions to machine learning can be found in e.g. handbooks [155, 328]. For more details on the use of fuzzy rough set theory in this area and more complete references, we refer the interested reader to our review paper [422].

1.3.1 Fuzzy Rough Feature and Instance Selection

Feature selection has been and remains a favoured application domain of fuzzy rough set theory. Advances continue to be made. The shared aim of fuzzy rough feature selection methods is to reduce the full feature set to a strong subset with the same (or possibly better) discerning capacity. In a classification context, it is not necessary to be able to discern between each pair of instances, only if they belong to different classes. Indeed, the fundamental goal of classification is to extract an effective separation between classes and not between individual instances. The unifying study of [107] uses a general monotone fuzzy rough set based measure $\mathcal{M}$, for which different concrete alternatives are discussed, to assess the ability of a feature subset $\mathcal{B}$ to discern between elements of different classes relative to the same ability of the full feature set $\mathcal{A}$ (with $\mathcal{M}(\mathcal{A}) = 1$). Different approaches can be taken to optimize $\mathcal{M}$ in order to construct a feature subset that approaches the classification capacity of $\mathcal{A}$ to a satisfactory degree.

A first group of fuzzy rough feature selection methods are those based on a discernibility matrix. In rough set theory, the discernibility matrix is defined as an $n \times n$ matrix, n being the number of observations in the dataset, where each entry corresponds to a pair of instances and contains a list of features by which the two elements can be discerned (set to $\emptyset$ for same-class instances). As explained in Sect. 1.2.2, a discerning feature is one for which the two instances take on a different value. In order to discern between the same amount of instances as the full feature set does, it suffices that the reduced set contains one feature of each entry in the matrix. This idea forms the foundation of rough feature selection methods based on a discernibility matrix. In fuzzy rough set theory, this concept is generalized to a fuzzy discernibility matrix. As Sect. 1.2.3 described, the discernibility of instances is no longer a black-and-white matter, as they can be (dis)similar to a certain degree. As for the rough set definition of the discernibility matrix, matrix entries are a set of attributes. However, in the fuzzy setting, they are determined for pairs of sufficiently dissimilar and

opposite-class instances. The selected attributes are those for which the values of the two instances are distinct enough. After the construction of the discernibility matrix, the discernibility function can be derived from it. When this function is transformed to the correct form, it represents all feature subsets satisfying the discerning capacity requirement (as well as an additional minimality condition). Example fuzzy rough set methods include [85, 87–90, 214, 409, 498]. Unfortunately, the transformation step, which often boils down to the conversion of a function in conjunctive normal form to disjunctive normal form, can be very time consuming. Additionally, it may be sufficient to derive a single good feature subset instead of multiple ones.

To address this shortcoming, faster algorithms using search heuristics have been proposed as well. Several proposals construct the requested feature subset by optimizing the quality measure in an iterative approach. Typically, a forward search is applied that starts from an empty feature subset and adds features until a sufficiently high value of the chosen measure $\mathcal{M}$ is reached. In each iteration, the feature with the highest quality (according to some specific criteria) is added to the current set. Another option is to iteratively remove certain features from $\mathcal{A}$ until a decrease in $\mathcal{M}$ is observed. In fact, almost any search heuristic could be applied. Examples of methods incorporating this idea can be found in e.g. [43, 126, 213, 226, 227, 244, 245, 247, 349, 352, 429, 491].

Fuzzy rough instance selection methods have been proposed in [241, 360, 414, 415, 417]. They rely on the fuzzy rough approximation operators to decide which instances should be kept in the dataset and which should be removed, e.g. because they are noisy. The combination of feature and instance selection has been considered as well [122, 125, 312].

1.3.2 Fuzzy Rough Prediction Methods

A general distinction can be made between supervised and unsupervised learning. The former deals with datasets in which all observations are associated with an outcome. The aim is to use these elements to predict the unknown outcome of new observations. Supervised approaches include neural networks [47], support vector machines [377], Bayesian learning [25], instance-based learning [8], rule induction [170] and decision tree learning [366]. In unsupervised learning on the other hand, the observations are not associated with an outcome and the goal is simply to discover structures or relations in the data. We can list clustering [7] and association rule induction [2] as general unsupervised learning techniques. A third setting is semi-supervised learning, which has been introduced in Sect. 1.1.2. We provide a short overview in the following paragraphs.

Supervised learning Arguably the most natural application of fuzzy rough set theory in classification lies with instance-based learning or nearest neighbour approaches. A nearest neighbour classifier is a lazy learner, meaning that it does not construct an explicit prediction model, but rather stores the full dataset to use in the classification of a new instance [110]. It does so by locating its k nearest neighbours among the

stored prototypes and aggregating their outcomes to a prediction, commonly by means of a majority vote. As the k nearest neighbours of an instance correspond to its k most similar elements, it is clear that a nearest neighbour method at its core relies on a similarity or distance relation, as does fuzzy rough set theory (Sect. 1.2.3). Fuzzy rough nearest neighbour classifiers have been proposed in e.g. [46, 242, 353, 361, 370, 416]. These methods introduce the use of the fuzzy rough approximation operators within the nearest neighbour classification idea.

Fuzzy rough set theory has been applied in decision tree learning as well, mainly in the splitting phases of the tree generation process. A decision tree is constructed by starting from a root node and iteratively splitting nodes into several child nodes with the aim of increasing the purity of leaf nodes. Each split is based on the values of the observations for a particular feature. In traditional decision trees, the selection of a splitting feature relies on measures like the information gain or impurity reduction [155], but this is where fuzzy rough alternatives have been integrated as well [14, 44, 137, 246, 483].

Fuzzy rough rule induction methods constructing fuzzy decision rules have been proposed in e.g. [200, 221, 243, 301] and we encounter uses in support vector machines (e.g. [84, 86]) and neural networks (e.g. [180, 181, 184, 371, 372]) as well. Aside from classification, fuzzy rough regression algorithms, where the outcome is a real number rather than a class label, have been proposed as well (e.g. [16, 242]).

Unsupervised learning In the study of [280], a fuzzy rough evaluation measure for the similarity between clusters was introduced in the fuzzy c-means clustering method [42]. This clustering technique constructs c soft clusters from the original data, to which every instance has a certain membership degree. A more advanced clustering method is the self-organizing map procedure [209], to which a fuzzy rough approach was taken in [182, 183].

Semi-supervised learning Comparatively little work has been done in the area of fuzzy rough sets for semi-supervised learning. A fuzzy rough set based self-labelling approach, which extends the labelled part of the data by assigning classes to some unlabelled elements, has been proposed in [311]. We can also refer to Chap. 5 for the application and study of fuzzy rough set theory in semi-supervised classification.

1.4 Research Objectives and Overview of the Book

We focus on several challenging classification tasks and develop new machine learning techniques based on fuzzy rough set theory to solve them. We structure our work as follows:

- Chapter 2: we first review the classification domain. We discuss more theoretical aspects like the bias-variance trade-off as well as prominent examples of classification algorithms. This chapter also describes the methodology to set up experimental studies comparing groups of classifiers.

- Chapter 3: this chapter considers the ordered weighted average based fuzzy rough sets proposed in [108]. This is a noise-tolerant extension of the traditional fuzzy rough set model. As this is the model used in our proposed methods in later chapters, it warrants a deeper study. In particular, the choice of weighting scheme is an important question, as it is the key component in determining the fuzzy rough lower and upper approximations of this model. We study the suitability of different weight settings within the class approximations on a variety of datasets and aim to evaluate if, based on simple dataset characteristics, a weighting scheme selection strategy can be proposed to facilitate the use of ordered weighted average based fuzzy rough sets.
- Chapter 4: the first machine learning task we focus on is that of class imbalance, that is, an uneven distribution of observations among the classes (see Sect. 1.1.1). Some classes appear in abundance, while others are rarely encountered. This poses a challenge to traditional learners and can severely hinder the construction of a strong prediction model. We wish to develop a fuzzy rough set based classifier to handle multi-class imbalanced data and compare it against the current state-of-the-art in this domain.
- Chapter 5: in this chapter, we turn our attention to semi-supervised classification (see Sect. 1.1.2) and evaluate the strength of classifiers based on the ordered weighted average based fuzzy rough set model in this setting. Using the weighting scheme selection strategy proposed in Chap. 3, we test whether these methods can extract sufficient information from the labelled part of the data without requiring any explicit labelling step of the unlabelled observations or whether the latter is necessary to guarantee a strong performance, in particular in relation to existing proposals.
- Chapter 6: this chapter discusses multi-instance classification, briefly introduced in Sect. 1.1.3 above. We wish to develop two groups of methods: fuzzy multi-instance classifiers and fuzzy rough multi-instance classifiers. The latter will be specifically focused on class imbalanced multi-instance data. Both groups will be set up as multi-instance classifier frameworks, in which several internal parameters can be modified. We will conduct an extensive experimental study to (i) advise the user on suitable settings and (ii) compare our work to the existing methods for (imbalanced) multi-instance classification.
- Chapter 7: the last classification challenge addressed with our fuzzy rough methods is the multi-label prediction task (see Sect. 1.1.4), in which several labels are predicted at once. We aim to develop a nearest neighbour based method that relies on fuzzy rough set theory to derive a consensus prediction from the class labelsets of the neighbours of a target instance. The neighbourhood information needs to be summarized in an adequate way, for which we deem the fuzzy rough set model an ideal tool. As before, we will thoroughly compare our proposal to existing nearest neighbour based multi-label classifiers.
- Chapter 8: to conclude this work, the final chapter summarizes our proposals and findings and outlines future research directions.

Chapter 2
Classification

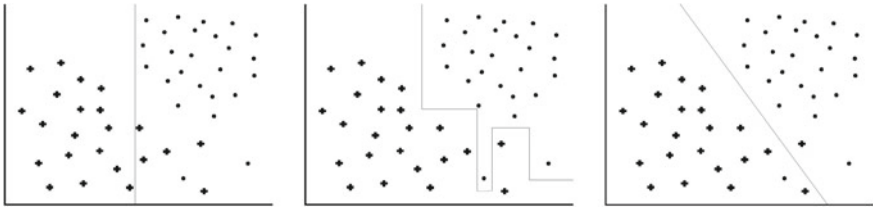
In this chapter, we review the traditional classification domain, the supervised learning task on which this book focuses. Before addressing several challenging classification problems in the next chapters, we first review the core aspects of this popular research area, as would be done in any machine learning course or handbook. Section 2.1 provides a brief and general introduction, explaining often referenced aspects as the bias-variance trade-off and the curse of dimensionality. In Sect. 2.2, we review several groups of classification algorithms, representatives of which are included in the experimental studies conducted throughout this work. The specifics on how to properly set up classification experiments are laid out in Sect. 2.3.

2.1 Introduction

In traditional classification tasks with input space X, an element $x \in X$ can be represented as a feature vector of length $|\mathcal{A}|$, with $\mathcal{A}$ the set of descriptive features. The ith position in this vector corresponds to the value of instance x for the ith attribute. This allows for an easy organization of classification data into the flat table format represented in Table 1.1. Each row corresponds to one observation $x \in X$. The last column is commonly reserved for the class label, while the first $|\mathcal{A}|$ columns contain the input feature values.

The classification task is to construct a prediction model based on the information contained in a *training or learning set* of labelled samples. More concretely, if X is the input space and $\mathcal{C}$ the set of possible classes, the aim is to learn a mapping $f : X \rightarrow \mathcal{C}$ based on training set $T = \{(x, C_x) \mid x \in X, C_x \in \mathcal{C}\}$. Several approaches to the derivation of f have been explored in the literature and will be discussed in Sect. 2.2. Each of them makes internal assumptions on how the information in the training set should be modelled, with the shared aim to find an appropriate approximation of

S. Vluymans, *Dealing with Imbalanced and Weakly Labelled Data in Machine Learning using Fuzzy and Rough Set Methods*, Studies in Computational Intelligence 807, https://doi.org/10.1007/978-3-030-04663-7_2



(a) Underfitting (b) Overfitting (c) A better boundary

Fig. 2.1 Illustration of a two-class datasets (circles and plus signs) with three possible derived decision boundaries

the decision boundary between classes. Apart from extracting sufficient information from the training set, the model should also be able to generalize well to unseen test data and make accurate predictions. A highly complex model can fit the training data perfectly, but, by modelling all peculiarities in this set, can lead to classification errors on new data. This phenomenon is known as *overfitting*, where the model almost literally learns the training data by heart. On the other end of the spectrum, *underfitting* the data leads to a simple (and probably highly interpretable) model, that will, unfortunately, also fail on new data because insufficient information has been learned from the provided labelled samples. An example of these two situations is depicted in Fig. 2.1, which represents a two-class two-dimensional dataset. The horizontal and vertical axis correspond to the first feature a_1 and second feature a_2 respectively. In Fig. 2.1a, a simple decision boundary is derived, only based on the first feature a_1. All elements with an a_1 value lower than the selected threshold (left of the vertical line) would be classified as a plus sign, while those with an a_1 value exceeding the threshold will be assigned to the circle class. This is a very simple and easy-to-understand decision rule, but can easily lead to misclassifications on both classes. Figure 2.1b represents a situation where the data has been overfitted. A highly complex boundary has been constructed to perfectly separate the elements of the training set. The third option represented by Fig. 2.1c likely forms the best compromise between model interpretability, generalization capacity and training information extraction. A linear decision boundary based on both features (sloped line) has been derived. It does not form a perfect partition of the classes in the training set, but probably results in better predictions on new data.

The issue discussed above is related to balancing accuracy and interpretability as well as the *bias-variance trade-off* [164]. A highly complex model with a large number of parameters to learn will depend more on its training data and can, consequently, be very different when constructed from a slightly modified training set. This model has a high *variance* and can exhibit a highly different classification behaviour after small changes in its training set. A very simple model will not suffer from this variance issue, but introduces a *bias* to a certain (simple) structure instead. It can be shown that the expected prediction error of the learned model can be decomposed in an expression containing error terms representing the bias and variance. Ideally, both components should therefore be minimized. Unfortunately, lowering the bias

of a model usually increases its variance and vice versa. A learning algorithm should balance these two aspects in order to have a strong prediction performance.

Another issue associated with learning from data is the so-called *curse of dimensionality*. The term was introduced in [36] and refers to the fact that the volume of the Euclidean space increases exponentially when more dimensions are added [258]. High-dimensional spaces are inherently sparse (also called *empty space phenomenon*). This (negatively) affects the suitability of certain classification approaches in high-dimensional data as well as the intuition behind and interpretability of the derived prediction models. For example, any notion of locality is lost, rendering nearest neighbour based methods inappropriate [41]. This is of particular relevance to the work presented in this book, because fuzzy rough set based classification algorithms are closely related to nearest neighbour approaches, as they also strongly depend on a distance or similarity relation between observations.

2.2 Classification Models

A wide spectrum of classification algorithms have been developed in the literature. In this section, we discuss some popular and widely-used approaches and categorize them according to their prediction rationale. Representatives of these groups will be used in the experimental studies conducted in later chapters. We consider nearest neighbour classification (Sect. 2.2.1), decision trees (Sect. 2.2.2), linear models (Sect. 2.2.3), neural networks (Sect. 2.2.4), rule based models (Sect. 2.2.5), probabilistic models (Sect. 2.2.6) and ensemble classifiers (Sect. 2.2.7).

2.2.1 Nearest Neighbour Classification

Arguably one of the most intuitive classification algorithms is the k nearest neighbour method (kNN, [110]), with $k \geq 1$ an integer number. It is a *lazy learner*, which means that it does not build an explicit classification model or mapping from the input space to the set of possible classes. Instead, all training elements are stored in memory. These elements are also referred to as *prototypes*. To classify a new element x, the k elements nearest to it among the stored prototypes are determined. The class that appears most often among these neighbouring elements is predicted for x. When several classes tie for the most appearances, one among them is usually selected at random.

The notion of *nearest* is enforced by an appropriate distance or similarity measure. The nearest neighbours of x are the k elements with the smallest distance to or largest similarity with x. The traditional Euclidean distance remains a popular distance measure, although alternatives like the HVDM distance [448] are used as well. Another option is to learn an appropriate distance measure (often in the form of

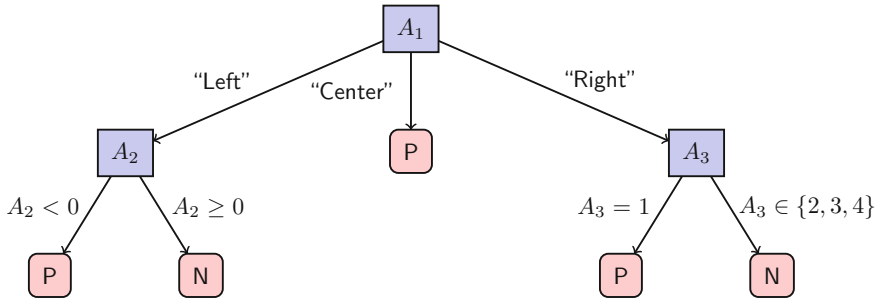


Fig. 2.2 Example of a classification tree prediction whether an instance belongs to the positive (P) or negative (N) class

a Mahalanobis distance [314]) from the data by means of a metric learning procedure [34, 445]. As the kNN classifier entirely relies on distance calculations, it can greatly benefit from the optimization of this measure to fit the data at hand.

In the procedure described above, each selected neighbour has an equal vote in the class decision of x. An often used alternative is to weigh the contribution of each neighbour based on its distance to the target [133]. To this end, neighbour weights inversely proportional to this distance are imposed. As a result, more distant neighbours influence the class prediction of x less than near ones.

2.2.2 *Decision or Classification Trees*

Another popular classification approach is to construct a decision tree based on the training data [366]. A tree is a connected, acyclic graph. Every internal node corresponds to a test based on one of the input features. The leaf nodes, which form end points on paths starting from the root, represent class assignments. To classify a new element, it is passed through the tree, starting at the root node and following the correct path towards a leaf. The class linked to the leaf is assigned to the test element. An example decision tree is depicted in Fig. 2.2. The first test is made based on the categorical attribute A_1. Depending on the feature value, the left, middle or right path is chosen. The left path leads to a second split based on the numeric feature A_2. Any value lower than zero leads to an assignment to class P, while elements with positive A_2 values are classified as N. On the right hand side, the second split is based on feature A_3.

The construction of a decision tree follows a top down procedure. At the outset, the tree consists of the root only and all training instances are grouped together in this node. Based on an *impurity measure* (assessing the current class distribution) the best feature and corresponding split are selected. This split is performed and nodes are created for each possible outcome of the test. The training set is split into subsets corresponding to the division of elements obtained after the test. When a

node contains elements of only one class (a *pure* node), it is retained as a leaf node and labelled with this class. If elements of more than one class are still present, the procedure is repeated and further splits are created starting from this node.

In order to create small trees, the impurity measure guides the split selection process towards feature splits that reduce the class heterogeneity of elements associated with a node. These measures are based on the class distribution in these nodes. When the classes are uniformly distributed within a node, the impurity is maximal. In the presence of only one class, a minimal impurity is found. The CART decision tree algorithm of [60] uses the Gini index as impurity measure, while the ID3 [354] and C4.5 [356] algorithms rely on Shannon's entropy [379].

If a tree is constructed up to the point when all leaves are pure, there is an increased risk of overfitting the training data and obtaining an overly complex model. A solution to this problem is to apply a pruning procedure [355]. We can discern between pre-pruning and post-pruning. Pre-pruning prematurely halts the tree growing process when a node is considered pure enough or represents too few training instances. Post-pruning allows the full construction of the tree until pure leaf nodes are obtained. Afterwards, branches are removed from the tree to minimize the classification error on a validation set.

2.2.3 Linear Models

Linear models construct a hyperplane to represent a linear separation between classes. These methods have initially been designed for two-class classification problems. In a d-dimensional space, a hyperplane is a subspace of dimension $d - 1$. In the two-dimensional plane, a hyperplane is a straight line. In the three-dimensional space, a hyperplane is a regular plane.

When two-class data is linearly separable, there exist multiple hyperplanes such that all elements of one class are located on one side and all elements of the second class can be found on the other. However, not all separating hyperplanes are equally suitable, as represented in Fig. 2.3. The three straight lines each form a perfect separation between the two classes, but the middle one should be favoured. The proximity of the other two to the training elements is more likely to induce classification errors on test data. The support vector machine (SVM) classifier [54, 109, 112] implements this idea by maximizing the margin defined by the separating hyperplane. The margin can be interpreted as the (symmetric) empty space around the hyperplane before a training element is encountered. In Fig. 2.3, it is clear that the middle hyperplane induces the widest margin between the two classes and is therefore the best option. The search for this hyperplane is formulated as a constrained quadratic optimization problem that can be solved using Lagrange multipliers [40]. To deal with data that is not linearly separable, the soft-margin SVM [109] introduces slack variables that allow elements to be located on the incorrect side of the hyperplane. These slack variables are added to the optimization objective weighted with a penalty term C (the complexity parameter). Another (or additional) way to address the non-linearity of

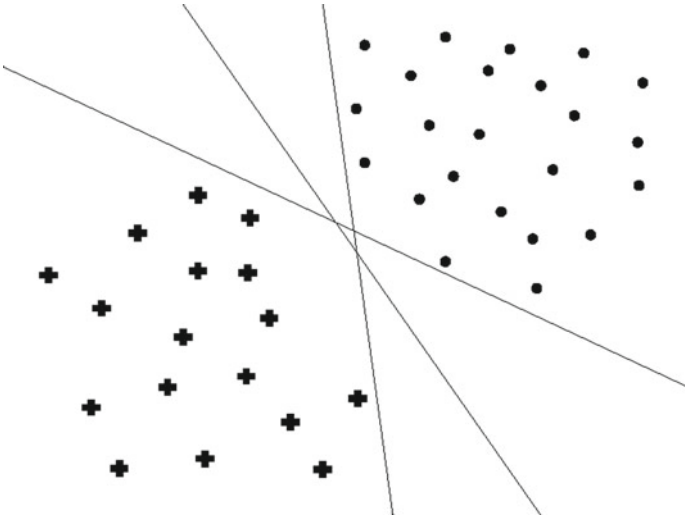


Fig. 2.3 Three separating hyperplanes between two classes

data is to apply an implicit mapping to a different (higher-dimensional) feature space. It can be shown that the SVM formulation and optimization contains only vector dot products, a situation in which the *kernel trick* can be applied to perform calculations in the induced feature space [9, 54]. Example kernel functions are listed in [67]. SVM algorithms are popular and widely used in the machine learning community.

2.2.4 Neural Network Classification

Another linear classification model (but handled in a separate section because of the family of neural network classifiers [47] it inspired) is the perceptron algorithm [337, 367]. The normal vector **w** of the separating hyperplane is randomly initialized. Afterwards, each training element is processed and the values in **w** are updated to ensure the correct classification of the element. This process is repeated until convergence, that is, until no more changes are made to **w**. Despite the initial enthusiasm of the research community towards this proposal, [327] pointed out some important limitations of the simple perceptron. These issues were addressed by the later development of the multi-layer perceptron and the back-propagation algorithm [368]. A perceptron can be modelled as a neural network with an input and output layer, while a multi-layer perceptron contains one or more hidden layers. It is a feedforward neural network, where the inclusion of the hidden layers and use of non-linear activation functions allows the modelling of non-linearity in data. An example network is presented in Fig. 2.4. Each connection between nodes is associated with a weight, determined at training time by means of the back-propagation

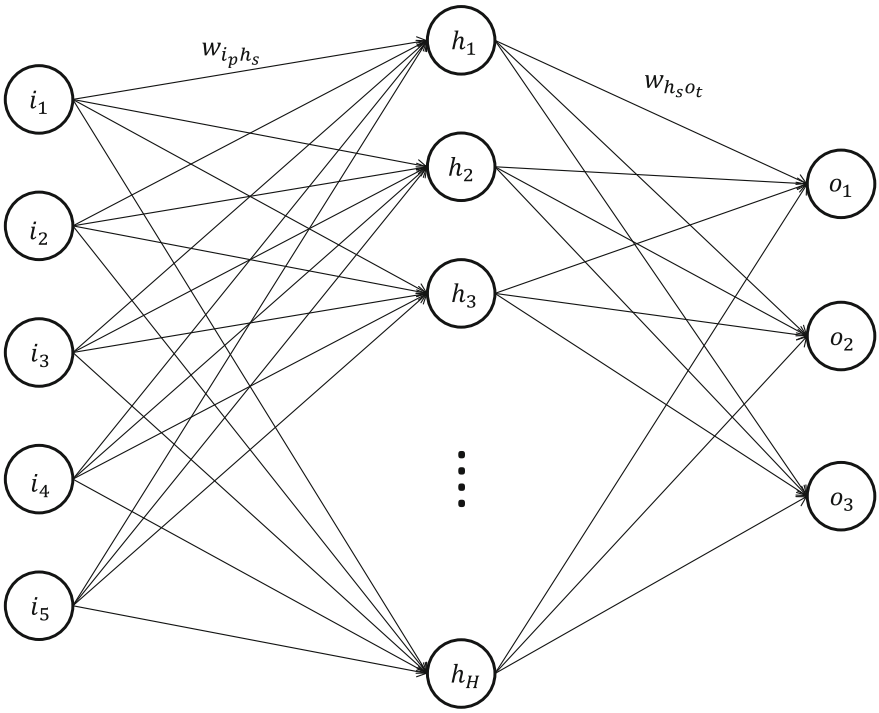


Fig. 2.4 Example multi-layer perceptron with one hidden layer. The input layer contains five nodes $(i_1, \ldots, i_5)$, the hidden layer H nodes $(h_1, \ldots, h_H)$ and the output layer three nodes (o_1, o_2, o_3). The connection weights w_{i_p,h_s} $(p = 1, \ldots, 5$ and $s = 1, \ldots, H)$ are associated with the edges between input and hidden nodes, while weights w_{h_s,h_t} $(s = 1, \ldots, H$ and $t = 1, \ldots, 3)$ correspond to connection between the hidden layer and the output nodes

procedure. With respect to classification, a neural network classifier for instances in a d-dimensional space often contains d input nodes (one for each feature) and m output nodes, where m is the number of possible classes. The trained network computes one value per output node. The presented instance is assigned to the class corresponding to the node with the highest value. Research in neural networks has been inspired by the biological neural processes and simulates how neurons work together in the human brain.

The universal approximation theorem implies that a multi-layer perceptron with one hidden layer, final number of nodes and non-linear activation functions can closely approximate any commonly used function [169, 222]. However, the number of hidden nodes required for this task may be large, resulting in an enormous amount of weight parameters to be trained. In recent years, deep learning and deep neural networks have emerged as popular and widely discussed research topics [38, 276, 376]. Deep neural networks contain several hidden layers. As the number of nodes required per layer can be kept modest, this can result in an exponential

reduction in the number of connection weights that need to be trained [38, 286]. As a consequence, complex functions can be modelled more efficiently, addressing the issue related with multi-layer-perceptrons mentioned above. Deep neural networks automatically extract features from the data in the form of feature hierarchies. This automation relieves the scientist of the laborious task of determining adequate features, but does imply a lack of interpretability of the final classification model.

2.2.5 Rule Models

A rule based classifier derives a set of rules from the dataset and uses these as its prediction model [170]. A rule can be written in the form 'IF P_1 and P_2 AND …AND P_t THEN C'. In this expression, the P_i variables ($i = 1, \ldots, t$) are rule premises based on the input features and C is the consequent, a class prediction. As an example, the decision tree in Fig. 2.2 can be converted to the following set of rules:

$$\begin{aligned}
\text{IF} A_1 == \text{Left AND } A_2 < 0 \text{ THEN } class\ P \\
\text{IF} A_1 == \text{Left AND } A_2 \geq 0 \text{ THEN } class\ N \\
\text{IF} A_1 == \text{Center THEN } class\ P \\
\text{IF} A_1 == \text{Right AND } A_3 == 1 \text{ THEN } class\ P \\
\text{IF} A_1 == \text{Right AND } A_3 \in \{2, 3, 4\} \text{ THEN } class\ N
\end{aligned}$$

A rule list learner constructs its rule set in a top down way, much like the construction of a decision tree (Sect. 2.2.2). Premises are added to a rule until all training elements satisfying the combined rule belong to a single class. The selection of an appropriate additional premise can be guided by an impurity measure. Once all instances covered by the combination of premises belong to the same class, this class prediction is used as the consequent of the rule. After a rule r has been learned, the set of training samples covered by it can be removed, such that the remaining rules are learned from samples not covered so far. Traditional rule list learning algorithms are CN2 [104] and Ripper [105]. Alternatively, a class based approach can be followed, where a rule set is learned for one class at a time. In the construction of a rule r for class C, the consequent is already fixed and only class C is considered in the impurity measure for the premise selection process [103].

2.2.6 Probabilistic Models

Probabilistic models for classification rely on probability distribution estimations to predict the class variable. We can make a distinction between discriminative and generative classifiers [48, 335]. If variables X and Y represent the input features and outcome respectively, the former algorithms aim to model the posterior probability

$P(Y \mid X)$. A class probability score is derived based on the input feature values and the class with the highest score can be used as prediction. Two example methods are described in the following paragraphs. Generative classifiers on the other hand learn the joint distribution $P(X, Y)$. By sampling from this joint distribution, new labelled data elements can be created.

The naive Bayes classifier is based on Bayes' rule in probability theory to compute the conditional probability $P(Y \mid X)$ [278]. An instance x is classified to the class C (that is, value C for outcome variable Y) for which $P(Y = C \mid X = x)$ is largest. Bayes' theorem states $P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$, such that $P(Y = C|X = x) = \frac{P(X=x \mid Y=C)P(Y=C)}{P(X=x)}$. Since the denominator in this expression is independent of the class value, the classification reduces to maximizing $P(X = x \mid Y = C)P(Y = C)$. The naive Bayes method assumes complete conditional independence among the dimensions of X (that is, among the descriptive features), such that we can decompose the objective function as $P(Y = C) \prod_{i=1}^{d} P(X_i = x_i \mid Y = C)$. As a result, the multi-dimensional probability estimates are reduced to easier-to-handle one-dimensional ones. Despite its strong independence assumption, a naive Bayes classifier can often perform quite well [129].

Another popular probabilistic model is called logistic regression [111]. In spite of what its name may suggest, this is a classification model used to predict the categorical outcome of instances. Originally devised for binary problems with classes C_1 and C_2, the posterior class probabilities are computed as $P(Y = C_1 \mid X = x) = \frac{\exp(\beta_0+\beta_1 \cdot x)}{1+\exp(\beta_0+\beta_1 \cdot x)}$ and $P(Y = C_2 \mid X = x) = \frac{1}{1+\exp(\beta_0+\beta_1 \cdot x)}$, in which the products $\beta_1 \cdot x$ should be understood as vector dot products. The regression coefficients (β_0 and vector β_1) can be estimated in a maximum likelihood procedure. The decision boundary can be rewritten as an expression where the log-odds of the posterior probabilities equals zero, namely

$$\log\left(\frac{P(Y = C_1 \mid X = x)}{P(Y = C_2 \mid X = x)}\right) = \beta_0 + \beta_1 \cdot x = 0.$$

This implies a linear decision boundary with respect to the log-odds. Multinomial logistic regression generalizes this procedure to datasets with more than two classes [223].

2.2.7 *Ensemble Classification*

Ensemble learning methods [63, 270, 504] construct several classification models from the same training set. This can be achieved by using different learning algorithms, making multiple modified versions of the training set or both. To classify a new instance x, each member of the ensemble computes a prediction for x and the final class assignment is derived by means of an aggregation procedure (e.g. a majority vote). It is beneficial to aim for a level of diversity between the ensemble members in order to capture different classification behaviour [272].

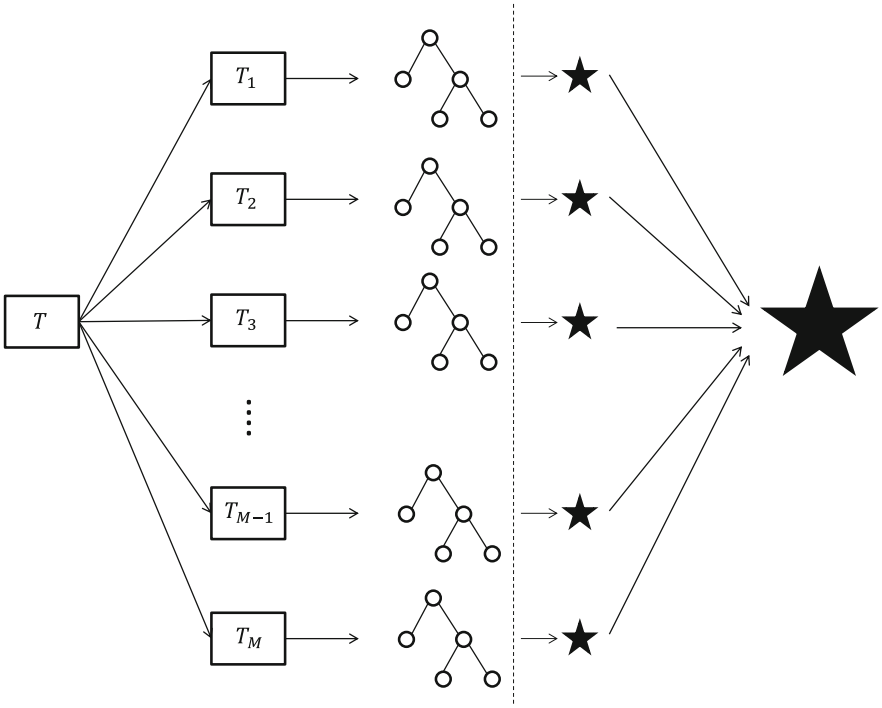


Fig. 2.5 Random forest classification model. The training process is depicted left from the vertical line. A total number of M decision trees are trained from bootstrap samples from the training set. On the right hand side, which represents the classification of an instance, each tree provides a prediction and a final classification is obtained by means of aggregation

A simple ensemble construction process is called *bagging* [58], short for bootstrap aggregating. Let M be the number of ensemble members. Each classifier is constructed using the same learning algorithm (e.g. a decision tree), but by training it on a different dataset. If T is the original dataset, the M training sets are constructed by sampling T with replacement until $|T|$ elements are obtained. A popular example incorporating the basic bagging technique is the random forest classifier [59]. It builds a decision tree on each of the induced training sets and uses an additional subspace sampling procedure to reduce the feature set to a different subset in each tree. Figure 2.5 provides a graphical presentation of a random forest model.

A second popular ensemble technique is *boosting* [373, 374]. It is a sequential approach that iteratively trains a classification algorithm on reweighed versions of the training set. At the end of an iteration, the weight of misclassified training elements is increased, such that they receive relatively more attention in the next step. The process is repeated for M iterations. Each constructed model is assigned a confidence score based on its training error, that weighs its contribution to the final aggregated prediction. The traditional boosting algorithm is called AdaBoost [162, 375].

Having constructed an ensemble of classifiers, not all members may be equally suited to classify every test instance. They may work particularly well in certain parts of the feature space, but produce poor or irrelevant results in others. Several procedures have been proposed to dynamically decide which classifiers should be used based on the test instance at hand. We can discern between dynamic classifier selection [113, 194] and dynamic ensemble selection [263]. Per test instance, the former selects one classifier, while the latter selects several ensemble members.

2.3 Conducting Classification Experiments

Many research papers in the machine learning domain focus on the development of a new method for a particular problem or application. Aside from a rigorous explanation and motivation, such studies also need to prove the advantages of their newly proposed approach over existing alternatives. To this end, the proposal is usually compared to an appropriate selection of state-of-the-art methods in an experimental evaluation on a sufficient number of datasets. Several online archives of datasets exist, including the popular UCI [287] and KEEL [11] dataset repositories. Using publicly available datasets enables other researchers to reproduce or extend the reported study.

A key aspect of any experimental study is the selection of one or more appropriate evaluation metrics (Sect. 2.3.1) that measure complementary features of the method performance. Secondly, a validation procedure (Sect. 2.3.2) is required in order to ensure reliable and representative results. Thirdly, once experimental results are obtained for all included methods on all datasets, they need to be adequately compared to each other. A crucial step towards a confident conclusion is a statistical analysis of the results (Sect. 2.3.3). We discuss these components in separate sections below.

2.3.1 Evaluation Measures

In this work, we focus on classification applications, where observations are labelled with a categorical outcome (their class). A classifier assigns a class label to a newly presented instance based on a previously learned prediction model. Its performance can be evaluated in several ways. In Sect. 2.3.1.1 we describe a number of traditional measures. Sect. 2.3.1.2 explains that some measures are unsuitable when the class distribution is skewed and presents some imbalance-resistant alternatives. Throughout this book, we take special care to select suitable evaluation measures for the studied problems. In particular, custom evaluation measures for multi-instance and multi-label classification are used in Chaps. 6 and 7 and their definitions are recalled there.

2.3.1.1 General Evaluation Measures

The intuitively most relevant facet to evaluate is the correctness of the predictions made. Let Ts be a test set of elements for which class labels were predicted and $corr(\cdot)$ a function that counts the number of correct predictions for its set argument. Probably the most commonly used evaluation measure is still the traditional global classification *accuracy*. It is defined as

$$acc(Ts) = \frac{corr(Ts)}{|Ts|}$$

and measures the ratio of correctly classified instances in the test set. The complement of the accuracy is referred to as the *error rate* and is given by

$$err(Ts) = 1 - acc(Ts) = \frac{|Ts| - corr(Ts)}{|Ts|}.$$

In binary classification problems, only two classes are present and can often be interpreted as 'positive' and 'negative'. A basic two-class *confusion matrix* can be constructed that summarizes the prediction behaviour as presented in Table 2.1. The rows and columns correspond to the actual and predicted classes respectively. The entries correspond to the number of *true positive* (TP), *true negative* (TN), *false positive* (FP) and *false negative* (FN) predictions. The former two can be normalized to the *true positive rate* (TPR) and *true negative rate* (TNR) as

$$TPR = \frac{TP}{TP + FN} \quad \text{and} \quad TNR = \frac{TN}{TN + FP}.$$

In some applications, the *false positive rate* (FPR) and *false negative rate* (FNR), defined as

$$FPR = \frac{FP}{TP + FN} \quad \text{and} \quad FNR = \frac{FN}{TN + FP},$$

may be of interest as well.

The TPR is also called *recall* (r) or *sensitivity*. A complementary measure from the information retrieval domain is the *precision* (also called *confidence* or *positive predictive value*), defined as

$$p = \frac{TP}{TP + FP}.$$

Table 2.1 Confusion matrix obtained after classification of a two-class dataset

Actual & Predicted	Positive	Negative
Positive	TP	FN
Negative	FP	TN

While the recall r measures the rate of correctly classified positive instances, the precision p represents the fraction of positive predictions that are correct. To evaluate the trade-off between these two aspects, the *F_β-measure*, defined as

$$F_\beta = (1 + \beta^2) \cdot \frac{p \cdot r}{\beta^2 \cdot p + r},$$

can be applied. Parameter β can take on any positive real value, but is usually set to $\beta = 1$. In the latter case, the measure is simply referred to as the *F-measure* and is given by

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

the harmonic mean of precision and recall. A confusion matrix similar to the one in Table 2.1 can be constructed for datasets with more than two classes as well. For a dataset with m classes, the entry on the ith row and jth column of the $m \times m$ matrix lists the number of class i elements that were assigned to class j in the prediction step.

Another component commonly assessed when comparing machine learning methods is their complexity. Several aspects can be captured under this general term. A first one is the theoretical complexity related to the construction of the prediction model as well as to the prediction of the outcome of new instances. Their theoretical complexity has a direct influence on the runtime of these two processes, an essential practical consideration. It is usually represented using so-called big-O notation, which indicates how the runtime of an algorithm increases with, for example, the size of the input (number of instances) or its dimensionality (number of features). Secondly, the actual complexity or understandability of the learned prediction model is often of interest as well. Even though the computer algorithm is somehow applied as a black box in the prediction process, the interpretability of the intermediate model is still important. Easy-to-understand classification rules can for instance lead to new insights on the application at hand.

2.3.1.2 Evaluation Measures in the Presence of Class Imbalance

The specific properties of a problem may require custom evaluation measures. For instance, it has been established in the machine learning community that the regular classification accuracy is inappropriate to use in the presence of class imbalance (e.g. [391], Sect. 1.1.1, Chapter 4). As an example, consider a two-class dataset in which 100 instances belong to the first class and 900 to the second and a classifier that predicts the second label for all elements. The accuracy value of this evaluation would be 90%, since 900 out of 1000 instances are classified correctly. This measure does not take the imbalance between classes into account. It can provide deceptive results and lead to unreliable conclusions. In particular, a strong performance on a majority class can easily overshadow a poor result on the minority class in the

accuracy calculation. In this toy example, the 90% accuracy rate does not adequately reflect the poor performance on the first class, which has been misclassified entirely.

Example alternatives like the *geometric mean (gmean)* or *balanced accuracy* (also called *average accuracy*) aggregate the class-wise accuracies to counteract the dominance of the majority class. The former does so by taking the geometric mean of the class-wise accuracies

$$gmean(Ts) = \sqrt[m]{\prod_{i=1}^{m} \frac{corr(Ts_i)}{|Ts_i|}},$$

with Ts_i the subset of instances belonging to class i, while the latter uses their arithmetic mean

$$balacc(Ts) = avgacc(Ts) = \frac{1}{m}\left(\sum_{i=1}^{m} \frac{corr(Ts_i)}{|Ts_i|}\right).$$

Another popular evaluation measure to use in the presence of class imbalance is the *Area under the ROC-Curve* (AUC). For a binary classification problem, a Receiver Operator Characteristics (ROC)-curve is defined in the unit square and models the trade-off of a classifier between its TPR and FPR [141]. The area between the curve and the horizontal axis is used to capture the represented graphical information in one value. An algorithmic way to compute the AUC is by means of the trapezoid rule [141, 319]. Instead of deriving an exact class assignment, the classifier calculates, for each instance x, the probability $p_+(x)$ that x belongs to the positive class. Afterwards, these values are sorted and each one is used as a threshold θ, such that only instances with $p_+(x) \geq \theta$ are finally classified as positive. This procedure leads to specific TPR and FPR values for each threshold, that can be plotted as points in a Euclidean coordinate system as represented in Fig. 2.6. For example, at threshold $\theta = 0.7$, eight of the ten positive instances and five of the ten negative instances are classified as positive, respectively leading to a TPR of 0.8 and FPR of 0.5 and together to the point with coordinates (0.5;0.8) in the plot. The area underneath the curve connecting these points can be computed as the sum of the areas of a triangle and a sequence of trapezoids.

Several extensions of the AUC measure to datasets with more than two classes have been proposed in the literature. An often used example is the mean AUC (MAUC, [205]) that computes the overall AUC as the average of each of the binary AUC values between pairs of classes. In [151], the ROC-curve was extended to a surface and the AUC measure was replaced by the volume underneath it. The authors of [207] introduced the AUCarea measure. The binary AUC values between class pairs are plotted in a polar diagram and the area within the figure is used as metric. A normalized version of the AUCarea measure was proposed in [203].

Class	p_+	Class	p_+
+	0.9	-	0.7
+	0.9	+	0.7
+	0.9	-	0.7
+	0.9	-	0.3
-	0.9	+	0.3
+	0.9	-	0.3
-	0.9	+	0.1
+	0.7	-	0.1
+	0.7	-	0.1
-	0.7	-	0.1

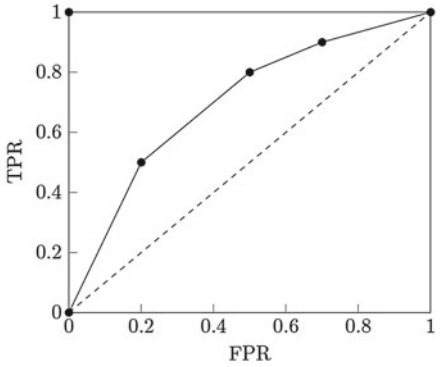


Fig. 2.6 Example of how the calculation of $p_+(x)$ values can lead to a ROC curve

2.3.2 Validation Techniques

In the classification setting, a prediction model is derived from a set of available instances (the *training set*) to use in the label prediction of newly presented instances. The class labels of the training elements are known, so when the prediction model is applied to this set, its performance can be evaluated by comparing the predicted labels to the known classes. However, this is insufficient. The classifier may have overfitted the training set, learned its eccentricities by heart as it were, and only perform well on these particular elements. An independent *test set*, not used during training, allows to assess the generalization capacity of a classifier to unseen data. To check the quality of classifiers, it is necessary to know the true class of these new instances as well, such that the predicted outcome can be compared to the ground truth. Such a reliable assessment of the classification strength is assisted by a validation scheme. We list several commonly used techniques below and refer the reader to e.g. [260, 264] and any machine learning handbook (e.g. [155]).

The most basic validation scheme is *holdout* that divides the dataset into two parts, one for training and one for testing. A typical division is to assign two thirds of the observations to the training set and the remaining third to the test set. A prediction model is derived from the training instances and used to predict the outcomes of the test elements. The *repeated holdout* scheme repeats this procedure a number of times, each time selecting a different subset of the original data as training set.

Another alternative is the *bootstrapping* validation scheme, which uses several bootstrap samples of the dataset. Such a sample is created by sampling the original dataset with replacement until a new dataset of equal size is obtained. The replacement sampling implies that duplicate samples can occur. Each bootstrap sample is used as a training set and the outcome is predicted for the original full dataset. This implies an overlap between the training and test sets, which should be taken into account with the estimation of the true prediction performance, for instance by using the .632+ bootstrap estimator [136].

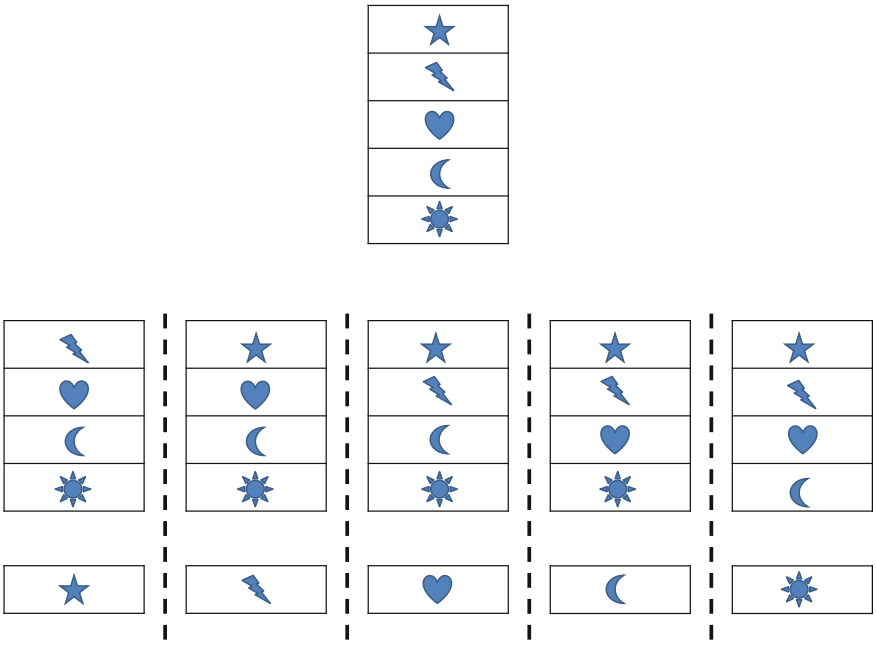


Fig. 2.7 Visualization of the five-fold cross validation procedure. For each of the five experiments, one of the five folds is kept separate as a test set, while the union of the other four is used for training

The most commonly used validation procedure is *K-fold cross validation*, with K a positive integer number. An example with $K = 5$ is represented in Fig. 2.7. The dataset is first split into K roughly equal non-overlapping parts (folds). Next, K experiments are conducted, where $K - 1$ folds are combined to form the training set and the remaining fold is used as test set. The prediction performance is evaluated on each test set and aggregated to a single final assessment value. Values of five and ten are common choices for K, although a general recommendation for setting this value remains difficult [492]. Another important question is how the dataset is divided into folds. The simplest approach is to use a random partition, although it may be prudent to respect properties of the original dataset as much as possible, such as its class distribution (*standard stratified cross K-fold validation*). More advanced partitioning strategies include distribution balanced stratified cross validation [482] and distribution optimally balanced stratified cross validation (DOB-SCV, [330]). The use of the latter has been recommended for imbalanced datasets in [307], since it avoids the critical data-shift problem, an issue referring to the possible difference in data distribution between training and test folds. The fold construction procedure in DOB-SCV distributes near same-class elements across the folds in order to adequately represent each region in all partitions. Finally, in *repeated cross validation*, as one does in repeated holdout, the full process is repeated for the same dataset several times with a different fold partitioning each time.

2.3.3 Statistical Analysis

After the experiments have been completed, a confident conclusion on the advantages of one method over the other(s) needs to be formulated. To do so for a particular evaluation measure E, the obtained results are usually grouped in the format presented in Table 2.2. The rows correspond to the n different datasets ($D_1, D_2, \ldots, D_n$) and the columns to the q included methods ($M_1, M_2, \ldots, M_q$). Entry $E(D_i, M_j)$ represents the performance of method M_j on dataset D_i as measured by E.

Column-wise averages can be computed to easily aggregate the performance of each method to a single value. When E is a measure to be maximized (minimized), intuition may dictate that the method corresponding to the highest (lowest) average value exhibited the best performance in the study. However, this is not a prudent procedure. A high (low) average may be the result of an outlying performance on a single dataset. Instead of averages, all individual $E(D_i, M_j)$ values need to be analysed. This can be achieved by means of a statistical analysis. The major dividing difference between statistical tests is their parametric or non-parametric nature. Tests belonging to the latter group are also called distribution-free, as they do not make any assumptions on the underlying distribution of the observations, while parametric tests commonly assume normality of the data. Although parametric tests usually have higher statistical power than their non-parametric alternatives when their assumptions are met, the use of non-parametric statistical analysis in machine learning has been explicitly recommended in e.g. [120, 124, 187]. The latter generally rely on the more robust median (as opposed to the mean) as centrality measure. The study of [120] specifically recommended the use of the Wilcoxon test for pairwise comparisons and the Friedman test combined with a post-hoc analysis for comparisons between multiple methods. We discuss these in the following paragraphs and use them throughout the book.

Pairwise comparisons The *Wilcoxon signed ranks test* [447] can be used to compare the performance of two methods M_1 and M_2 to each other. It is based on the differences in performance of the two methods on all datasets (that is, the values $d_i = E(D_i, M_1) - E(D_i, M_2)$). Its null hypothesis is that the two methods exhibit no difference in performance or, concretely, that the differences are distributed sym-

Table 2.2 Collection of experimental data for evaluation measure E

	M_1	M_2	…	M_q
D_1	$E(D_1, M_1)$	$E(D_1, M_2)$	…	$E(D_1, M_q)$
D_2	$E(D_2, M_1)$	$E(D_2, M_2)$	…	$E(D_2, M_q)$
⋮	⋮	⋮	⋱	⋮
D_n	$E(D_n, M_1)$	$E(D_n, M_2)$	…	$E(D_n, M_q)$

metrically around their median, such that one of the methods does not consistently outperform the other.

To assess whether the null hypothesis can be rejected based on the observations at hand, a ranking method is applied. The absolute values of the differences d_i are ranked from 1 to n, where the smallest difference is assigned the smallest rank. When several values tie, all are assigned the same rank. For example, if two values tie for ranks 9 and 10, they are both assigned rank 9.5. Two rank sums are computed: R^+ corresponding to the positive differences ('M_1 beats M_2') and R^- corresponding to the negative differences ('M_2 beats M_1'). Ranks of zero differences ($d_i = 0$) are evenly divided among R^+ and R^-, such that

$$R^+ = \sum_{d_i>0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i) \quad \text{and} \quad R^- = \sum_{d_i<0} rank(d_i) + \frac{1}{2}\sum_{d_i=0} rank(d_i).$$

The test statistic is set to $t = \min(R^+, R^-)$. When the associated p-value, which is defined as the probability that a more extreme observation than t is observed when the null hypothesis holds, is smaller than the given significance level α, the null hypothesis of equivalent performance is rejected.

Multiple comparisons In many practical settings, one wishes to compare a group of methods among each other (Table 2.2 with $q > 2$). This can be achieved in a two-step procedure.

First, the *Friedman test* [165] is applied to decide whether any statistically significant differences are present among the methods. This test assesses whether there are at least two methods in the evaluated group with significantly different median values. It does so by ranking the performance of the methods for each dataset. For dataset D_i, the method with the best value $E(D_i, M_j)$ is assigned rank 1, the second best is assigned rank 2 and so on, until the worst performing method is assigned rank q. In case of ties, average ranks are assigned to all tied methods, as was done in the difference ranking by the Wilcoxon test described above. Next, the mean rank of each method across all datasets is computed. We denote this value as its Friedman rank. Since the null hypothesis assumes equivalent performance of the q methods, their mean ranks should be approximately equal. The Friedman test statistic is derived from these ranks. When its associated p-value is smaller than the significance level α, the null hypothesis is rejected and we conclude that significant performance differences do exist among the q methods. However, the Friedman test does not show where these differences occur. To extract this information, a post-hoc test is required.

The *Holm post-hoc procedure* [219] uses the method with the lowest Friedman rank as control method and compares it to each of the $q - 1$ other algorithms. The Friedman ranks of method M_i and the control method are used to derive an unadjusted p-value p_i. These p-values can be reordered such that

$$p_1 \leq p_2 \leq \ldots p_{q-1} \tag{2.1}$$

by renumbering the methods. Next, a step-down approach rejects every null hypothesis up until H_i (that is, 'the performances of method M_i and the control method

are equivalent'), where i is the first index in sequence (2.1) for which $p_i > \frac{\alpha}{q-i}$. The adjusted p-values are defined as

$$p_{i,Holm} = \min(1, \max_{1 \leq j \leq i}[(q - j) \cdot p_j])$$

and represent the smallest global significance level at which the corresponding null hypothesis H_i would still be rejected. Based on this definition, we can derive that the control method significantly outperforms all methods with adjusted p-values smaller than α.

Chapter 3
Understanding OWA Based Fuzzy Rough Sets

As noted in Chap. 1, the traditional fuzzy rough set model is intrinsically sensitive to noise and outliers in the data. One generalization to deal with this issue in an intuitive way is the ordered weighted average (OWA) based fuzzy rough set model, that replaces the strict minimum and maximum operators by more elaborate OWA aggregations. This model is recalled in Sect. 3.1. The definitions of the OWA based fuzzy rough lower and upper approximations rely on a weighting scheme determining the weights used in the aggregation step. We recall several data-independent settings in Sect. 3.2 and introduce a novel data-dependent weighting scheme as well. We show that the suitability of each scheme depends on the characteristics of the data at hand. Moreover, we wish to gain a deeper understanding of how the OWA based fuzzy rough approximations are computed and the effect of the weighting scheme on this process. In Sects. 3.3 and 3.4, we therefore develop weighting scheme selection strategies for the OWA based fuzzy rough lower and upper approximation operators based on simple dataset characteristics. We prove the validity of our proposed guidelines in Sect. 3.5 on a corpus of independent datasets that were not used in the previous sections. Finally, Sect. 3.6 concludes the chapter.

3.1 Ordered Weighted Average Based Fuzzy Rough Sets

Due to its use of the strict minimum and maximum operators in the approximation definitions, the traditional fuzzy rough set model is sensitive to noise and outliers in the data. Several noise-tolerant alternatives have been proposed in the literature to address this issue. In light of its favourable properties, we select the OWA based fuzzy rough sets [108] from among them. In this section, we specify the definition of this model. Section 3.1.1 complements Sect. 1.2.3 and explains how the traditional fuzzy

S. Vluymans, *Dealing with Imbalanced and Weakly Labelled Data in Machine Learning using Fuzzy and Rough Set Methods*, Studies in Computational Intelligence 807, https://doi.org/10.1007/978-3-030-04663-7_3

rough approximations of decision classes in classification data can be computed. The ordered weighted average aggregation procedure is recalled in Sect. 3.1.2. The OWA based model itself is presented in Sect. 3.1.3.

3.1.1 Fuzzy Rough Approximations of Decision Classes

As explained in Sect. 1.2.3, the fuzzy rough lower and upper approximations of a concept C in the general implicator/t-norm model from [358] are defined as

$$\underline{C}(x) = \min_{y \in T}[\mathcal{I}(R(x, y), C(y))] \tag{3.1}$$

and

$$\overline{C}(x) = \max_{y \in T}[\mathcal{T}(R(x, y), C(y))]. \tag{3.2}$$

Recall that relation $R(\cdot, \cdot)$ expresses the similarity between two instances. In a classification context, C would correspond to a decision class and therefore be crisp. This means that the membership degrees $C(y)$ in expressions (3.1)–(3.2) only take on values in $\{0, 1\}$. In particular, $C(y) = 0$ when $y \notin C$ and $C(y) = 1$ when $y \in C$.

Taking this into account, expression (3.1) reduces to

$$\begin{aligned}
\underline{C}(x) &= \min_{y \in T}[\mathcal{I}(R(x, y), C(y))] \\
&= \min(\min_{y \in C}[\mathcal{I}(R(x, y), C(y))], \min_{y \notin C}[\mathcal{I}(R(x, y), C(y))]) \\
&= \min(\min_{y \in C}[\mathcal{I}(R(x, y), 1)], \min_{y \notin C}[\mathcal{I}(R(x, y), 0)]) \\
&= \min(1, \min_{y \notin C}[\mathcal{I}(R(x, y), 0)]) \\
&= \min_{y \notin C}[\mathcal{I}(R(x, y), 0)] \\
&= \min_{y \notin C}[\mathcal{N}_{\mathcal{I}}(R(x, y))].
\end{aligned} \tag{3.3}$$

We have used the property $(\forall a \in [0, 1])(\mathcal{I}(a, 1) = 1)$, which is due to $\mathcal{I}(1, 1) = 1$ and the fact that $\mathcal{I}$ is decreasing in its first argument. Operator $\mathcal{N}_{\mathcal{I}}$ is the induced negator of implicator $\mathcal{I}$ and is defined as $(\forall a \in [0, 1])(\mathcal{N}_{\mathcal{I}}(a) = \mathcal{I}(a, 0))$. The Kleene-Dienes, Łukasiewicz and Reichenbach implicators listed in Table 1.6 all have the same induced negator, namely the standard negator $\mathcal{N}_{\mathcal{I}}(a) = 1 - a$.

In a similar way, the fuzzy rough upper approximation (3.2) of a crisp decision class C reduces to

$$\begin{aligned}
\overline{C}(x) &= \max_{y \in T}[\mathcal{T}(R(x, y), C(y))] \\
&= \max(\max_{y \in C}[\mathcal{T}(R(x, y), C(y))], \max_{y \notin C}[\mathcal{T}(R(x, y), C(y))]) \\
&= \max(\max_{y \in C}[\mathcal{T}(R(x, y), 1)], \max_{y \notin C}[\mathcal{T}(R(x, y), 0)]) \\
&= \max(\max_{y \in C}[\mathcal{T}(R(x, y), 1)], 0) \\
&= \max_{y \in C}[\mathcal{T}(R(x, y), 1)] \\
&= \max_{y \in C}[R(x, y)]. \qquad (3.4)
\end{aligned}$$

This derivation relies on t-norm properties $(\forall a \in [0, 1])(\mathcal{T}(a, 0) = 0)$, due to $\mathcal{T}(1, 0) = 0$ and the fact that $\mathcal{T}$ is increasing in its first argument, and $(\forall a \in [0, 1])(\mathcal{T}(a, 1) = a)$.

In summary, in a classification dataset the fuzzy rough lower and upper approximations of a crisp decision class C are given by

$$\underline{C}(x) = \min_{y \notin C}[1 - R(x, y)] \quad \text{and} \quad \overline{C}(x) = \max_{y \in C}[R(x, y)],$$

where we have used one of the three popular implicators listed above. The membership degree of an instance x to the lower approximation of C is fully determined by the instance $y \notin C$ with the highest value for $R(x, y)$, that is, the instance $y \notin C$ closest to x. Similarly, the membership degree to the upper approximation of C is determined by the closest instance $y \in C$. The use of the minimum and maximum operators (and the consequent strong dependence of $\underline{C}(x)$ and $\overline{C}(x)$ on single instances y) renders these fuzzy rough approximation operators highly sensitive to noise and outliers in the data [118, 229]. To address this issue, several noise-tolerant fuzzy rough set models have been proposed in the literature. We focus on the ordered weighted average based model from [108], because it has been shown to be a preferred noise-tolerant fuzzy rough model among other alternatives in several studies [118].

3.1.2 Ordered Weighted Average Aggregation

Before going into detail on the OWA based fuzzy rough set model, we devote a separate section to its core component, the ordered weighted average aggregation [460, 462]. An OWA aggregation of a set of values V uses a weight vector W of length $|V|$. The aggregated value is obtained in two steps. First, the elements in V are sorted in decreasing order. Secondly, their weighted sum is computed by assigning the weights in W to the elements at the corresponding positions in the sorted sequence. In particular, we use the following definition.

Definition 3.1.1 (*OWA aggregation*) The OWA aggregation of a set of values V using weight vector $W = \langle w_1, w_2, \ldots, w_{|V|} \rangle$, with $(\forall i)(w_i \in [0, 1])$ and $\sum_{i=1}^{|V|} w_i = 1$, is given by

$$\mathrm{OWA}_W(V) = \sum_{i=1}^{|V|} (w_i v_{(i)}),$$

where $v_{(i)}$ is the ith largest element in V.

As an illustration of this aggregation procedure, consider the following small example. Let $V = \{0.1, 0.5, 0.3, 0.7, 0.7, 0.4\}$ and $W = \langle \frac{1}{21}, \frac{2}{21}, \frac{3}{21}, \frac{4}{21}, \frac{5}{21}, \frac{6}{21} \rangle$. The OWA aggregation of V is obtained as

$$\mathrm{OWA}_W(V) = \frac{1}{21} \cdot 0.7 + \frac{2}{21} \cdot 0.7 + \frac{3}{21} \cdot 0.5 + \frac{4}{21} \cdot 0.4 + \frac{5}{21} \cdot 0.3 + \frac{6}{21} \cdot 0.1 \approx 0.3619.$$

When we use weight vector $W = \langle \frac{1}{63}, \frac{2}{63}, \frac{4}{63}, \frac{8}{63}, \frac{16}{63}, \frac{32}{63} \rangle$ instead, we derive

$$\mathrm{OWA}_W(V) = \frac{1}{63} \cdot 0.7 + \frac{2}{63} \cdot 0.7 + \frac{4}{63} \cdot 0.5 + \frac{8}{63} \cdot 0.4 + \frac{16}{63} \cdot 0.3 + \frac{32}{63} \cdot 0.1 \approx 0.2429.$$

Note that the OWA aggregation of a singleton set always equals this single value itself, that is, $\mathrm{OWA}_W(\{v\}) = v$.

As the definition and example demonstrate, the result of an OWA aggregation largely depends on the choice of weight vector W. Characterizing properties of a weight vector are its *orness* and *andness* values. The former reflects how strongly the aggregation represents a regular maximum, while the latter indicates the relation with the minimum operator. For a weight vector W, $orness(W)$ and $andness(W)$ both belong to the unit interval and are defined such that $andness(W) = 1 - orness(W)$. In particular,

$$orness(W) = \frac{1}{p-1} \sum_{i=1}^{p} [(p - i) \cdot w_i],$$

where $p > 1$ is the length of the vector. If $orness(W) > \frac{1}{2}$, the aggregation with W corresponds to a *softened maximum*. Otherwise, if $andness(W) > \frac{1}{2}$, a *softened minimum* is modelled. For weight vector $W = \langle \frac{1}{p}, \frac{1}{p}, \ldots, \frac{1}{p} \rangle$, with which the aggregation corresponds to a regular average, $orness(W) = andness(W) = \frac{1}{2}$ holds. The traditional maximum and minimum operators can be modelled as OWA aggregations as well by means of the weight vectors $W_{max} = \langle 1, 0, \ldots, 0, 0 \rangle$ and $W_{min} = \langle 0, 0, \ldots, 0, 1 \rangle$ respectively. For these vectors, $orness(W_{max}) = andness(W_{min}) = 1$ and $orness(W_{min}) = andness(W_{max}) = 0$ holds.

3.1.3 OWA Based Fuzzy Rough Sets

The ordered weighted average based fuzzy rough set model was proposed in [108]. To deal with the noise sensitivity of the traditional fuzzy rough set model in expressions (3.1)–(3.4), the minimum and maximum operators are replaced by appropriate OWA_W aggregations, that is, $andness(W) > \frac{1}{2}$ (softened minimum) for the former and $orness(W) > \frac{1}{2}$ (softened maximum) for the latter. For a general concept C, the OWA based fuzzy rough approximations are obtained from (3.1)–(3.2) as

$$\underline{C}(x) = \text{OWA}_{W_L}(\{\mathcal{I}(R(x, y), C(y)) \mid y \in T\}) \tag{3.5}$$

and

$$\overline{C}(x) = \text{OWA}_{W_U}(\{\mathcal{T}(R(x, y), C(y)) \mid y \in T\}), \tag{3.6}$$

with $andness(W_L) > \frac{1}{2}$ and $orness(W_U) > \frac{1}{2}$. In the case of classification data where C corresponds to a crisp decision class, we can start from (3.3) and (3.4) and obtain

$$\underline{C}(x) = \text{OWA}_{W_L}(\{\mathcal{N}_{\mathcal{I}}(R(x, y)) \mid y \notin C\}) \tag{3.7}$$

and

$$\overline{C}(x) = \text{OWA}_{W_U}(\{R(x, y) \mid y \in C\}). \tag{3.8}$$

Possible choices for W_L and W_U are discussed in Sect. 3.2. We can already note that when $W_L = \langle 0, 0, \ldots, 0, 1\rangle$ and $W_U = \langle 1, 0, \ldots, 0, 0\rangle$, the above expressions reduce to the traditional fuzzy rough set model (3.1)–(3.4) due to Definition 3.1.1.

In [118], theoretical properties of a collection of noise-tolerant fuzzy rough set models were compared. The authors note that the OWA based fuzzy rough set model satisfies the set monotonicity and relation monotonicity properties, the most relevant characteristics from a practical application standpoint. Aside from the OWA based model, the proposals from [150, 465, 497] also satisfy these properties and can therefore, based on this criterion, be favoured in practice over other noise-tolerant models as well. In an experimental robustness analysis, the authors showed that the OWA based model is often the most robust one with respect to both attribute and class noise. The same conclusion was evident from our own related experimental stability evaluation in [424]. One challenge associated with the OWA based fuzzy rough set model lies with the definition of the weight vectors W_L and W_U. In the remainder of this chapter, we explain how sensible (and powerful) choices can be made based on the data at hand.

3.2 OWA Weighting Schemes

Having defined OWA based fuzzy rough sets in Sect. 3.1, we continue with a description and a preliminary comparison of several OWA weighting schemes used within this fuzzy rough set model. In Sect. 3.2.1, we recall the definition of four data-independent weighting schemes, one of which coincides with the traditional (non-OWA) fuzzy rough set model. To complement these settings, we introduce a new data-dependent weighting scheme in Sect. 3.2.2. Section 3.2.3 presents an initial comparison of the five schemes and motivates the more in-depth comparison conducted in Sects. 3.3 and 3.4, where we propose guidelines for the weighting scheme selection process.

3.2.1 Data-Independent Weighting Schemes

Section 3.1.3 already recalled one possible weighting scheme, namely the one for which the OWA based fuzzy rough set model reduces to the traditional one. We refer to this setting as *Strict* and it is defined as $W_L^{strict} = \langle 0, 0, \ldots, 0, 1\rangle$ and $W_U^{strict} = \langle 1, 0, \ldots, 0, 0\rangle$. These weight vectors are evidently data-independent. They contain only one non-zero position that does not depend on the actual values that are being aggregated. Aside from *Strict*, we consider three other data-dependent weighting schemes, where we assume that the size of the set to aggregate is p. For each of them, for a fixed length p, vectors W_L and W_U are reversals of each other. In the proofs below, we assume $p > 1$. In the other case, the aggregation is trivial.

Additive weights (*Add*) This weighting scheme models linearly decreasing or increasing weights and sets

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

and

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

These weight vectors are normalized versions of the $\langle 1, 2, \ldots, p-1, p\rangle$ and $\langle p, p-1, \ldots, 2, 1\rangle$ vectors respectively and correspond to the Borda count or law of Borda-Kendall in decision making [273]. Within vector W_L, each weight w_{i+1} is obtained by adding the constant value $\frac{2}{p(p+1)}$ to the preceding weight w_i. As a consequence, every position has a constant increase in its relevance to determine the aggregated value compared to the previous one. The opposite relation holds for W_U, where earlier positions are assigned more weight.

Theorem 3.2.1 *Weight vectors W_L^{add} and W_U^{add} correspond to a softened minimum and a softened maximum respectively.*

Proof We need to prove that $orness(W_L^{add}) < \frac{1}{2}$ and $orness(W_U^{add}) > \frac{1}{2}$. We use the definition of the *orness* measure as given above. Since W_L^{add} and W_U^{add} are reversals of each other, it suffices to prove the former inequality.

For the additive weight vector W_L^{add}, $w_i = \frac{2i}{p(p+1)}$ holds. We find

$$\begin{aligned}
orness(W_L^{add}) &= \frac{1}{p-1}\sum_{i=1}^{p}[(p-i)\cdot w_i] \\
&= \frac{1}{p-1}\sum_{i=1}^{p}[(p-i)\cdot \frac{2i}{p(p+1)}] \\
&= \frac{2}{(p-1)p(p+1)}\left(p\sum_{i=1}^{p} i - \sum_{i=1}^{p} i^2\right) \\
&= \frac{2}{(p-1)p(p+1)}\left(p\frac{p(p+1)}{2} - \frac{p(p+1)(2p+1)}{6}\right) \\
&= \frac{p}{p-1} - \frac{2p+1}{3(p-1)} \\
&= \frac{p-1}{3(p-1)} \\
&= \frac{1}{3} < \frac{1}{2}.
\end{aligned}$$

□

Exponential weights (*Exp*) In this scheme, the weights are drawn from an exponential function with base 2. In particular, weight w_{i+1} in W_L is determined by multiplying w_i by the constant factor 2. The constant ratio $\frac{w_{i+1}}{w_i} = 2$ in W_L implies that every position is twice as relevant for the aggregation as the previous one. In W_U, $\frac{w_i}{w_{i+1}} = 2$ holds. The weight vectors are given by

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

and

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

Theorem 3.2.2 *Weight vectors W_L^{exp} and W_U^{exp} correspond to a softened minimum and a softened maximum respectively.*

Proof We need to prove that $orness(W_L^{exp}) < \frac{1}{2}$ and $orness(W_U^{exp}) > \frac{1}{2}$. Since W_L^{exp} and W_U^{exp} are reversals of each other, it suffices to prove the former inequality.

For the exponential weight vector W_L^{exp}, $w_i = \frac{2^{i-1}}{2^p-1}$ holds. We find

$$\begin{aligned}
orness(W_L^{exp}) &= \frac{1}{p-1}\sum_{i=1}^{p}[(p-i)\cdot w_i] \\
&= \frac{1}{p-1}\sum_{i=1}^{p}\left[(p-i)\cdot\frac{2^{i-1}}{2^p-1}\right] \\
&= \frac{1}{(p-1)(2^p-1)}\sum_{i=1}^{p}[(p-i)\cdot 2^{i-1}] \\
&= \frac{1}{(p-1)(2^p-1)}\left((p-1)\sum_{i=1}^{p}2^{i-1} - \sum_{i=1}^{p}[(i-1)\cdot 2^{i-1}]\right) \\
&= \frac{1}{(p-1)(2^p-1)}\left((p-1)\sum_{j=0}^{p-1}2^{j} - \sum_{j=0}^{p-1}j2^{j}\right) \\
&= \frac{1}{(p-1)(2^p-1)}\left((p-1)\frac{1-2^p}{1-2} - \frac{2-p2^p+(p-1)2^{p+1}}{(2-1)^2}\right) \\
&= \frac{1}{(p-1)(2^p-1)}\left((p-1)(2^p-1)-2+p2^p-p2^{p+1}+2^{p+1}\right) \\
&= \frac{p2^p-p-2^p+1-2+p2^p-p2^{p+1}+2^{p+1}}{(p-1)(2^p-1)} \\
&= \frac{-p-2^p-1+2^{p+1}}{(p-1)(2^p-1)} \\
&= \frac{2^p-p-1}{(p-1)(2^p-1)},
\end{aligned} \tag{3.9}$$

where we have used formulas $\sum_{k=0}^{n} kx^k = \frac{x-(n+1)x^{n+1}+nx^{n+2}}{(x-1)^2}$ and $\sum_{k=0}^{n-1} r^k = \frac{1-r^n}{1-r}$. Based on expression (3.9), we easily derive

$$orness(W_L^{exp}) < \frac{1}{2} \Leftrightarrow (p-3)2^p + p + 3 > 0.$$

This condition is satisfied for any value $p \geq 2$. □

Inverse additive weights (*Invadd*) The third scheme is also based on the ratio between consecutive weights in the weight vectors. Instead of keeping this value constant as done by *Exp*, a variable weight ratio is used. For W_L, the ratio increase $\frac{w_{i+1}}{w_i}$ is put to $\frac{p-i+1}{p-i}$. This value is larger than one and increases with i, such that

relatively more weight increase is obtained at the tail of the vector. In particular, W_L is defined as

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

with $D_p = \sum_{i=1}^{p} \frac{1}{i}$, the pth harmonic number. It should be clear that this vector is the normalized version of $\langle \frac{1}{p}, \frac{1}{p-1}, \ldots, \frac{1}{2}, 1 \rangle$. By reversing this weight vector, we obtain

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

Here, the ratio $\frac{w_{i+1}}{w_i} = \frac{i}{i+1}$ increases with i as well. Since $\frac{i}{i+1} < 1$, this implies that the weight decrease slows down towards the tail of the vector.

Theorem 3.2.3 *Weight vectors W_L^{invadd} and W_U^{invadd} correspond to a softened minimum and a softened maximum respectively.*

Proof We need to prove that $orness(W_L^{invadd}) < \frac{1}{2}$ and $orness(W_U^{invadd}) > \frac{1}{2}$. Since W_L^{invadd} and W_U^{invadd} are reversals of each other, it suffices to prove the former inequality.

For the inverse additive weight vector W_L^{invadd}, $w_i = \frac{1}{(p-i+1)D_p}$ holds. We find

$$\begin{aligned}
orness(W_L^{invadd}) &= \frac{1}{p-1} \sum_{i=1}^{p} [(p-i) \cdot w_i] \\
&= \frac{1}{p-1} \sum_{i=1}^{p} \left[(p-i) \cdot \frac{1}{(p-i+1)D_p} \right] \\
&= \frac{1}{(p-1)D_p} \sum_{i=1}^{p} \left[\frac{p-i}{p-i+1} \right] \\
&= \frac{1}{(p-1)D_p} \left(\sum_{i=1}^{p} 1 - \sum_{i=1}^{p} \left[\frac{1}{p-i+1} \right] \right) \\
&= \frac{1}{(p-1)D_p} \left(p - \sum_{j=1}^{p} \frac{1}{j} \right) \\
&= \frac{p - D_p}{(p-1)D_p}. \qquad (3.10)
\end{aligned}$$

Based on this expression, we derive the condition

$$orness(W_L^{invadd}) < \frac{1}{2} \Leftrightarrow p(D_p - 2) + D_p > 0.$$

It is easy to verify that the D_p values increase with p and that $D_4 = \frac{25}{12} > 2$. This implies that the above condition is satisfied for all values $p \geq 4$. When $p = 2$ or $p = 3$, expression (3.10) yields an *orness* value of $\frac{1}{3}$ and $\frac{7}{22}$ respectively. Both values are smaller than $\frac{1}{2}$ as well. □

The following relationship holds between the three schemes listed above, implying that the exponential weighting scheme is most closely related to the strict minimum and maximum, followed by the inverse additive and additive weights.

Theorem 3.2.4 *The orness and andness values associated with the additive, exponential and inverse additive weighting schemes can be sorted as*

$$\begin{array}{rcl} orness(W_L^{exp}) & \leq & orness(W_L^{invadd}) \leq orness(W_L^{add}), \\ orness(W_U^{exp}) & \geq & orness(W_U^{invadd}) \geq orness(W_U^{add}), \\ andness(W_L^{exp}) & \geq & andness(W_L^{invadd}) \geq andness(W_L^{add}), \\ andness(W_U^{exp}) & \leq & andness(W_U^{invadd}) \leq andness(W_U^{add}), \end{array}$$

when the aggregation length p is larger than one.

Proof Due to the symmetry in the definitions of the *andness*, *orness* and the three weighting schemes, it is sufficient to prove that the first inequality holds.

We first prove the component on the left-hand side, namely $orness(W_L^{exp}) \leq orness(W_L^{invadd})$. From the proofs of Theorems 3.2.2 and 3.2.3, we obtain

$$orness(W_L^{exp}) = \frac{2^p - p - 1}{(p-1)(2^p-1)} \quad \text{and} \quad orness(W_L^{invadd}) = \frac{p - D_p}{(p-1)D_p}.$$

Since $p > 1$, we can remove factor $p - 1$ from both denominators and find

$$\begin{array}{rcl} orness(W_L^{exp}) \leq & & orness(W_L^{invadd}) \\ \Leftrightarrow \quad \frac{2^p-p-1}{(p-1)(2^p-1)} & \leq & \frac{p-D_p}{(p-1)D_p} \\ \Leftrightarrow \quad \frac{2^p-p-1}{2^p-1} & \leq & \frac{p-D_p}{D_p} \\ \Leftrightarrow \quad 1 - \frac{p}{2^p-1} & \leq & \frac{p}{D_p} - 1 \\ \Leftrightarrow \quad 0 & \leq & \frac{p}{D_p} + \frac{p}{2^p-1} - 2 \\ \Leftrightarrow \quad 0 & \leq & \frac{p2^p-p+pD_p-2D_p2^p+2D_p}{D_p(2^p-1)} \\ \Leftrightarrow \quad 0 & \leq & p2^p - p + pD_p - 2D_p2^p + 2D_p, \end{array}$$

where the last step is valid since the denominator $D_p(2^p - 1)$ is positive for all values $p > 1$. We can further rewrite this expression as

$$\begin{array}{rl} & 0 \leq p2^p - p + pD_p - 2D_p2^p + 2D_p, \\ \Leftrightarrow & 0 \leq p(2^p - 1) + pD_p - 2D_p(2^p - 1), \\ \Leftrightarrow & 0 \leq \quad (2^p - 1)(p - 2D_p) + pD_p. \end{array}$$

The above condition is satisfied for $p \geq 5$, as all factors and terms in the expression are positive in this case. This property is clear for $2^p - 1$ and pD_p and we can prove

$(\forall p \geq 5)(p - 2D_p \geq 0)$ by means of induction. To this end, we first prove the base case for $p = 5$, for which $D_5 = \frac{137}{60}$, such that

$$p - 2D_p = 5 - 2 \cdot \frac{137}{60} = \frac{300 - 274}{60} = \frac{26}{60} \geq 0.$$

We now assume that $p - 2D_p \geq 0$ holds and prove that $p + 1 - 2D_{p+1} \geq 0$ holds as well. We find

$$\begin{aligned} p + 1 - 2D_{p+1} &= p + 1 - 2(D_p + \frac{1}{p+1}) \\ &= p + 1 - 2D_p + \frac{2}{p+1} \\ &= p - 2D_p + 1 - \frac{2}{p+1} \geq 0, \end{aligned}$$

since (i) $p - 2D_p \geq 0$ according to the induction hypothesis and (ii) $1 \geq \frac{2}{p+1}$ as $p \geq 5$. From the above, we can conclude $(\forall p \geq 5)(p - 2D_p \geq 0)$ and, consequently, $(\forall p \geq 5)((2^p - 1)(p - 2D_p) + pD_p \geq 0)$. It remains to be shown that $(2^p - 1)(p - 2D_p) + pD_p \geq 0$ is satisfied for $p \in \{2, 3, 4\}$. Based on the values $D_2 = \frac{3}{2}$, $D_3 = \frac{11}{6}$ and $D_4 = \frac{25}{12}$, it can easily be verified that this statement holds.

To complete the proof, we show that $orness(W_L^{invadd}) \leq orness(W_L^{add})$. It has been shown that $orness(W_L^{add}) = \frac{1}{3}$ (see proof of Theorem 3.2.1) and we can derive

$$\begin{array}{lccl} & orness(W_L^{invadd}) & \leq & orness(W_L^{add}) \\ \Leftrightarrow & \frac{p - D_p}{(p-1)D_p} & \leq & \frac{1}{3} \\ \Leftrightarrow & 3p - 3D_p & \leq & pD_p - D_p \\ \Leftrightarrow & 0 & \leq & pD_p + 2D_p - 3p. \end{array}$$

We can prove the latter inequality by means of induction on p. We first consider the base cases $p = 2$ and $p = 3$, for which $D_2 = \frac{3}{2}$ and $D_3 = \frac{11}{6}$. We find

$$2 \cdot \frac{3}{2} + 2 \cdot \frac{3}{2} - 3 \cdot 2 = 6 - 6 = 0$$

for $p = 2$ and

$$3 \cdot \frac{11}{6} + 2 \cdot \frac{11}{6} - 3 \cdot 3 = \frac{55}{6} - 9 = \frac{1}{6} \geq 0.$$

for $p = 3$. We now assume that $pD_p + 2D_p - 3p \geq 0$ holds and prove $(p + 1)D_{p+1} + 2D_{p+1} - 3(p + 1) \geq 0$. We easily derive

$$
\begin{aligned}
(p+1)D_{p+1} + 2D_{p+1} - 3(p+1) &= (p+1)\left(D_p + \frac{1}{p+1}\right) + 2\left(D_p + \frac{1}{p+1}\right) - 3p - 3 \\
&= pD_p + \frac{p}{p+1} + D_p + \frac{1}{p+1} + 2D_p + \frac{2}{p+1} - 3p - 3 \\
&= pD_p + \frac{p+1}{p+1} + D_p + 2D_p + \frac{2}{p+1} - 3p - 3 \\
&= pD_p + 1 + D_p + 2D_p + \frac{2}{p+1} - 3p - 3 \\
&= pD_p + 2D_p - 3p + D_p - 2 + \frac{2}{p+1} \geq 0.
\end{aligned}
$$

The inequality holds because of (i) the induction hypothesis $pD_p + 2D_p - 3p \geq 0$ and (ii) $D_p - 2 + \frac{2}{p+1} \geq 0$ when $p \geq 2$. □

3.2.2 A Data-Dependent Weighting Scheme

When computing an OWA aggregation of a set V, the four schemes described in Sect. 3.2.1 do not take the values in V into account and only depend on its size p. Having selected a scheme among *Strict*, *Add*, *Exp* or *Invadd*, any set of size p will be aggregated in the same way, that is, by using the same weight vector. However, it can be useful to capture characteristics of V into the weight vectors. To study this setting, we introduce a new weighting scheme called *Mult*.

First, we note that the literature presents several methods to learn OWA weights from data [458], although not in the context of OWA based fuzzy rough sets, which is our focus here. In [338], an OWA weight generation procedure is proposed that maximizes the dispersion of the weights for a given $orness$ value, which needs to be specified by the user. An analytic solution to this optimization problem is offered in [168]. Alternatively, the weights can be calculated when a number of samples, in the form of a set of values of size p and their associated aggregation outcome, are provided. Examples of this approach can be found in [31, 32, 152, 402, 461]. Although interesting, these methods are of no use in the present context, because we cannot train the weights with a given $orness$ value or known aggregation outcomes, simply because it is not clear how these parameters should be set. Another dependent OWA weight vector was proposed in [459], which models a weighted mean of the aggregation values. The weight for each value is related to its distance to the overall mean. In [52], a cluster-based OWA aggregation was proposed. To aggregate a set of values, the reliability of each value to the entire set is evaluated based on a clustering of all values. A shortcoming of this procedure is its high computational cost as a result of the clustering step. This was pointed out by [53], which proposed a more efficient procedure to determine the reliability of a value. Like our *Mult* scheme discussed below, these methods compute their weights based on the values to be aggregated. Nevertheless, we cannot use them within the OWA based fuzzy rough approximations, because they do not act as a softened minimum

or maximum, but model averages instead. We now proceed with the definition of our data-dependent OWA weighting scheme *Mult* for use within OWA based fuzzy rough approximations.

We first discuss the definition of W_L^{mult}. Similar to *Invadd*, consecutive weights differ by a factor that depends on i. However, instead of modelling a vector with a relatively flat beginning and steep increase towards the end, we wish W_L^{mult} to follow the distribution of the sorted sequence of values to aggregate. Let $V = \langle v_1, \ldots, v_p \rangle$ be this set sorted in decreasing order. The sorting step ensures that v_p is the smallest value. For all other values v_1 to v_{p-1}, the similarity with the minimum can be computed as $s_L(v_i) = 1 - |v_i - v_p|$. These values belong to the unit interval. We define the function $m_L(\cdot)$, for $i \in \{1, 2, \ldots, p-1\}$, as

$$m_L(v_i) = \begin{cases} 1 & \text{if } v_i = v_{i+1}, \\ s_L(v_i) & \text{if } v_i \neq v_{i+1}. \end{cases}$$

The value $m_L(v_i)$ represents the factor by which w_{i+1} is multiplied to obtain w_i. As a consequence, the ratio $\frac{w_{i+1}}{w_i}$ is equal to $\frac{1}{m_L(v_i)}$, which is at least one for all $i \in \{1, 2, \ldots, p-1\}$. Weight vector W_L^{mult} is constructed from right to left, that is, starting from w_p. In a first step, w_p is set to one and every value w_i is obtained by multiplying w_{i+1} by the factor $m_L(v_i)$. This yields an intermediate vector

$$W_L^{mult*} = \left\langle \prod_{i=1}^{p-1} m_L(v_i), \prod_{i=2}^{p-1} m_L(v_i), \ldots, m_L(v_{p-2}) \cdot m_L(v_{p-1}), m_L(v_{p-1}), 1 \right\rangle.$$

Consider the sorted sets $V_1 = \langle 0.9, 0.8, 0.5, 0.2, 0.1 \rangle$ and $V_2 = \langle 0.8, 0.8, 0.4, 0.4, 0.2, 0.2 \rangle$ to aggregate for example. Following the above definition, it is easy to derive that the intermediate weight vectors W_L^{mult*} for these aggregations are $\langle 0.0324, 0.0162, 0.54, 0.9, 1 \rangle$ and $\langle 0.32, 0.32, 0.8, 0.8, 1, 1 \rangle$ respectively. Clearly, the distributions within a value set and the corresponding weight vector are similar. For example, a distinct staircase structure is present in both the value set and weight vector of V_2, because our use of the factors $m_L(v_i)$ ensures that when the values v_i and v_{i+1} are the same, they are also assigned the same weight. If $v_i \neq v_{i+1}$, w_{i+1} is multiplied by the factor $s_L(v_i)$. Due to its definition as the similarity of v_i with v_p, this factor decreases when i decreases, because values earlier in the ordered sequence V are less similar to the minimum value v_p. The decreasing factor implies that we can expect the weights to drop more rapidly towards the beginning of the vector. As such, the first values in V are far less relevant to the aggregation than the later ones, which coincides with our intuition of a softened minimum. The conditions in Definitions 3.1.1 require the weights to sum to one. The vector W_L^{mult} is therefore obtained as a normalized version of W_L^{mult*}, by dividing all weights by their total sum.

A mirrored procedure is followed to obtain W_U^{mult}. Value v_1 is the maximum value in the sorted sequence V. The similarity of other values with this maximum is defined as $s_U(v_i) = 1 - |v_1 - v_i|$, for $i \in \{2, 3, \ldots, p\}$ and we define function $m_U(\cdot)$ as

$$m_U(v_i) = \begin{cases} 1 & \text{if } v_i = v_{i-1}, \\ s_U(v_i) & \text{if } v_i \neq v_{i-1}. \end{cases}$$

These values are used as factors to obtain w_i from w_{i-1}. The intermediate weight vector W_U^{mult*} is constructed from left to right by setting weight w_1 to one and obtaining each w_i (with $i \in \{2, 3, \ldots, p\}$) by multiplying w_{i-1} with $m_U(v_i)$. We find

$$W_U^{mult*} = \left\langle 1, m_U(v_2), m_U(v_3) \cdot m_U(v_2), \ldots, \prod_{i=2}^{p-1} m_U(v_i), \prod_{i=2}^{p} m_U(v_i) \right\rangle$$

and obtain W_U^{mult} as a normalized version of this vector. Note that, contrary to the schemes discussed in Sect. 3.2.1, W_L^{mult} and W_U^{mult} are no reversals of each other.

We can prove that the above definitions of W_L^{mult} and W_U^{mult} yield softened minimum and softened maximum aggregations respectively.

Theorem 3.2.5 *Weight vector W_L^{mult} corresponds to a softened minimum, that is,*

$$andness(W_L^{mult}) > \frac{1}{2}.$$

When all values in V are the same, $andness(W_L^{mult}) = \frac{1}{2}$ holds.

Proof Weight vector W_L^{mult} is obtained by dividing all positions in W_L^{mult*} by the total sum D of the weights, such that weight w_i is given by

$$w_i = \frac{1}{D} w_i^* = \frac{1}{D} \prod_{j=i}^{p-1} m_L(v_j),$$

for $i \in \{1, \ldots, p-1\}$ and with w_i^* the ith value in W_L^{mult*}. Weight w_p equals $\frac{1}{D}$, since $w_p^* = 1$.

Based on the *andness* and *orness* definitions and $D = \sum_{i=1}^{p} w_i^*$, we can derive

$$\begin{aligned} & andness(W_L^{mult}) \geq \frac{1}{2} \\ \Leftrightarrow \quad & orness(W_L^{mult}) \leq \frac{1}{2} \\ \Leftrightarrow \quad & \frac{1}{p-1} \sum_{i=1}^{p} [(p-i) \cdot w_i] \leq \frac{1}{2} \end{aligned}$$

$$\Leftrightarrow \frac{1}{p-1}\sum_{i=1}^{p}\left[(p-i)\cdot\frac{1}{D}w_i^*\right] \leq \frac{1}{2}$$

$$\Leftrightarrow \frac{1}{(p-1)D}\sum_{i=1}^{p}[(p-i)\cdot w_i^*] \leq \frac{1}{2}$$

$$\Leftrightarrow \quad 2\cdot\sum_{i=1}^{p}[(p-i)\cdot w_i^*] \leq (p-1)D$$

$$\Leftrightarrow \quad 2\cdot\sum_{i=1}^{p}[(p-i)\cdot w_i^*] \leq (p-1)\sum_{i=1}^{p} w_i^*$$

$$\Leftrightarrow \quad \sum_{i=1}^{p}[(2p-2i)\cdot w_i^*] \leq \sum_{i=1}^{p}[(p-1)\cdot w_i^*]$$

$$\Leftrightarrow \quad \sum_{i=1}^{p}[(2i-1-p)\cdot w_i^*] \geq 0.$$

If p is even, we find, by grouping weights with opposite coefficients in the summation,

$$\sum_{i=1}^{p}[(2i-1-p)\cdot w_i^*] \geq 0$$

$$\Leftrightarrow \sum_{i=1}^{\frac{p}{2}}[(2i-1-p)\cdot w_i^*] + \sum_{i=\frac{p}{2}+1}^{p}[(2i-1-p)\cdot w_i^*] \geq 0$$

$$\Leftrightarrow \quad \sum_{i=1}^{\frac{p}{2}}[(2i-1-p)\cdot (w_i^* - w_{p-i+1}^*)] \geq 0$$

$$\Leftrightarrow \quad \sum_{i=1}^{\frac{p}{2}}[(p+1-2i)\cdot (w_{p-i+1}^* - w_i^*)] \geq 0. \tag{3.11}$$

The final expression holds because, when $i \in \{1, \ldots, \frac{p}{2}\}$, (i) all coefficients $p+1-2i$ are strictly positive and (ii) $w_{p-i+1}^* \geq w_i^*$ due to the definition of the weights and function $m_L(\cdot)$. Note that the equality in (3.11) only holds when all weights w_i^* are equal, which occurs when $(\forall i \in \{1, \ldots, p-1\})(m_L(v_i) = 1)$, that is, when all values in V are the same.

Similarly, when p is odd, we derive

$$\sum_{i=1}^{p}[(2i-1-p)\cdot w_i^*] \geq 0$$

$$\Leftrightarrow \sum_{i=1}^{\frac{p-1}{2}}[(2i-1-p)\cdot w_i^*] + \left[\left(2\frac{p+1}{2}-1-p\right)\cdot w_{\frac{p+1}{2}}^*\right] + \sum_{i=\frac{p+3}{2}}^{p}[(2i-1-p)\cdot w_i^*] \geq 0$$

$$\Leftrightarrow \sum_{i=1}^{\frac{p-1}{2}}[(2i-1-p)\cdot w_i^*] + \sum_{i=\frac{p+3}{2}}^{p}[(2i-1-p)\cdot w_i^*] \geq 0$$

$$\Leftrightarrow \sum_{i=1}^{\frac{p-1}{2}}[(2i-1-p)\cdot (w_i^* - w_{p-i+1}^*)] \geq 0$$

$$\Leftrightarrow \sum_{i=1}^{\frac{p-1}{2}}[(p+1-2i)\cdot (w_{p-i+1}^* - w_i^*)] \geq 0.$$

This expression holds for the same reasons as given above. It reduces to an equality only when all weights in the vector are the same, which only occurs when all values in V are equal. This completes the proof. □

Theorem 3.2.6 *Weight vector W_U^{mult} corresponds to a softened maximum, that is,*

$$orness(W_U^{mult}) > \frac{1}{2}.$$

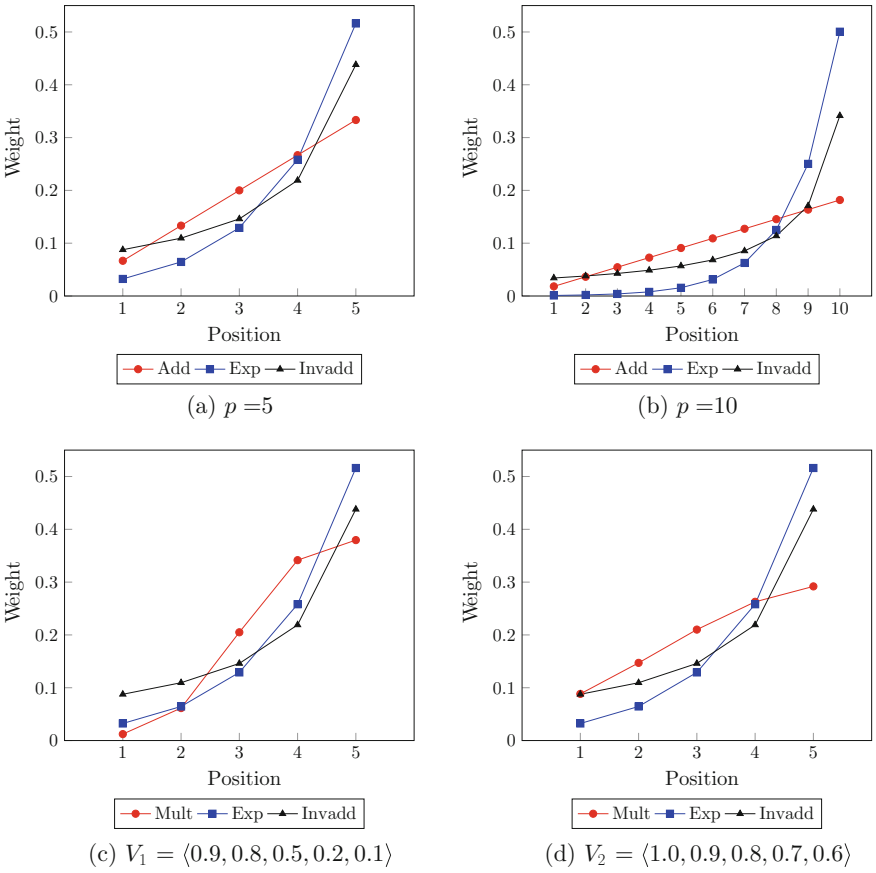
When all values in V are the same, $orness(W_U^{mult}) = \frac{1}{2}$ holds.

Proof Analogous to the proof of Theorem 3.2.5.

3.2.3 Preliminary Comparison

In the remainder of this chapter, the five OWA weighting schemes listed above (*Strict*, *Add*, *Exp*, *Invadd* and *Mult*) are compared to each other within OWA based fuzzy rough sets. The aim of this study is to provide guidelines for the weight selection process for the OWA based fuzzy rough lower and upper approximations in order to gain more insight into this model and make it more accessible to other users. This section provides an initial comparison between the five settings.

Figure 3.1 presents a graphical comparison of the weight vectors W_L generated by the different schemes. On the horizontal axis, we plot the position of a value in the ordered sequence. The possible indices are the natural numbers between 1 and p, where the latter is the size of the set to be aggregated. The vertical axis represents the value of the weight at a given position as a real number between zero and one. The plots do not contain a representation of *Strict*, because this degenerate scheme coincides with a strict minimum and assigns weight one to the final position and a zero weight to all others. The weight distribution across vector W_L^{strict} is therefore not interesting to depict. Theorems 3.2.1–3.2.5 showed that the W_L^{add}, W_L^{exp}, W_L^{invadd}



(a) $p = 5$ (b) $p = 10$

(c) $V_1 = \langle 0.9, 0.8, 0.5, 0.2, 0.1 \rangle$ (d) $V_2 = \langle 1.0, 0.9, 0.8, 0.7, 0.6 \rangle$

Fig. 3.1 Illustration of the different weighting schemes

and W_L^{mult} vectors all correspond to softened minima. This characteristic is evident from the figures as well, since more weight is assigned to the higher positions, such that more emphasis is given to the smaller values in the ordered sequence (Definition 3.1.1).

Figure 3.1a and b compare the three data-independent settings *Add*, *Exp* and *Invadd* for sets of sizes five and ten. As is clear from their description in Sect. 3.2.1, the additive weights in W_L^{add} take on the form of a straight line with slope $\frac{2}{p(p+1)}$. The exponential weights W_L^{exp} are drawn from an exponential function with base two. The relative weight increase for *Exp* is the same in each step, that is, w_{i+1} is obtained by multiplying w_i with a constant factor two. For the inverse additive weights, the $\frac{w_{i+1}}{w_i} = \frac{p-i+1}{p-i}$ ratio becomes larger when the position i increases. It is clear from Fig. 3.1a and b that this results in higher weights for the lower positions for *Invadd* compared to *Exp*. The exponential weights cancel out the contribution of the values at the lowest positions (in particular when p is high), while *Invadd* always

assigns them a non-negligible weight. This is an important difference between these two settings. The figures indicate that *Add* divides the weights more evenly across the positions than any other setting. Although fair, this may not always be beneficial. Especially when p is large, the weight associated with the true minimum may be relatively too low. For large values of p, *Add* becomes closely related to a regular average. This is already evident from Fig. 3.1b for $p = 10$. The flattening behaviour of W_L^{add} becomes considerably more pronounced as p increases further.

Figure 3.1c and d compare *Mult* to *Exp* and *Invadd*. These alternatives are related to each other in the sense that a weight can be obtained from the previous or next one by multiplication. The factor in this multiplication is fixed for *Exp*, while it varies for *Invadd* and *Mult*. We consider two different (sorted) value sets: $V_1 = \langle 0.9, 0.8, 0.5, 0.2, 0.1\rangle$ in Fig. 3.1c and $V_2 = \langle 1.0, 0.9, 0.8, 0.7, 0.6\rangle$ in Fig. 3.1d. Since V_1 and V_2 have the same size, the exponential and inverse additive weight vectors W_L^{exp} and W_L^{invadd} do not differ for these two aggregations. The *Mult* setting on the other hand uses the values in V_1 and V_2 to determine the weights and the plots show that these are very different for V_1 and V_2. The weights in W_L^{mult} closely follow the distribution of their values and its *orness* values are 0.7648 (Fig. 3.1c) and 0.6307 (Fig. 3.1d).

Both the definitions presented in Sects. 3.2.1 and 3.2.2 and the examples in Fig. 3.1 illustrate the considerable differences in the characteristics of the five weighting schemes. This motivates us to set up an in-depth study on the effect of the choice of the weighting scheme on the OWA based fuzzy rough approximations. In order to make our observations and conclusions practically usable, we provide selection guidelines for the OWA weighting scheme used in the (3.7) and (3.8) approximations. As noted in Sect. 3.1.1, the induced negator of several popular implicators coincides with the standard negator, such that, for these implicator choices, the OWA based fuzzy rough approximations of a decision class C are given by

$$\underline{C}(x) = \mathrm{OWA}_{W_L}(\{1 - R(x, y) \mid y \notin C\}) \quad \text{and} \quad \overline{C}(x) = \mathrm{OWA}_{W_U}(\{R(x, y) \mid y \in C\}). \tag{3.12}$$

The former is based on the distance (complement of similarity) to elements not in C, while the latter is based on the similarity with elements in C. This strong dependence on similarity values makes any fuzzy rough approach inherently related to nearest neighbour based techniques (Sect. 2.2.1). This relation is particularly pronounced when the *Strict* or *Exp* schemes are used. For *Strict*, we find $\underline{C}(x) = \min_{y \notin C}[1 - R(x, y)]$ and $\overline{C}(x) = \max_{y \in C}[R(x, y)]$. This lower approximation locates the nearest neighbour of x that does not belong to C, while the upper approximation is based on the nearest neighbour in C. In *Exp*, the nearest neighbour connection is more subtle, but is further pronounced as the aggregation length p increases. When p increases, *Exp* effectively sets several of the lowest weight positions to zero due to its exponential nature. For example, when $p = 50$, only 14 weights in the exponential vectors W_L^{exp} and W_U^{exp} are non-zero. This implies that $\underline{C}(x)$ and $\overline{C}(x)$ are based on only 14 near elements. The nearest neighbour characteristic of *Strict* and *Exp* is one of the important aspects which helps to explain the results in the remainder of this chapter.

Finally, we note that the computational complexity to aggregate the values in a set V with an OWA procedure is $\mathcal{O}(|V|\log(|V|))$ due to the cost of the sorting step (Definition 3.1.1). Sorting the values is not required in *Strict*, such that the cost reduces to $\mathcal{O}(|V|)$ for this setting. From the standpoint of computational efficiency, *Strict* could therefore be favoured over the other alternatives. However, as we will show in Sects. 3.3 and 3.4 and as has been sufficiently demonstrated in the literature, traditional fuzzy rough sets have certain shortcomings that the OWA based model addresses.

3.3 Lower Approximation Weighting Scheme Selection

In this section, we focus on the OWA based fuzzy rough lower approximation operator as given in (3.12). We derive weighting scheme selection guidelines for this operator based on an experimental evaluation on 50 datasets. Our experimental set-up is described in Sect. 3.3.1. Next, Sect. 3.3.2 motivates our study by showing that none of the weighting scheme alternatives discussed in Sect. 3.2 dominates all others, although each performs notably well on a non-negligible subset of our 50 datasets. Our weighting scheme selection strategy is presented in Sect. 3.3.3 and discussed and explained in further detail in Sect. 3.3.4.

3.3.1 Experimental Set-Up

We compare the five selected weighting schemes described in Sects. 3.2.1 and 3.2.2 (*Strict*, *Add*, *Exp*, *Invadd*, *Mult*) within the OWA based fuzzy rough lower approximation operator in a classification setting. We follow the study of [229] and use the lower approximation operator as a classifier. In particular, to classify an instance x, its membership degree to the lower approximation of all decision classes is computed, assigning x to the class for which this membership degree is highest. Weight vector W_L in (3.12) is set to the five alternatives listed above. This procedure shows how well the different weighting schemes can separate natural groups (here, classes) of observations.

We evaluate the performance on the 50 datasets listed in Table 3.1 by means of ten-fold cross validation. The datasets and their partitions can be obtained from the KEEL dataset repository at www.KEEL.es. For each dataset, the table lists the number of instances, number of features and classes and specifies whether all features are nominal (categorical) or not. Along with the number of classes, we indicate the level of imbalance between them by means of the imbalance ratio (IR). Following (1.1), we compute this measure as the ratio of the sizes of the largest and smallest classes in the dataset. For two-class datasets, this coincides with the measure traditionally used in studies on class imbalance (e.g. [391], Chap. 4). The classification performance of the fuzzy rough lower approximation is evaluated with the balanced accuracy (see

Sect. 2.3.1.2). We use this imbalance-tolerant evaluation measure, since Table 3.1 indicates that several datasets included in this study are severely imbalanced.

The fuzzy relation $R(\cdot, \cdot)$ that measures the similarity between instances is defined as

$$R(x, y) = \frac{1}{|\mathcal{A}|} \sum_{a \in \mathcal{A}} R_a(x, y), \tag{3.13}$$

where x and y are two instances and $\mathcal{A}$ is the feature set. The feature-wise relation $R_a(\cdot, \cdot)$ measures the similarity between x and y based on feature a. When a is a numeric feature, this relation is defined as

$$R_a(x, y) = 1 - \frac{|a(x) - a(y)|}{range(a)}, \tag{3.14}$$

where $a(x)$ and $a(y)$ are the values of x and y for feature a and the denominator $range(a)$ corresponds to its range. When a is a nominal feature, we set

$$R_a(x, y) = \begin{cases} 1 & a(x) = a(y) \\ 0 & a(x) \neq a(y). \end{cases} \tag{3.15}$$

Relation (3.13) has shown a good behaviour within fuzzy rough set based methods in related studies (e.g. [118, 361, 413]) and we use it throughout this work.

3.3.2 Motivation

As a further illustration of how the weighting scheme influences the calculations and results of the OWA based fuzzy rough approximations, we compute the balanced accuracy of the OWA based fuzzy rough lower approximation classifier by means of ten-fold cross validation on the 50 datasets listed in Table 3.1. The results per dataset for each of the five weighting schemes can be found in Tables 3.2 and 3.3. These tables divide the datasets into several groups, which have not yet been defined at this point, but will be described and motivated in Sects. 3.3.3 and 3.3.4. For each dataset, the highest value obtained by any of the weighting schemes is printed in bold.

As could have been expected based on the discussion in Sect. 3.2.3, substantial differences are observed between the five weighting schemes. The mean balanced accuracy of the OWA based fuzzy rough lower approximation predictor across the 50 datasets is 0.6693 (*Strict*), 0.6282 (*Add*), 0.6891 (*Exp*), 0.6867 (*Invadd*) and 0.6871 (*Mult*). The strict model attains the highest value in eleven datasets, *Add* in eight, *Exp* and *Mult* in nine and *Invadd* in fourteen. The total number of wins amounts to 51, because *Invadd* and *Mult* tie for the best value on the *wine* dataset (Table 3.3). In deriving our weighting scheme selection strategy, we do not wish to overfit these results by only focusing on the best setting for each dataset. Instead, we interpret any

Table 3.1 Description of the 50 datasets used in our experiments. We list the number of features (nFeat), the number of instances (nInst), the number of classes (nCl) and the IR. Together with the number of features, we specify whether they are all nominal (Y) or not (N)

Name	nFeat	nInst	nCl	IR	Name	nFeat	nInst	nCl	IR
abalone	8(N)	4174	28	689.00	page-blocks	10(N)	5472	5	175.46
australian	14(N)	690	2	1.25	phoneme	5(N)	5404	2	2.41
automobile	25(N)	159	6	16.00	pima	8(N)	768	2	1.87
balance	2(N)	625	3	5.88	ring	20(N)	7400	2	1.02
banana	2(N)	5300	2	1.23	saheart	9(N)	462	2	1.89
bands	19(N)	365	2	1.70	satimage	36(N)	6435	6	2.45
breast	9(Y)	277	2	2.42	segment	19(N)	2310	7	1.00
bupa	6(N)	345	2	1.38	sonar	60(N)	208	2	1.14
car	6(Y)	1728	4	18.62	spambase	57(N)	4597	2	1.54
cleveland	13(N)	297	5	12.62	spectfheart	44(N)	267	2	38.56
contra	9(N)	1473	3	1.89	splice	60(Y)	3190	3	2.16
crx	15(N)	653	2	1.21	texture	40(N)	5500	11	1.00
derma	34(N)	358	6	5.55	thyroid	21(N)	7200	3	40.16
ecoli	7(N)	336	8	28.60	tic-tac-toe	9(Y)	958	2	1.89
flare	11(Y)	1066	6	7.70	titanic	3(N)	2201	2	2.10
german	20(N)	1000	2	2.34	twonorm	20(N)	7400	2	1.00
glass	9(N)	214	6	8.44	vehicle	18(N)	846	4	1.10
haberman	3(N)	306	2	2.78	vowel	13(N)	990	11	1.00
heart	13(N)	270	2	1.25	wdbc	30(N)	569	2	1.68
ionosphere	33(N)	351	2	1.79	wine	13(N)	178	3	1.48
mammo	5(N)	830	2	1.06	winequal-r	11(N)	1599	6	68.10
marketing	13(N)	6876	9	2.49	winequal-w	11(N)	4898	7	439.60
monk-2	6(N)	432	2	1.12	wisconsin	9(N)	683	2	1.86
mov_lib	90(N)	360	15	1.00	yeast	8(N)	1484	10	92.60
nursery	8(Y)	12690	5	2160.00	zoo	16(Y)	101	7	10.25

result that is within 0.05 of the best value as acceptable and any other as poor. For each dataset, the poor results are underlined in Tables 3.2 and 3.3. We can observe that *Strict* performs poorly on 18 datasets, *Add* on 15, *Exp* on 12 and *Invadd* and *Mult* on 10.

Solely based on their mean performance, we would conclude that (i) *Exp*, *Invadd* and *Mult* are competitive weight settings and outperform *Strict* and (ii) *Add* does not work well. However, we observe that *Exp*, *Invadd* and *Mult* perform poorly on at least one fifth of the datasets, meaning that it is not a good idea to select one of these alternatives as a default option. It is also not prudent to exclude *Add* from consideration, as it gives the best result on eight datasets. Clearly, the optimal weighting scheme differs between datasets and this increases the difficulty of choosing between them. In the following sections, we study this phenomenon and discuss why certain weighting schemes are preferred in specific situations. We present a clear weighting scheme selection strategy for the OWA based fuzzy rough lower approximation.

Table 3.2 Balanced accuracy results of the lower approximation classifier for the datasets in the first five groups

Dataset	*Strict*	*Add*	*Exp*	*Invadd*	*Mult*
Only nominal features					
breast	0.4946	**0.5908**	0.5537	0.5826	0.5783
car	0.2459	0.3732	0.2638	0.3474	**0.3892**
flare	0.1971	0.4179	0.3107	**0.4731**	0.4274
nursery	0.2267	**0.2276**	0.2004	0.2268	0.2240
splice	0.5491	**0.5972**	0.5416	0.5737	0.5770
tic-tac-toe	0.5049	**0.5902**	0.5211	0.5658	0.5659
zoo	0.9229	0.8526	0.8883	0.8883	**0.9621**
Mean	0.4487	0.5214	0.4685	0.5225	**0.5320**
Perfectly balanced, numerical, low complexity					
mov_lib	**0.8722**	0.5889	0.8678	0.8500	0.8589
segment	**0.9766**	0.8433	0.9745	0.9494	0.9649
texture	0.9869	0.7596	**0.9884**	0.9664	0.9775
vowel	**0.9939**	0.6000	0.9859	0.9788	0.9869
Mean	**0.9574**	0.6980	0.9541	0.9361	0.9470
Numerical, at least 30 features, at most 1000 instances					
derma	0.9554	0.8844	**0.9765**	0.9689	0.9722
ionosphere	**0.8777**	0.7321	0.8684	0.8059	0.8311
sonar	0.8515	0.7962	0.8592	**0.8928**	0.8569
spectfheart	0.6165	0.6387	0.6295	**0.7310**	0.7100
wdbc	0.9507	0.9313	**0.9655**	0.9412	0.9473
Mean	0.8503	0.7965	0.8598	**0.8679**	0.8635
More than five classes, IR $\leq$ 10					
glass	0.7106	0.4659	**0.7130**	0.6039	0.6236
marketing	0.2101	0.2307	0.2575	**0.2701**	0.2653
satimage	0.8946	0.6360	**0.9026**	0.8309	0.8707
Mean	0.6051	0.4442	**0.6244**	0.5683	0.5865
More than five classes, IR $>$ 10					
abalone	0.1116	0.0903	0.1150	0.1047	**0.1364**
automobile	**0.7471**	0.5243	0.6734	0.6550	0.6600
ecoli	0.7170	0.5598	0.7413	0.7381	**0.7476**
winequal-r	**0.3616**	0.2811	0.3590	0.3355	0.3561
winequal-w	**0.4498**	0.2567	0.4419	0.3863	0.4067
yeast	0.5181	0.3960	0.5359	0.5554	**0.5564**
Mean	**0.4842**	0.3514	0.4777	0.4625	0.4772

3.3.3 Proposed Weight Selection Strategy

The full balanced accuracy results for the OWA based fuzzy rough lower approximation classifier are presented in Tables 3.2 and 3.3. Each column corresponds to one of the five evaluated weight settings. The datasets are divided into eight groups based on simple and easy-to-compute data characteristics. We explain and validate this division in the remainder of this section. As a reminder, values that are not printed

Table 3.3 Balanced accuracy results of the lower approximation classifier for the datasets in the last three groups

Dataset	*Strict*	*Add*	*Exp*	*Invadd*	*Mult*
At most five classes, at most 4000 instances, IR $\leq$ 2					
australian	0.7250	**0.8730**	0.7946	0.8675	0.8513
bands	**0.7308**	0.6509	0.7164	0.7173	0.6439
bupa	0.6160	0.6422	0.6498	**0.6782**	0.6696
contra	0.4030	0.4690	0.4374	**0.4836**	0.4604
crx	0.7715	0.8695	0.8326	**0.8706**	0.8650
heart	0.7808	0.8225	0.8058	**0.8242**	0.8117
mammo	0.7456	0.8205	0.8082	**0.8214**	0.8114
monk-2	0.7409	0.9052	0.9059	**0.9171**	0.8144
pima	0.6516	0.7014	0.6754	**0.7040**	0.6778
saheart	0.5853	0.6620	0.6104	**0.6769**	0.6315
vehicle	0.6942	0.5670	**0.7082**	0.6615	0.7006
wine	0.9631	0.9631	0.9673	**0.9679**	**0.9679**
wisconsin	0.9617	0.9301	0.9654	0.9497	**0.9737**
Mean	0.7207	0.7597	0.7598	**0.7800**	0.7599
At most five classes, at most 4000 instances, IR $>$ 2					
balance	0.5417	0.7576	0.6378	0.6528	**0.7874**
cleveland	**0.3001**	0.2919	0.2855	0.2820	0.2710
german	0.5619	**0.6576**	0.5750	0.5740	0.5629
haberman	0.5360	**0.6363**	0.5475	0.5651	0.5267
titanic	0.5211	0.6997	0.6812	**0.7109**	0.7083
Mean	0.4922	**0.6086**	0.5454	0.5570	0.5712
At most five classes, more than 4000 instances					
banana	0.8728	0.7246	0.8929	0.8848	**0.8975**
page-blocks	0.7610	0.3198	**0.7915**	0.5506	0.5978
phoneme	0.8738	0.7730	**0.8743**	0.8165	0.8455
ring	**0.7181**	0.5000	0.6924	0.5139	0.5742
spambase	0.8987	0.8027	**0.9091**	0.8730	0.8447
thyroid	**0.6219**	0.5280	0.5928	0.5720	0.4340
twonorm	0.9462	**0.9757**	0.9619	0.9754	0.9723
Mean	0.8132	0.6605	**0.8164**	0.7409	0.7380

in boldface nor underlined can be considered as acceptable alternatives to the best setting for a dataset.

For each of the eight dataset groups, one weighting scheme emerges as the best performing one and is framed in Tables 3.2 and 3.3. The selected setting does not necessarily correspond to the one attaining the highest mean value for a group, but should yield acceptable results for each member in its group. We wish to develop a sufficiently general weighting scheme selection strategy and should not overfit these results by proposing weight guidelines that are too specific. In particular, we propose the following groups of datasets, along with their recommended OWA weighting schemes for the fuzzy rough lower approximation.

1. **Datasets with only nominal features**: our feature-wise similarity relation (3.15) for a nominal feature a is either zero or one, that is, $R_a(x, y) = 0$ when x and y have different values for a and $R_a(x, y) = 1$ when their feature values for a coincide. As a result of this lack of variety in instance similarity values, we can expect many elements to be found at the exact same distance of a given target and, consequently, many coinciding values in the sets to aggregate in the lower approximation calculation. This renders a nearest neighbour approach unsuitable, a property which is reflected in the poor results of *Strict* and *Exp*. Our proposed *Mult* weighting scheme is able to model the distinct staircase structure in the values to be aggregated (see the example in Sect. 3.2.2). It guarantees that equal values are assigned equal weights in the OWA aggregation. Its data-dependent nature renders *Mult* an appropriate choice for datasets with only nominal features, as evidenced by the results in Table 3.2. We recommend its use for this group of datasets. The results for *Add* and *Invadd* are close together and acceptable on average, but they both perform poorly on at least one dataset in this group. For the *Add* setting, which yields inferior results on the *flare* and *zoo* datasets, this is explained by the fact that these datasets have a high number of classes (see Sect. 3.3.4.1). When a small dataset with only nominal features contains only a few classes, *Add* could be used instead of *Mult*.
2. **Perfectly balanced numerical datasets with low data complexity**: this group contains four datasets, for which all classes have the same sizes and all features are numerical. They also have a low dataset complexity as measured by the multi-class Fisher discriminant score [218], a metric defined for datasets with only numeric features. Data complexity can be interpreted as the intrinsic difficulty associated with a dataset due to, for instance, geometrical properties such as high class overlap. The Fisher score evaluates the maximum relative separation of feature values between elements of different classes and a higher value corresponds to a lower complexity. The four datasets in this group have Fisher scores of 2.5413 (*mov_lib*), 15.6143 (*segment*), 10.2872 (*texture*) and 2.0389 (*vowel*). These are high values compared to the mean score of 1.8451 across the 50 datasets in Table 3.1. We recommend the use of the traditional fuzzy rough set model, corresponding to setting *Strict*, for the datasets in this group. On these easy datasets, the OWA based fuzzy rough set model does not have a clear advantage over the original proposal. By selecting *Strict*, the sorting step in the OWA aggregation can be avoided. We note that the additive weighting scheme performs very poorly on this group, but this is due to the high number of classes in each of these datasets (see Sect. 3.3.4.1). Finally, we should also remark that the complexity condition is crucial. For example, dataset *mammo* is almost perfectly balanced, but *Strict* performs poorly for it. The Fisher score of *mammo* is 0.4926.
3. **Numerical datasets with at least 30 features and at most 1000 instances**: five datasets are included in this group. The *mov_lib* dataset from group 2 could be included as well, because it contains 360 elements and 90 features. Note that the included datasets contain solely numeric features. Datasets with a high dimensionality tend to be sparse (empty space phenomenon) and distance- or similarity-based approaches in general lose some of their power as the pairwise

distance or similarity between elements becomes less informative. None of the training instances used in the lower approximation calculation (3.12) can truly be considered as 'close' to the target x. Since all training elements will show a small similarity with x, the values aggregated in the OWA step of the lower approximation calculation for x can be expected to be more similar to each other than they would be in lower dimensional datasets. The *Add* scheme clearly fails in this situation, since it is most closely related to an average (Sect. 3.2.3). For each class, the values in the aggregation will be highly similar. When we aggregate them with an average-like procedure, the final values for all classes will be more or less the same. As a result, the prediction by *Add* is close to a random guess. The mean results of *Strict*, *Exp*, *Invadd* and *Mult* are close together. We recommend the use of the latter, as it does not perform poorly on any of the datasets in this group. The three remaining options sometimes provide a bad result. Although our *Mult* setting never obtains the highest balanced accuracy, it is the safest choice. We have developed this data-dependent alternative in such a way that its weights follow the distribution of the values to be aggregated. We see the benefit of this idea on these complex datasets, where the weights capture important differences between elements and can discern better between classes.

4. **Datasets with more than five classes and IR** ≤ 10: we recommend the use of *Exp* for this group (see Sects. 3.3.4.1 and 3.3.4.3).
5. **Datasets with more than five classes and IR** > 10: the traditional fuzzy rough set model performs best and we therefore recommend the use of *Strict* (see Sects. 3.3.4.1 and 3.3.4.3).
6. **Datasets with at most five classes, at most 4000 instances and IR** ≤ 2: for these small and relatively balanced datasets, the inverse additive weighting scheme stands out as the best performing one. We observe that there is only one truly poor weighting scheme for this group, namely *Strict*. These datasets are relatively easy to handle, in the sense that they are not too large, do not have too many classes and are not too imbalanced. This proves to be a setting in which any OWA aggregation performs better than the strict model, as there are no prior factors that can severely hinder the OWA procedure. The four true OWA aggregations all provide relatively good results and their mean balanced accuracy values are close together. Nevertheless, *Add*, *Exp* and *Mult* fail on some of these datasets, while *Invadd* never does. Consequently, we advise the use of the latter for this group. We note that *Invadd* has also been shown to be the best performing weighting scheme in general (i.e., on average) in previous studies (e.g. [413]). Its strength is most evident for this group of datasets. In fact, from the fourteen datasets on which *Invadd* attained the highest balanced accuracy among the evaluated alternatives, nine are contained in this group. When there are no prior challenging factors (e.g. large size, many classes, class imbalance), *Invadd* seems to be a solid default weighting scheme.
7. **Datasets with at most five classes, at most 4000 instances and IR** > 2: the *Add* setting exhibits the best performance on this group of datasets. Its strength in the presence of class imbalance on datasets with a low number of classes is explained in Sect. 3.3.4.3.

8. **Datasets with at most five classes and more than 4000 instances**: when a dataset contains many instances, Table 3.3 indicates that *Exp* is a favoured setting. We explain this behaviour in Sect. 3.3.4.2.

The selected thresholds are based on the results in Tables 3.2 and 3.3. They are justified in Sect. 3.3.4 and validated in Sect. 3.5 on independent datasets not considered in the study so far. When appearing too artificial, the user can relax these guidelines, for instance by replacing 'more than five classes' by 'many classes', 'at most 4000 instances' by 'a small dataset' and so on. This does not detract from our contribution, since our goal is to capture and understand the general behaviour of the various weight settings. In practice, users would first determine whether their dataset belongs to the first group. If not, they evaluate the characteristics of the second group and the third one after that. When the dataset does not belong to any of the first three groups, the user should decide to which of the final five it belongs, which are mutually exclusive.

3.3.4 Detailed Discussion

We now consider the results discussed in the previous section in more detail. We answer a set of questions related to the observations made above and provide further insight in the performance of the different OWA weighting schemes. We consider three different challenges: a high number of classes (Sect. 3.3.4.1), a high number of instances (Sect. 3.3.4.2) and class imbalance (Sect. 3.3.4.3).

3.3.4.1 High Number of Classes

A high number of classes is interpreted here as 'more than five'. This concerns all datasets in groups 4 and 5 by definition. Apart from these, the four datasets in group 2, *flare* and *zoo* in group 1 and *derma* in group 3 all contain more than five classes as well. In all, 16 of the 50 datasets in Table 3.1 have more than five classes. The mean balanced accuracy of the weighting schemes over these datasets are presented in Fig. 3.2. The *Strict* setting obtains the best result on six datasets, *Mult* and *Exp*

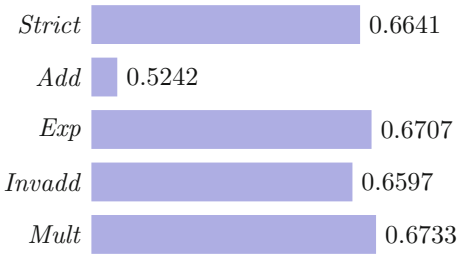


Fig. 3.2 Mean balanced accuracy of the weighting schemes across the 16 datasets in Table 3.1 containing more than five classes

each obtain the best result on four, while *Invadd* takes first place on the remaining two. Aside from its low mean value, *Add* also provides a notably poor result on 14 out of these 16 datasets. The other alternatives each perform poorly on two datasets. We address the following questions:

1. Why does *Add* not perform well when the number of classes is high?
2. In datasets with many classes, why are *Strict* and *Exp* the preferred settings?

Question 1 Based on its mean balanced accuracy and the number of datasets on which it performs poorly, *Add* is clearly inferior to the other weighting schemes (including *Strict*) when the number of classes is high. The reason is that there exists too little difference in the sets of values to aggregate in (3.12), that is, these sets have a high degree of overlap. Consider a dataset with six classes (C_1 to C_6). The membership degree of an element x to $\underline{C_i}$ is computed by aggregating values $1 - R(x, y)$ for instances y in any of the five other classes. This implies an overlap of four classes between the aggregation sets of $\underline{C_i}(x)$ and $\underline{C_j}(x)$. For example, $\underline{C_1}(x)$ and $\underline{C_2}(x)$ both use all values $1 - R(x, y)$ for $y \in C_3 \cup C_4 \cup C_5 \cup C_6$. The only difference between the value vectors used in $\underline{C_1}(x)$ and $\underline{C_2}(x)$ is that the former also uses class C_2 and the latter also class C_1.

The increased expected overlap between the value vectors in the lower approximation aggregations holds for the OWA model in general and is not specific for the *Add* scheme. Nevertheless, since *Add* assigns a large relative importance to all values (Sect. 3.2.3), the high overlap implies that the aggregated values for the different classes will be close together. Consequently, the ability to discern between classes decreases and prediction errors are made. Other weight settings are hindered less by the overlap issue, since their weight distribution places a clearer emphasis towards the minimum (Sect. 3.2.3).

Secondly, on top of the overlap problem, a high number of classes can also imply an increase in the size of the sets to aggregate. When the additive weight vector becomes longer, its behaviour approaches that of a regular average, which further accentuates the issues that the $\underline{C_i}(x)$ values are not sufficiently distinct.

Question 2 In our proposed weight selection strategy presented in the previous section, we recommend *Strict* or *Exp* for datasets with more than five classes (groups 4 and 5). Table 3.2 shows that these are indeed the preferred configurations for such datasets in this study. Although they have been computed over a larger set of datasets, the mean results listed at the beginning of this section confirm this observation.

Aside from the poor performance of *Add*, Table 3.2 also indicates that *Invadd* and *Mult* perform relatively less strongly on the datasets in groups 4 and 5. As explained in our answer to the previous question, when there are many classes in a dataset, there is a high degree of overlap between the value vectors in the lower approximation computations. Although not to such an extent as *Add*, the *Invadd* and *Mult* settings also assign non-negligible weights to all values to be aggregated. As a result, they are also at risk for aggregated class lower approximation values that are too close to each other to adequately distinguish between them. *Mult* outperforms

Invadd, because its weights are set up to decrease more rapidly going from right to left in the lower approximation weight vector.

Strict and *Exp* are nearest neighbour approaches and only consider a small portion of the values in their aggregation step, because they effectively put some weights to zero. As a result, they avoid the overlap problem, which explains why they are the preferred weight setting in this situation. We make a further distinction between groups 4 and 5 based on the class imbalance present in the dataset. The reason why *Strict* is preferred over *Exp* on datasets with a large IR is discussed in Sect. 3.3.4.3.

3.3.4.2 High Number of Instances

In Sect. 3.3.3, we have put the threshold on a high number of instances to 4000. Evidently, this is a small number in the big data era, but it is sufficiently large considering the dataset characteristics described in Table 3.1. There are 13 datasets in our study with a size larger than 4000. These include the seven datasets from group 8, *marketing* and *satimage* from group 4, *abalone* and *winequal-w* from group 5, *nursery* from group 1 and *texture* from group 2. The mean balanced accuracy of the weighting schemes on these datasets is presented in Fig. 3.3. Settings *Strict* and *Exp* are clearly preferable in this situation and the latter is the overall best choice.

The difference between *Strict* and *Exp* on the one hand and *Add*, *Invadd* and *Mult* on the other is that the nearest neighbour nature of the former two can cancel out the contribution of some instances (Sect. 3.2.3). For *Strict*, this is always the case, as this scheme assigns zero weights to all but one position. For *Exp*, zero weights occur when the length of the weight vector increases. Due to the rapid (exponential) descent in weights from right to left in vector W_L^{exp}, only a small portion of values are effectively assigned non-zero weights.

A larger dataset size implies a larger length of the weight vectors. The experimental results show that a nearest neighbour approach is more suitable for this situation than other OWA aggregations that take all values into account. *Add*, *Invadd* and *Mult* lose some of their characteristics in this situation, because their definitions insist on assigning some weight to all values. As the aggregation length increases, important values (i.e., those close to the minimum) are assigned progressively smaller weights to accommodate for this property, since OWA weights always sum to one

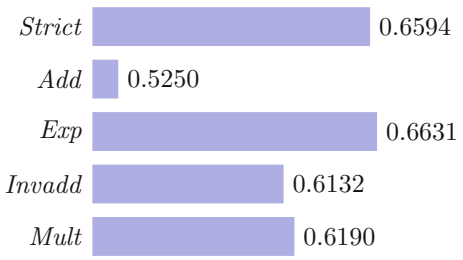


Fig. 3.3 Mean balanced accuracy of the weighting schemes across the 13 datasets in Table 3.1 containing more than 4000 instances

(Definition 3.1.1). This is most prominently noticeable in the *Add* scheme, where the weight vector almost flattens out to a regular average for large set aggregations.

The reason why *Exp* is preferred over *Strict* is the same as why kNN classification (with $k > 1$) is preferred over 1NN classification. By relying on multiple near elements, a better robustness against noise is obtained with more confident predictions as a result. Furthermore, weighted kNN is often favoured over uniform kNN, because the former assigns relatively more importance to nearer neighbours in its predictions [133].

3.3.4.3 Class Imbalance

We have used the IR of a dataset on two occasions in our proposed weighting scheme selection guidelines presented in Sect. 3.3.3, namely to make a distinction between groups 4 and 5 on the one hand and between groups 6 and 7 on the other. We now explain why the appropriate weighting scheme can depend on the class imbalance in a dataset. To do so, we answer two questions:

1. In datasets with a low number of classes and instances, why is *Add* the only good choice when the dataset is at least mildly imbalanced?
2. In datasets with many classes, why is *Strict* preferred in case of large imbalance and *Exp* in case of small to mild imbalance?

Question 1 This question pertains to datasets with at most five classes, at most 4000 instances and an IR of at least two. The latter implies that the largest class is at least twice as large as the smallest class. The five datasets in group 7, the *breast*, *car* and *splice* datasets from group 1 and the *spectfheart* dataset from group 3 meet these requirements. The mean balanced accuracy of the weighting schemes on these nine datasets are presented in Fig. 3.4. The additive scheme attains the best average result and the highest number of wins, namely on four out of nine datasets. Although *Add* appeared to be an inferior weight alternative in Sect. 3.3.2, it is clearly dominant on small, imbalanced datasets. We expect that the poor performance of *Strict* and *Exp* is due to their relation to the nearest neighbour classifier and the sensitivity of the latter to class imbalance.

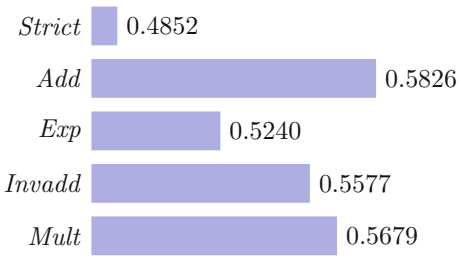


Fig. 3.4 Mean balanced accuracy of the weighting schemes across the nine datasets in Table 3.1 with at most five classes, at most 4000 instances and an IR of at least two

Considering the results in more detail, we noticed that the other OWA alternatives often fall into the trap of class imbalance, that is, they assign instances to a majority class too easily. This implies a high accuracy on the majority class, but severely lower accuracies on minority classes. Our use of the balanced accuracy as evaluation measure guarantees that a bad performance on small classes is not overshadowed by a strong result on large classes. *Add* usually has similar accuracy rates for all classes in these datasets, reflected in its superior balanced accuracy values.

As evident from (3.12), the membership degree to the OWA based fuzzy rough lower approximation of a class C is calculated by aggregating values based on elements that do not belong to C. Assume that a dataset contains two classes (C_1 and C_2) and that the former is the majority class. Due to the above condition on the IR, this means that C_1 is at least twice as large as C_2. To predict a class label for an instance x, our classifier computes two values: $\underline{C_1}(x) = \text{OWA}_{W_L}(\{1 - R(x, y) \mid y \in C_2\})$ and $\underline{C_2}(x) = \text{OWA}_{W_L}(\{1 - R(x, y) \mid y \in C_1\})$. Due to the difference in class sizes, the first aggregation is taken over far less values than the second. The largest influence of this fact is felt by the *Add* scheme. As discussed above, the longer the additive weight vector, the more closely it resembles a regular average. Since the aggregation lengths can be very different, the characteristics of *Add* on either class can severely vary as well. The longer aggregation ($\underline{C_2}(x)$) will be far closer to an average of its values than the shorter one ($\underline{C_1}(x)$). This difference in the treatment of the classes is far less pronounced for the other weighting schemes (and even non-existent in *Strict*). Here lies the key to why *Add* is preferred in the presence of class imbalance: its high sensitivity to the length of the vector makes it process minority and majority classes very differently. It does not allow for majority elements to dominate minority elements. The contributions of majority instances are almost averaged in the calculations, while those of minority instances are aggregated with a truer OWA procedure. A similar conclusion holds when there are more than two classes.

In summary, like many other classifiers, the lower approximation predictor is sensitive to class imbalance. The additive weighting scheme inherently treats majority and minority classes differently, which is why it is the preferred choice here. We note that the study of [361] provides a more detailed study on appropriate OWA weight vectors when dealing with two-class imbalanced datasets (see Chap. 4).

Question 2 This question relates to datasets with more than five classes. We recommend to use *Exp* when the IR is at most ten (group 4) and *Strict* otherwise (group 5). Section 3.3.4.1 explains why *Strict* and *Exp* are the favoured weight options for datasets with more than five classes. For datasets with a moderate IR, *Exp* is preferred over *Strict* for the same reason as given in Sect. 3.3.4.2, namely the higher prediction confidence and robustness when more than one near neighbour is used in the classification process. For a highly imbalanced dataset, the strict model is a better option. This is due to the class imbalance problem (Chap. 4), which has a larger influence on kNN ($k > 1$) than on 1NN. As a consequence, *Exp* loses some of its strength when the imbalance becomes large.

3.4 Upper Approximation Weighting Scheme Selection

We now turn our attention to the OWA based fuzzy rough upper approximation. We follow the same experimental set-up as in Sect. 3.3, although we now use the fuzzy rough upper approximation operator as predictor. Section 3.4.1 shows that a similar phenomenon occurs for the upper approximation as for the lower approximation, namely that no single weighting scheme can be put forward as a confident strong default choice. To remedy this situation, we present our proposed weighting scheme selection strategy for the upper approximation in Sect. 3.4.2.

3.4.1 Motivation

The full balanced accuracy results of the OWA based fuzzy rough upper approximation classifier are presented in Tables 3.4 and 3.7. We include the 50 datasets listed in Table 3.1. The datasets have been divided into several groups, for which the use of specific weighting schemes is advised in Sect. 3.4.2. We again evaluate five OWA weighting schemes: *Strict*, *Add*, *Exp*, *Invadd* and *Mult*. As before, for each dataset, the highest balanced accuracy attained by one of these alternatives is printed in boldface. Values that are more than 0.05 smaller than this optimum are underlined.

The mean balanced accuracies across the 50 datasets are 0.6706 (*Strict*), 0.6621 (*Add*), 0.6896 (*Exp*), 0.6917 (*Invadd*) and 0.6774 (*Mult*). The *Invadd* setting yields the highest balanced accuracy on average and attains the most wins as well. In particular, this alternative attains the best value on seventeen datasets, while the other schemes do so on eleven (*Strict* and *Add*), eight (*Exp*) and four (*Mult*) datasets respectively. The average results are close together. While all included weighting schemes win on several datasets, they perform poorly on many as well. *Strict* and *Add* both yield inferior results on as many as twenty datasets. This number goes down to fourteen for *Mult* and twelve for *Exp* and *Invadd*, but remains high nonetheless. As argued in Sect. 3.3.2, this behaviour warrants a deeper study.

3.4.2 Proposed Weighting Scheme Selection Strategy

Table 3.4 lists the results of the datasets categorized in the same first three groups as presented in Sect. 3.3.3. The advised weighting schemes are framed in the table. We propose the following:

1. **Datasets with only nominal features**: it is evident from the experimental results that both *Strict* and *Exp* are poor choices on these datasets, as they were for the lower approximation operator as well. As explained in Sect. 3.3.3, the lack of variety in the feature similarity values renders a nearest neighbour approach, to which both *Strict* and *Exp* are strongly related, unsuitable. The remaining

three alternatives (*Add*, *Invadd* and *Mult*) provide similar results. *Invadd* can be preferred, since it does not yield a poor result on any of these datasets.

2. **Perfectly balanced numerical datasets with low data complexity**: the same conclusion as for the lower approximation holds, namely that *Strict* is the most suitable alternative on these datasets. We refer the reader back to Sect. 3.3.3 for an explanation. By opting for *Strict*, the need for the sorting step in the true OWA aggregations is avoided. As we observed for the lower approximation, the *Add* alternative does not perform well on these datasets. They all contain many classes, but the associated overlap issue discussed in Sect. 3.3.4.2 can not be a direct cause of the performance degradation of *Add* here, as the upper approximation aggregation (3.12) occurs over elements within a class and not within the complement of a class. Quite possibly, the stronger averaging effect of *Add* takes the class prediction procedure too far away from the most appropriate 1-nearest neighbour classifier and results in a confusion between classes. Indeed, the mean results of the five alternatives on these datasets decrease according to the $orness$ values of the weighting schemes.
3. **Numerical datasets with at least 30 features and at most 1000 instances**: Table 3.4 shows that *Mult* is the best option here, as it was for the lower approximation classifier in Sect. 3.3.3. It does not win on any of the datasets, but does not provide a poor result either, while the other alternatives each do on at least one of them.

Having covered the datasets listed in Table 3.4, 34 remain. In Sect. 3.3.3, these made up groups 4–8, which were mutually exclusive. Due to the different nature of the fuzzy rough lower and upper approximations, we do not follow the same division here. In particular, as stated above, the upper approximation does not suffer from the overlap issue in its value vectors, as the aggregation is computed over elements in a class and not in the complement of a class (see (3.12)).

Nevertheless, the length of the aggregation does remain a crucial aspect. Group 8 in Sect. 3.3.3 contained datasets with at most five classes and more than 4000 instances. We advised the use of *Exp* on them, a setting which is not hindered as much by the aggregation length as other alternatives are. We can make a similar observation for the upper approximation albeit not based on the overall size of the dataset. As $|C|$ values are aggregated in the calculation of the upper approximation of class C, the size of the smallest class is an important indicator here. Table 3.5 lists the results of the nine datasets for which the smallest class contains at least 400 elements. Evidently, *Exp* is the best choice on these datasets and we advise its use for this group. The poor performance of *Add* should not come as a surprise, since, as discussed above, this setting is most sensitive to the aggregation length.

Next, Table 3.6 lists the remaining datasets with an IR of at most two. With the exception of the *vehicle* dataset, *Invadd* clearly performs very well and is the preferred alternative for these datasets. The best option for *vehicle* is *Exp*, with *Strict* and *Mult* as acceptable alternatives. This dataset stands out among the ones included in Table 3.6 as having the highest number of classes. The *vehicle* dataset consists of four classes, *wine* and *contra* of three and the other datasets of only two.

Table 3.4 Balanced accuracy results of the upper approximation classifier for the datasets in the first three groups

Dataset	*Strict*	*Add*	*Exp*	*Invadd*	*Mult*
Only nominal features					
breast	0.4946	**0.5908**	0.5537	0.5826	0.5783
car	0.2538	0.3802	0.2665	0.3606	**0.4031**
flare	0.2603	0.4456	0.3368	**0.5090**	0.4411
nursery	0.1280	0.2842	0.2291	**0.2972**	0.2656
splice	0.5610	**0.5928**	0.5426	0.5737	0.5904
tic-tac-toe	0.5049	**0.5902**	0.5211	0.5658	0.5659
zoo	**0.9300**	0.9113	0.8967	0.9205	0.9163
Mean	0.4475	0.5422	0.4780	**0.5442**	0.5372
Perfectly balanced, numerical, low complexity					
mov_lib	**0.8722**	0.6311	0.8533	0.7667	0.7878
segment	**0.9766**	0.8688	0.9732	0.9372	0.9558
texture	0.9869	0.8216	**0.9876**	0.9335	0.9698
vowel	**0.9939**	0.6141	0.9879	0.9394	0.9697
Mean	**0.9574**	0.7339	0.9505	0.8942	0.9208
Numerical, at least 30 features, at most 1000 instances					
derma	0.9554	0.9488	**0.9765**	0.9709	0.9643
ionosphere	**0.8777**	0.7321	0.8684	0.8059	0.8311
sonar	0.8515	0.7962	0.8592	**0.8928**	0.8569
spectfheart	0.6165	0.6387	0.6295	**0.7310**	0.7100
wdbc	0.9507	0.9313	**0.9655**	0.9412	0.9473
Mean	0.8503	0.8094	0.8598	**0.8683**	0.8619

Table 3.5 Balanced accuracy results of the upper approximation classifier on the datasets for which the smallest class contains at least 400 elements

Dataset	nMinClass	*Strict*	*Add*	*Exp*	*Invadd*	*Mult*
mammo	403	0.7456	0.8205	0.8082	**0.8214**	0.8103
marketing	505	0.2397	0.2532	0.2673	**0.2780**	0.2709
satimage	626	0.8946	0.7878	**0.9008**	0.8453	0.8624
titanic	711	0.5211	0.6997	0.6812	**0.7109**	0.7083
phoneme	1586	0.8738	0.7730	**0.8743**	0.8165	0.8455
spambase	1812	0.8987	0.8027	**0.9091**	0.8730	0.8447
banana	2376	0.8728	0.7246	0.8929	0.8848	**0.8975**
ring	3664	**0.7181**	0.5000	0.6924	0.5139	0.5742
twonorm	3697	0.9462	**0.9757**	0.9619	0.9754	0.9723
Mean		0.7456	0.7041	**0.7765**	0.7466	0.7540

Table 3.6 Balanced accuracy results of the upper approximation classifier on the datasets for which the smallest class contains fewer than 400 elements and with an IR of at most two

Dataset	IR	*Strict*	*Add*	*Exp*	*Invadd*	*Mult*
vehicle	1.10	0.6942	0.5857	**0.7104**	0.6457	0.6890
monk-2	1.12	0.7409	0.9052	0.9059	**0.9171**	0.8455
crx	1.21	0.7715	0.8695	0.8326	**0.8706**	0.8650
australian	1.25	0.7250	**0.8730**	0.7946	0.8675	0.8513
heart	1.25	0.7808	0.8225	0.8058	**0.8242**	0.8117
bupa	1.38	0.6160	0.6422	0.6498	**0.6782**	0.6696
wine	1.48	0.9631	0.9536	0.9673	**0.9679**	**0.9679**
bands	1.70	**0.7308**	0.6509	0.7164	0.7173	0.6439
wisconsin	1.86	0.9556	0.9301	0.9654	0.9497	**0.9706**
pima	1.87	0.6516	0.7014	0.6754	**0.7040**	0.6778
saheart	1.89	0.5853	0.6620	0.6104	**0.6769**	0.6315
contra	1.89	0.4038	0.4880	0.4449	**0.4914**	0.4497
Mean		0.7182	0.7570	0.7566	**0.7759**	0.7561

Table 3.7 Balanced accuracy results of the upper approximation classifier on the datasets for which the smallest class contains fewer than 400 elements and with an IR higher than two

Dataset	nCl	*Strict*	*Add*	*Exp*	*Invadd*	*Mult*
german	2	0.5619	**0.6576**	0.5750	0.5740	0.5629
haberman	2	0.5360	**0.6363**	0.5475	0.5651	0.5244
balance	3	0.5906	**0.6941**	0.6484	0.6528	0.6056
thyroid	3	0.6219	**0.6389**	0.5851	0.6029	0.3890
cleveland	5	0.3001	**0.3411**	0.2998	0.3050	0.2556
page-blocks	5	0.7610	0.6580	**0.7775**	0.6755	0.5868
automobile	6	**0.7471**	0.5788	0.6520	0.5863	0.6053
glass	6	**0.7106**	0.6504	0.6804	0.6876	0.6792
winequal-r	6	**0.3616**	0.3557	0.3458	0.3505	0.3272
winequal-w	7	**0.4498**	0.2930	0.4313	0.3572	0.3120
ecoli	8	0.7170	0.6992	0.7554	**0.7737**	0.7392
yeast	10	0.5181	**0.5779**	0.5414	0.5583	0.5352
abalone	28	0.1116	0.1236	0.1245	**0.1402**	0.1321

Finally, the results on the remaining datasets are collected in Table 3.7. The smallest class of these datasets contains fewer than 400 instances and the overall IR exceeds two. The datasets are sorted according to the number of classes (nCl) and the preference for *Add* is clear when this value is low. For a higher number of classes (e.g. at least four), the nearest neighbour approaches *Strict* and *Exp* perform best.

3.5 Guideline Validation

We now proceed with an experimental validation of our proposed weighting scheme selection guidelines for the OWA based fuzzy rough approximations. For ease of reference, we first summarize our suggestions in a schematic manner in Sect. 3.5.1. Section 3.5.2 compares the balanced accuracy results of our proposed strategies to those of the five fixed weighting schemes. The performance on a selection of independent datasets, that have not been used in the derivations above, is evaluated in Sect. 3.5.3. As a final part of this validation, Sect. 3.5.4 evaluates the use of our guidelines within two other algorithms relying on the fuzzy rough approximation operators: an instance selection method and a weighted nearest neighbour classifier. At each step of our validation process, we are able to conclude the benefits of our proposal.

3.5.1 Guidelines Summary

As a summary of Sects. 3.3 and 3.4, we first present a schematic overview of our two proposed sets of guidelines. For the OWA based fuzzy rough lower approximation, we advise the following:

1. If the dataset contains only nominal features, use *Mult*.
2. If the dataset is perfectly balanced, has only numerical features and a low dataset complexity, use *Strict*.
3. If the dataset contains at least 30 features (all numeric) and at most 1000 instances, use *Mult*.
4. In all remaining cases:
   (a) If the dataset contains more than five classes and has an IR of at most ten, use *Exp*.
   (b) If the dataset contains more than five classes and has an IR exceeding ten, use *Strict*.
   (c) If the dataset contains at most five classes, at most 4000 instances and has an IR of at most two, use *Invadd*.
   (d) If the dataset contains at most five classes, at most 4000 instances and has an IR exceeding two, use *Add*.
   (e) If the dataset contains at most five classes and more than 4000 instances, use *Exp*.

For the OWA based fuzzy rough upper approximation, our suggestions are:

1. If the dataset contains only nominal features, use *Invadd*.
2. If the dataset is perfectly balanced, has only numerical features and a low dataset complexity, use *Strict*.

3. If the dataset contains at least 30 features (all numeric) and at most 1000 instances, use *Mult*.
4. In all remaining cases:
   (a) If the size of the smallest class is at least 400, use *Exp*.
   (b) If not and the overall IR is at most two, use *Invadd*.
   (c) If not and the number of classes is low (two or three), use *Add*.
   (d) If not, use *Strict*.

We wish to stress that since our selection strategies are based on simple data characteristics, the computation time to assess which weighting scheme should be selected based on our instructions is negligible. In a similar fashion, the effects of data characteristics on feature selection methods have been recently explored in [339].

3.5.2 Data from Table 3.1

When we follow our proposed strategy to select a weight setting for the 50 datasets in Table 3.1, the mean balanced accuracy of the lower approximation classifier increases to 0.7109. This is a noticeable increase compared to the highest value of 0.6891 listed in Sect. 3.3.2. On 25 out of the 50 datasets, our weight selection strategy chooses the weight setting with the best performance. On the 25 remaining ones, an alternative for which the balanced accuracy is at most 0.05 lower is chosen. To verify whether the increase in balanced accuracy is statistically significant, we compare the performance of the lower approximation classifier using our weight guidelines to that of the same classifier with one of the five fixed weight settings. The Wilcoxon test, reported in Table 3.8, shows that the proposed strategy outperforms every weight setting with statistical significance. This good behaviour is not unexpected, as our strategy was derived based on the performance of the weight settings on these datasets.

We repeat the same analysis for the upper approximation classifier used in Sect. 3.4. The highest mean balanced accuracy of one of the fixed weighting schemes is 0.6917 and corresponds to *Invadd* (see Sect. 3.4.1). When using our upper approximation weight selection guidelines, the mean balanced accuracy of the classifier increases to 0.7110. Our weight selection strategy chooses the weight setting with the best performance on 24 out of the 50 datasets. As for the lower approximation classifier, the use of our weighting scheme selection strategy leads to a statistically significant performance improvement compared to each of the weighting schemes, as evidenced by the Wilcoxon test results reported in Table 3.8.

3.5.3 Independent Data

In this section, we evaluate the performance of our proposed weight selection strategies on independent datasets, that is, datasets that have not been used in this chapter

Table 3.8 Results of the Wilcoxon test comparing the results of the lower and upper approximation classifiers using our weight guidelines to the versions using one of the five fixed weight settings on the datasets in Table 3.1. P-values implying significant differences at the 5% significance level are printed in bold

	Lower			Upper		
	R^+	R^-	p	R^+	R^-	p
Proposal versus *Strict*	1048.5	226.5	0.000071	1024.0	201.0	0.000042
Proposal versus *Add*	1127.0	98.0	0.00000041	1039.0	236.0	0.000102
Proposal versus *Exp*	999.5	275.5	0.000466	952.0	273.0	0.000719
Proposal versus *Invadd*	921.0	304.0	0.002114	828.5	396.5	0.031272
Proposal versus *Mult*	956.0	319.0	0.002074	1012.0	263.0	0.00029

Table 3.9 Description of the 20 independent datasets used in Sect. 3.5.3. We list the number of features (nFeat), the number of instances (nInst), the number of classes (nCl) and the IR. Together with the number of features, we specify whether they are all nominal (Y) or not (N)

Name	nFeat	nInst	nCl(IR)	Name	nFeat	nInst	nCl(IR)
appendicitis	7(N)	106	2(4.05)	iris	4(N)	150	3(1)
banknote	4(N)	1372	2(1.25)	letter	16(N)	20000	26(1.11)
biodeg	40(N)	1055	2(1.96)	magic	10(N)	19020	2(1.84)
credit	15(N)	653	2(1.21)	messidor	19(N)	1151	2(1.13)
ctg	21(N)	2126	10(10.92)	mushroom	22(Y)	5644	2(1.62)
eye_detection	14(N)	14980	2(1.23)	optdigits	64(N)	5620	10(1.03)
faults	33(N)	1941	2(1.88)	penbased	16(N)	10992	10(1.08)
grub	8(N)	155	4(2.58)	seismic	18(N)	2584	2(14.2)
hepatitis	19(N)	80	2(5.15)	sensor	24(N)	5456	4(6.72)
housevotes	16(Y)	232	2(1.15)	transfusion	4(N)	748	2(3.2)

thus far. This evaluation will further reinforce the demonstrated efficacy and validity of our proposal. The 20 datasets used in this section were obtained from the KEEL, UCI and Weka platforms and are listed in Table 3.9. They present the different characteristics used in our division of the datasets in Sects. 3.3 and 3.4. These datasets and their partitions can be downloaded from http://www.cwi.ugent.be/sarah.php.

Lower approximation We present the balanced accuracy values of the lower approximation classifier in Table 3.10. Apart from the results obtained using our weight selection strategy proposed in Sect. 3.3, the table also shows the performance of the five individual weight settings. As before, the best value for each dataset is printed in boldface, while any value that is more than 0.05 lower than this optimum

Table 3.10 Balanced accuracy results of the classification by the OWA based fuzzy rough lower approximation operator on independent datasets. With the results of our weighting scheme selection strategy, we list the selected weight setting between brackets. The bottom three lines present the results of the Wilcoxon test comparing our proposed strategy (corresponding to R^+) to the other weighting schemes

Dataset	*Strict*	*Add*	*Exp*	*Invadd*	*Mult*	Proposal
appendicitis	0.7514	**0.7938**	0.7479	0.7868	0.7542	**0.7938** (A)
banknote	**0.9987**	0.9247	**0.9987**	0.9961	**0.9987**	0.9961 (I)
biodeg	0.8153	0.7139	0.8381	0.8216	**0.8513**	0.8216 (I)
credit	0.8194	0.8695	0.8582	**0.8704**	0.8571	**0.8704** (I)
ctg	**0.7379**	0.3999	0.7301	0.6283	0.6793	**0.7379** (S)
eye_detection	0.8456	0.5903	**0.8615**	0.8147	0.7226	**0.8615** (E)
faults	0.9897	0.6748	**0.9919**	0.9611	0.9904	0.9611 (I)
grub	0.2638	**0.3963**	0.3075	0.3638	0.3500	**0.3963** (A)
hepatitis	0.8184	0.8232	0.8199	**0.8633**	0.7715	0.8232 (A)
housevotes	0.9137	0.9017	**0.9209**	0.9125	0.9125	0.9125 (M)
iris	0.9333	**0.9533**	0.9400	**0.9533**	**0.9533**	0.9333 (S)
letter	0.9428	0.6124	**0.9632**	0.9556	0.9537	**0.9632** (E)
magic	0.8129	0.7606	**0.8384**	0.8126	0.7907	**0.8384** (E)
messidor	0.6282	0.6056	0.6534	0.6546	**0.6638**	0.6546 (I)
mushroom	0.9593	0.8026	**0.9749**	0.9713	0.9034	0.9034 (M)
optdigits	0.9843	0.9094	**0.9857**	0.9798	0.9834	**0.9857** (E)
penbased	0.9936	0.7624	**0.9939**	0.9671	0.9902	**0.9939** (E)
seismic	0.5514	**0.7015**	0.5299	0.5872	0.4998	**0.7015** (A)
sensor	0.9224	0.7393	**0.9255**	0.9024	0.9080	**0.9255** (E)
transfusion	0.5708	0.6528	0.6048	**0.6745**	0.6047	0.6528 (A)
Mean	0.8126	0.7294	0.8242	0.8238	0.8069	**0.8363**
# best/poor	2/4	4/12	10/3	4/2	4/6	11/1
R^+	167.5	178.0	122.5	144.5	165.5	–
R^-	42.5	12.0	87.5	65.5	44.5	–
p	0.01821	0.00027	0.47034	0.14827	0.02272	–

is underlined. The advantages of our proposal are clear. We obtain the highest balanced accuracy on average, the most wins and the fewest poor results. The table also shows that our selection strategy is not infallible either, as it does not perform well on the *mushroom* dataset. The *Mult* setting is selected, because this dataset contains only nominal features. However, if we were to ignore this particular guideline, *mushroom* would be assigned to group 8, because it has 2 classes and 5644 instances. In this case, it would be processed with *Exp*, which is the preferred setting for this dataset.

We compare the performance of our lower approximation weight guidelines to the five weight settings using a Wilcoxon test and include the results in Table 3.10. We can conclude that it performs significantly better than *Strict*, *Add* and *Mult*. We cannot

Table 3.11 Balanced accuracy results of the classification by the OWA based fuzzy rough upper approximation operator on independent datasets. With the results of our weighting scheme selection strategy, we list the selected weight setting between brackets. The bottom three lines present the results of the Wilcoxon test comparing our proposed strategy (corresponding to R^+) to the other weighting schemes

Dataset	*Strict*	*Add*	*Exp*	*Invadd*	*Mult*	Proposal
appendicitis	0.7514	**0.7938**	0.7479	0.7868	0.7542	**0.7938** (A)
banknote	**0.9987**	0.9247	**0.9987**	0.9961	**0.9987**	**0.9987** (E)
biodeg	0.8153	0.7139	0.8381	0.8216	**0.8513**	0.8216 (I)
credit	0.8194	0.8695	0.8582	**0.8704**	0.8571	**0.8704** (I)
ctg	0.7384	0.6812	0.7269	**0.7599**	0.6293	0.7384 (S)
eye_detection	0.8456	0.5903	**0.8615**	0.8147	0.7226	**0.8615** (E)
faults	0.9897	0.6748	**0.9919**	0.9615	0.9908	**0.9919** (E)
grub	0.2875	**0.3963**	0.3225	0.3938	0.3825	0.2875 (S)
hepatitis	0.6915	0.8058	0.7215	**0.8215**	0.7715	0.8058 (A)
housevotes	0.9137	0.9017	**0.9209**	0.9125	0.9125	0.9125 (I)
iris	0.9333	**0.9533**	0.9400	**0.9533**	**0.9533**	0.9333 (S)
letter	0.9537	0.6558	**0.9649**	0.8952	0.9170	**0.9649** (E)
magic	0.7821	0.7019	**0.8020**	0.7561	0.7907	**0.8020** (E)
messidor	0.6282	0.6056	0.6534	0.6546	**0.6638**	0.6534 (E)
mushroom	0.9593	0.8026	**0.9749**	0.9713	0.9034	0.9713 (I)
optdigits	0.9843	0.9280	**0.9861**	0.9733	0.9818	**0.9861** (E)
penbased	**0.9936**	0.8151	0.9936	0.9274	0.9904	0.9936 (E)
seismic	0.5514	**0.7015**	0.5299	0.5872	0.4998	**0.7015** (A)
sensor	0.9224	0.6510	**0.9252**	0.8845	0.9033	0.9224 (S)
transfusion	0.5708	0.6528	0.6048	**0.6745**	0.6047	0.6528 (A)
Mean	0.8065	0.7410	0.8181	0.8208	0.8039	**0.8332**
# best/poor	2/5	4/12	9/4	5/3	4/5	9/1
R^+	192.5	184.0	119.0	139.0	164.5	–
R^-	17.5	26.0	91.0	71.0	45.5	–
p	0.0004387	0.0019856	0.2720952	0.1788100	0.0253	–

conclude that our approach provides a significantly better result than *Invadd* or *Exp*, although the higher values for R^+ do indicate a preference in our favour. Table 3.10 also demonstrates that a higher balanced accuracy is obtained on average in this case, as well as a higher number of wins and lower number of poor results.

Upper approximation The analogous results for the upper approximation classifier are presented in Table 3.11 and confirm the validity of our analysis conducted in Sect. 3.4. When the classifier incorporates our weight selection guidelines, its mean balanced accuracy is noticeably higher than when fixing the weighting scheme to one of the other five alternatives. A poor result is obtained on only one dataset,

namely *grub*. The *Strict* weighting scheme is selected, while *Add*, *Invadd* or *Mult* yield better balanced accuracy results. This dataset does not belong to any of the first three groups nor does its smallest class contain at least 400 instances or is it overall IR at most two. This makes it fit into the group of datasets listed in Table 3.7. Due to lack of evidence in our training data, we only advised the use of *Add* on these datasets when the number of classes is two or three. The *grub* dataset contains four classes, such that our guidelines recommend the use of *Strict* for it.

We compare the performance of our upper approximation weight guidelines to the five weight settings by means of the Wilcoxon test and include these results in Table 3.11. The performance differences between our weight selection strategy and the fixed weighting schemes are found to be statistically significant for *Strict*, *Add* and *Mult*. As for the lower approximation classifier, we cannot conclude that our approach significantly outperforms *Exp* or *Invadd*, although the higher R^+ values are again a point in our favour.

Combination As an additional validation, we have checked the performance of the classifier that uses the combination of the OWA based fuzzy rough lower and upper approximation membership degrees as predictor. In particular, to classify an instance x, its score for each class C is computed as $\frac{\underline{C}(x)+\overline{C}(x)}{2}$ and the instance is assigned to the class for which this value is largest. When we fix the OWA weighting schemes to one of the five alternatives included in this study, we obtain mean balanced accuracies of 0.6707 (*Strict*), 0.6606 (*Add*), 0.6901 (*Exp*), 0.6912 (*Invadd*) and 0.6818 (*Mult*) on the 50 datasets in Table 3.1 and 0.8065 (*Strict*), 0.7416 (*Add*), 0.8182 (*Exp*), 0.8221 (*Invadd*) and 0.8046 (*Mult*) on the 20 independent datasets in Table 3.9. When using our proposed guidelines for the lower and upper approximations, we obtain mean balanced accuracies of 0.7145 and 0.8336 respectively, which both show a clear improvement over the values for the fixed weighting schemes. These observations

Table 3.12 Results of the Wilcoxon test comparing the results of the combination classifier using our weight guidelines to the versions using one of the five fixed weight settings. P-values implying significant differences at the 5% significance level are printed in bold

	Data Table 3.1			Data Table 3.9		
	R^+	R^-	p	R^+	R^-	p
Proposal versus *Strict*	1075.5	199.5	0.000023	171.5	18.5	**0.0010682**
Proposal versus *Add*	1051.0	224.0	0.000063	187.0	23.0	**0.0012092**
Proposal versus *Exp*	963.5	261.5	0.000456	137.5	72.5	0.1981229
Proposal versus *Invadd*	950.5	324.5	0.002476	129.5	60.5	0.17533
Proposal versus *Mult*	989.5	235.5	0.000173	170.5	39.5	**0.012848**

are further supported by the results of the Wilcoxon tests as presented in Table 3.12. We briefly note that there are only small differences in performance when either the lower approximation, upper approximation or combination classifier are used with our optimized weight selection strategy.

3.5.4 Other Applications

As a final validation step, we use our guidelines within the OWA-FRPS instance selection method [415] and the POSNN classifier [413, 416]. Both use the fuzzy rough positive region $POS(\cdot)$. In a classification dataset, $POS(x)$ is computed as the membership degree of instance x to the OWA based lower approximation of its own decision class [107]. In [413], the use of *Invadd* was advised for both methods,

Table 3.13 Balanced accuracy results of the 1NN classifier after prototype selection by OWA-FRPS or without preprocessing (None). The OWA-FRPS method uses either fixed *Invadd* weights or those weights advised by our guidelines. We report the reduction in the number of instances after the prototype selection step as well

Dataset	None	*Invadd*		Proposal	
	Balacc	Balacc	Red.	Balacc	Red.
appendicitis	0.7514	0.7500	0.5755	0.7382	0.3721
banknote	0.9987	0.9987	0.0449	0.9987	0.0449
biodeg	0.8153	0.8187	0.3188	0.8187	0.3188
credit	0.8194	0.8642	0.4013	0.8642	0.4013
ctg	0.7438	0.7289	0.0428	0.7205	0.1106
eye_detection	0.8456	0.8436	0.0945	0.8465	0.2110
faults	0.9897	0.9897	0.0000	0.9897	0.0000
grub	0.3013	0.3688	0.6101	0.3238	0.7905
hepatitis	0.6915	0.7015	0.4699	0.7860	0.5404
housevotes	0.8984	0.9038	0.2213	0.9112	0.4797
iris	0.9333	0.9400	0.2141	0.9400	0.1096
letter	0.9544	0.9548	0.0013	0.9536	0.0123
magic	0.7821	0.7807	0.1612	0.7988	0.2793
messidor	0.6282	0.6316	0.4909	0.6316	0.4909
mushroom	0.8873	0.8996	0.0000	0.8826	0.0000
optdigits	0.9845	0.9849	0.0209	0.9843	0.0837
penbased	0.9937	0.9937	0.0011	0.9937	0.0066
seismic	0.5514	0.5000	0.9865	0.5000	0.9975
sensor	0.9224	0.9226	0.0013	0.9205	0.0130
transfusion	0.5837	0.6193	0.8702	0.6636	0.8219
Mean	0.8038	0.8097	0.2763	0.8133	0.3042

regardless of the dataset on which they are applied. As we observed in Sect. 3.3.2, this weighting scheme can appear as a good default choice based on its average performance, but does not necessarily perform well on all datasets. We now evaluate whether using our guidelines instead benefits the performance of OWA-FRPS and POSNN. We use the independent datasets listed in Table 3.9.

Table 3.13 lists the results associated with the OWA-FRPS method. This algorithm computes the quality of all training instances as their membership degree to the positive region and derives a threshold above which the quality is deemed sufficiently high. Only instances with a quality exceeding this threshold are retained in the dataset. We combine it with the 1NN classifier as done in [413, 415]. When we fix the weighting scheme to *Invadd*, the average reduction in the number of instances is 27.63% and the mean balanced accuracy of 1NN after OWA-FRPS is 0.8097. Using our guidelines results in an average reduction of 30.42% and a mean balanced accuracy of 0.8133. This provides three advantages: (i) we relieve the user from setting the OWA weighting scheme, when they are not fully comfortable with using a default option, (ii) on average, the datasets are reduced slightly more and (iii) the classification performance of the posterior classifier is maintained, even somewhat improved (albeit not statistically significantly).

The POSNN classifier is a weighted extension of the fuzzy nearest neighbour classifier of [257]. The contribution of each neighbour is weighted by its membership degree to the OWA based positive region. Note that the lower approximation operator is not used as a predictor, as done in the construction of our guidelines in Sect. 3.3, but rather as a weighting mechanism. We have set the number of neighbours to ten and report the balanced accuracy results in Table 3.14. The use of *Invadd*, as recommended in [413], leads to a mean balanced accuracy of 0.8084 on the datasets in Table 3.9, while using our guidelines increases this value to 0.8101. According to the results of the Wilcoxon test, this improvement is significant ($R^+ = 148.5$, $R^- = 41.5$, $p = 0.029774$).

Table 3.14 Balanced accuracy results for the POSNN classifier with $k = 10$, fixing the weighting scheme to *Invadd* or using our proposed lower approximation weight guidelines

Dataset	*Invadd*	Proposal	Dataset	*Invadd*	Proposal
appendicitis	0.7542	0.7542	iris	0.9400	0.9400
banknote	0.9967	0.9967	letter	0.9518	0.9525
biodeg	0.8449	0.8449	magic	0.7925	0.7997
credit	0.8571	0.8571	messidor	0.6613	0.6613
ctg	0.6295	0.6354	mushroom	0.8925	0.8925
eye_detection	0.8397	0.8408	optdigits	0.9825	0.9834
faults	0.9828	0.9828	penbased	0.9893	0.9893
grub	0.3400	0.3400	seismic	0.5068	0.5156
hepatitis	0.7554	0.7554	sensor	0.9007	0.8991
housevotes	0.9151	0.9185	transfusion	0.6362	0.6430
			Mean	0.8084	0.8101

3.6 Conclusion

In this chapter, we have recalled the OWA based fuzzy rough set model from [108], a noise-tolerant alternative to traditional fuzzy rough sets. It replaces the minimum and maximum operators in the original fuzzy rough lower and upper approximations by OWA aggregations. The OWA based fuzzy rough approximation operators are highly interpretable and easy to implement and offer a robustness in the presence of noise that has been clearly demonstrated in previous work. Nevertheless, OWA based fuzzy rough set approaches have not experienced the same popularity as the traditional fuzzy rough set model. We believe that further advances can be made in many areas by exploiting the strengths and interpretability of this versatile and flexible model. As an OWA aggregation of a set of values relies on the definition of a weight vector, the defining component of the OWA based fuzzy rough approximations is their choice of weighting scheme. If guidelines are available for the selection of an appropriate weighting scheme, the OWA based model would, in our opinion, become more accessible to the research community and benefit it as a whole. We have observed the strength of this fuzzy rough set model in our own research (as described in later chapters) and are confident that our conclusions presented here can contribute to a more widespread adoption of OWA based approaches.

Through systematic and rigorous comparison of five different weighting schemes, we have been able to provide weighting scheme selection strategies for both the OWA based lower and upper approximation operators. We have explained the behaviour and suitability of the alternative options in detail, such that the actions of the OWA based fuzzy rough operators can be understood at a fundamental level. The efficacy of our weighting scheme selection strategy is further supported and validated by an evaluation on independent datasets and in different applications.

However, we should be forthcoming with respect to one limitation of the presented work. Our guidelines have been proposed based on the evidence provided by the datasets in Table 3.1. As a consequence, any data characteristics that are not sufficiently represented among this group could not have been taken into account in our derivations. In particular, the largest dataset included in this study consists of only 20000 observations, a small number in several present-day applications. Although we do not believe our conclusions to become invalid when the number of instances is considerably higher (i.e., the exponential and strict weights will likely remain the only suitable ones), we can not support this statement with provided empirical evidence. In point of fact, we would argue against the direct use of OWA based fuzzy rough sets on excessively large datasets, but postpone this discussion to the concluding Chap. 8.

One could put two other aspects of our approach up for scrutiny, namely our selection of the five weighting schemes and the similarity measure (3.13). These can, in our opinion, be justified. Additional weighting schemes can be found in the literature, as described in Sect. 3.2.2 for instance, or even be devised by the reader. In light of the different properties of each scheme (Sect. 3.2), we nevertheless feel that we have made an appropriate selection. When the reader wishes to assess

the adequacy of their custom weighting scheme within OWA based fuzzy rough sets, they should be able to do so based on our discussions in Sects. 3.3 and 3.4. We have based ourselves on the general defining characteristics of the schemes, in particular in Sect. 3.3.4, and our conclusions should carry over to other weighting scheme alternatives as well. With regard to the second point, we have chosen to fix the fuzzy relation measuring similarity between instances to expression (3.13). This is a reasonable and intuitive similarity measure, which has been used in previous studies on fuzzy rough classifiers as well. Alternatives exist, but we do not expect that our observations and conclusions in Sects. 3.3 and 3.4 will greatly change when a different (sensible) relation is used. When an alternative similarity measure is more suitable for a particular dataset, the performance of the classifier will improve, but we believe that the relative rankings of the weighting schemes will remain the same. Since our focus has been on the latter aspect, optimizing the similarity relation is of secondary importance and our default use of (3.13) is justified. Our conclusions are not strongly based on the instance similarity values. Naturally, like the nearest neighbour classifier, any fuzzy rough classifier may benefit from the application of metric learning (see Sect. 2.2.1). A user that wishes to apply our simple classifier in a prediction task can opt to use our guidelines in conjunction with a data-dependent similarity measure derived with a metric learning technique.

Chapter 4
Learning from Imbalanced Data

In this chapter, we consider the classification of imbalanced data. When a dataset presents an imbalance between its classes, that is, an uneven distribution of observations among them, the classification task is inherently more challenging. Traditional classification algorithms (see Sect. 2.2) tend to favour majority over minority class elements due to their incorrect implicit assumption of an equal class representation during learning. As a consequence, the recognition of minority instances is hampered. Since minority classes are usually the ones of interest, custom techniques are required to deal with such data skewness. We study them in this chapter. Section 4.1 discusses the traditional setting of binary class imbalance, where the observations are divided between one majority and one minority class. We recall the challenges and solutions associated with this classification problem. In Sect. 4.2, we transfer our attention to the more general setting of multi-class imbalance, considering datasets with three or more classes of (considerably) unequal sizes. The focus of the research community has shifted to this task in recent years. Section 4.3 is dedicated to our FROVOCO method, proposed as a novel fuzzy rough set based method for multi-class imbalanced classification. A thorough experimental evaluation of our proposal is conducted in Sect. 4.4. We validate the internal settings of FROVOCO and compare it to the state-of-the-art in this domain, demonstrating its superior classification performance. Finally, Sect. 4.5 concludes the chapter.

4.1 Binary Class Imbalance

Research on class imbalance has long been focused on two-class classification problems and advances still continue to be made (e.g. [198, 290, 394]). In this setting, elements from the majority class (greatly) outnumber minority class instances, although the latter is often the class of interest. The relative lack of information on the

S. Vluymans, *Dealing with Imbalanced and Weakly Labelled Data in Machine Learning using Fuzzy and Rough Set Methods*, Studies in Computational Intelligence 807, https://doi.org/10.1007/978-3-030-04663-7_4

minority concept leads to a poor recognition of these elements by the classification model. Minority class misclassifications are a logical (but undesired) consequence. The traditional binary class imbalance problem is discussed in Sect. 4.1.1, while Sect. 4.1.2 lists the approaches to deal with this challenge. In Sect. 4.1.3, we recall the IFROWANN method, a fuzzy rough set based classifier for binary imbalanced data.

4.1.1 The Class Imbalance Problem

The *class imbalance problem*, a term coined in [239], refers to the challenges posed on traditional classification by an uneven distribution of elements across the decision classes. Review works on (binary) imbalanced data include [57, 211, 391]. The recent contribution [57] also considers imbalanced regression, when certain values of the target value (e.g. extremely low ones) are under-represented in the training set. Traditionally, in a two-class imbalanced problem, the minority class is considered the *positive* class, while the majority elements are labelled as *negative*. These monikers indicate that the minority class is usually the class of interest [306]. The sheer abundance of negative information hinders the recognition capability of positive instances. Furthermore, as already noted in Sect. 2.3.1.2, general classification evaluation metrics can provide misleading results when the smallest class is indeed the one of interest [57]. The evaluation measure should not allow for a strong performance on the majority class to overshadow a poor result on the minority class and should instead provide a balanced evaluation of the performance on both classes. The example measures listed in Sect. 2.3.1.2 are valid options in this setting.

The most defining characteristic of an imbalanced dataset is its skewed class distribution [306]. This uneven division of observations across the classes can easily be measured by the *imbalance ratio* (IR). As recalled in (1.1), this value is defined as the ratio of the sizes of the overall majority and minority classes. Its definition implies that the IR is a number larger than or equal to one. When $\text{IR} = 1$, the dataset is perfectly balanced. Larger values indicate a larger difference in the class sizes. Any dataset with an imbalance ratio exceeding 1.5 is considered imbalanced, while $\text{IR} = 9$ is often used as the threshold above which datasets are regarded as highly imbalanced (e.g. [305, 419]). It has been shown that the IR is not the only property of imbalanced datasets that poses challenges to learning algorithms. Several authors (e.g. [28, 147, 240, 306]) describe how general data-related challenges like small sample size, areas of small disjuncts and under-represented sub-concepts, high dimensionality, class overlap, class noise, lack of density and the dataset shift problem can have a more pronounced effect in class imbalanced data. The study of [309] showed that the IR in itself is insufficient to predict the performance of a classification method on an imbalanced dataset. The actual complexity of the dataset and intrinsic characteristics influence the classifier recognition capability as well.

Application areas wherein binary class imbalance is naturally encountered include medical diagnosis and prediction (e.g. [22, 344, 382]), bioinformatics (e.g. [23, 139, 403]), corporation and market prediction (e.g. [289, 511]) and anomaly detection (e.g. [154, 442]).

4.1.2 Dealing with Binary Class Imbalance

Solutions to dealing with binary class imbalance are often categorized into three groups [306, 308]: data level approaches, algorithm level solutions and cost-sensitive methods.

Data level approaches Data level solutions to class imbalance are preprocessing techniques that are applied on the training set before constructing a classification model. Their core aim is to attain a better balance between the classes, be it perfect balance at IR $= 1$ or simply a reduction of the IR to an acceptable level. It has been shown that aspiring perfect balance is not advantageous in every setting [259]. As preprocessing techniques, these data level methods are generally independent of the subsequent classifier, which implies an inherent level of versatility. In essence, they resample the dataset and we can distinguish between the three settings visualized in Fig. 4.1:

1. **Undersampling**: these methods remove instances from the dataset in order to reduce the imbalance ratio. The simplest way to achieve this effect is to ran-

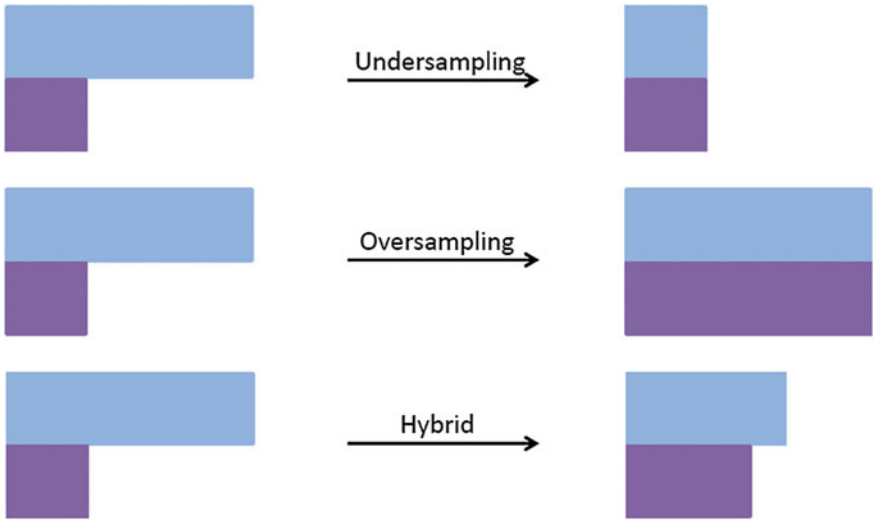


Fig. 4.1 Illustration of the three resampling techniques dealing with binary class imbalance. Undersampling and oversampling methods commonly strive for a perfect balance between the two classes

domly discard some elements from the majority class. This idea is incorporated in the random undersampling method proposed in [28]. Alternative undersampling procedures based on more involved heuristics have been proposed in e.g. [188, 267, 275, 332, 467, 469]. Most undersampling methods leave the minority class untouched and only remove majority class elements, but it has been shown that allowing a (small) reduction of the former holds benefits of its own [420].

2. **Oversampling**: these techniques take the opposite approach. Instead of reducing the majority class, the size of the minority class is increased. New minority elements are added to the training set as (i) duplicates or perturbed variants of original instances or (ii) results of interpolation. As the most straightforward approach, similar to the random undersampling technique, random oversampling randomly selects minority class elements for duplication [28]. Arguably the most popular oversampling method is the SMOTE technique [81], which constructs new minority instances by means of linear interpolation between existing minority elements and one of their nearest minority class neighbours. It has repeatedly been pointed out that the application of SMOTE can result in an overgeneralization of the minority class and several extensions have been proposed to address this issue. Furthermore, its instance generation procedure is not always justified [35]. As described in [57], these extensions can be divided into three groups (i) methods that apply a pre- or post-processing technique in conjunction with SMOTE (e.g. [28, 359, 360, 417]), (ii) methods that limit the generation of minority elements to specific regions of the feature space (e.g. [26, 204, 210]) and (iii) methods that modify the element generation process (e.g. [64–66, 313]).
3. **Hybrid methods**: hybrid data level solutions combine undersampling and oversampling within their data balancing approach. They either combine over- and undersampling (performing both procedures at once) or perform undersampling after oversampling as a data cleaning step. In fact, the oversampling methods that combine SMOTE with post-processing fit into the latter group. Other proposals can be found in e.g. [334, 386].

Algorithm level approaches Instead of modifying the training data, algorithm level approaches adapt the learning phases of classifiers to take into account the skewness in the class distribution. Several of the traditional classification paradigms listed in Sect. 2.2 have inspired imbalance-resistant classifiers (e.g. [29, 101, 146, 232, 380]). The IFROWANN method, discussed below in Sect. 4.1.3, is another example. We also include ensemble approaches (Sect. 2.2.7) to imbalanced classification in this category, although some authors list these as a separate group. Traditional ensemble settings like boosting [373, 374] and bagging [58] have been modified to deal with class imbalanced problems [174]. Imbalance-resistant ensemble methods based on the traditional AdaBoost method have been proposed in e.g. [140, 253, 390, 399]. Several authors integrate the data level solutions listed above into the boosting or bagging schemes, producing methods like SMOTEBoost [83], MSMOTEBoost [230], RUSBoost [378], EUSBoost [177], OverBagging [437], UnderBagging [24], UnderOverBagging [437] and SMOTEBagging [437]. The EasyEnsemble and BalanceCascade methods from [299] were proposed as ensembles of ensembles and

combine boosting and bagging. Other ensemble methods for imbalanced data classification aim to improve the balance between classes and guarantee sufficient diversity between ensemble members at the same time (e.g. [95, 128]).

Cost-sensitive methods A third solution group, although these techniques could again be catalogued under the algorithm level approaches as well, are the cost-sensitive methods. These algorithms incorporate misclassification costs in their internal workings, where the cost of a prediction error on a minority element is larger than the cost of misclassifying a majority instance [308]. By integrating these costs, the invalid assumption of equal class distributions made by traditional classifiers can be dealt with. Several of the general classification models discussed in Sect. 2.2 have been extended in this manner including cost-sensitive nearest neighbour classification [206], cost-sensitive decision trees [265, 303, 400], cost-sensitive neural networks [193, 507] and cost-sensitive support vector machines [418, 501].

4.1.3 The IFROWANN Method

In [361], the authors proposed the IFROWANN method, short for Imbalanced Fuzzy Rough Ordered Weighted Average Nearest Neighbour classification. It is an algorithm level solution to binary class imbalance. As its name implies, the method relies on OWA based fuzzy rough set theory (Sect. 3.1.3).

IFROWANN was proposed as an extension of the fuzzy rough nearest neighbour classifier (FRNN, [242]) to deal with two-class imbalanced data. The FRNN method classifies an instance x by (i) computing its membership degree to the fuzzy rough lower and upper approximations of all decision classes and (ii) assigning x to the class C for which $\frac{\underline{C}(x)+\overline{C}(x)}{2}$ is largest. FRNN uses the traditional fuzzy rough set model in its calculations. To deal with the challenges posed by class imbalance, the authors of [361] introduce some changes in the FRNN algorithm. First, by focusing on datasets with only two classes, they show that the evaluation $\frac{\underline{C}(x)+\overline{C}(x)}{2}$ can be restricted to the lower approximation operator, as the upper approximation does not carry any additional information on top of that provided by the lower approximation in this setting. Using $\underline{C}(x)$ instead of $\frac{\underline{C}(x)+\overline{C}(x)}{2}$ as class score formula results in exactly the same predictions. IFROWANN uses the OWA based fuzzy rough set model to compute the membership degree of elements to the lower approximation of the two classes. When one of the popular fuzzy implicators (Kleene-Dienes, Łukasiewicz or Reichenbach, see Table 1.6) is used, expression (3.7) further simplifies to

$$\underline{C}(x) = \text{OWA}_{W_L}(\{1 - R(x, y) \mid y \notin C\}), \tag{4.1}$$

where W_L is the OWA weight vector. When P and N are the positive (minority) and negative (majority) classes respectively, IFROWANN computes $\underline{P}(x)$ and $\underline{N}(x)$ and classifies x as positive when $\underline{P}(x) \geq \underline{N}(x)$ and as negative otherwise.

To deal with the imbalanced nature of the dataset, the authors of [361] allow the use of different OWA weighting schemes for the two classes and evaluate several combinations in their study. In particular, the following six configurations were proposed in [361], where the first and second components refer to the weight vectors for $\underline{P}$ and $\underline{N}$ respectively:

$$\begin{aligned}
\mathcal{W}_1 &= \langle W_L^{add}, W_L^{add} \rangle, \\
\mathcal{W}_2 &= \langle W_L^{add}, W_L^{exp} \rangle, \\
\mathcal{W}_3 &= \langle W_L^{exp}, W_L^{add} \rangle, \\
\mathcal{W}_4 &= \langle W_L^{exp}, W_L^{exp} \rangle, \\
\mathcal{W}_5 &= \left\langle W_L^{add,\gamma}, W_L^{add} \right\rangle, \\
\mathcal{W}_6 &= \left\langle W_L^{add,\gamma}, W_L^{exp} \right\rangle.
\end{aligned}$$

We refer to Sect. 3.2.1 for the definition of the additive and exponential weight vectors. Weight vector $W_L^{add,\gamma}$ in the fifth and sixth settings is defined as

$$W_L^{add,\gamma} = \left\langle 0, 0, \ldots, 0, \frac{2}{r(r+1)}, \frac{4}{r(r+1)}, \ldots, \frac{2(r-1)}{r(r+1)}, \frac{2}{r+1} \right\rangle, \tag{4.2}$$

with $\gamma \in [0, 1]$ a user-defined parameter. The value r, which corresponds to the number of non-zero positions in this vector, is defined as $r = \lceil |P| + \gamma(|N| - |P|) \rceil$, with $|P|$ and $|N|$ the sizes of the positive and negative classes respectively. Note that the total length of vector $W_L^{add,\gamma}$ equals $|N|$, as this is the number of values to aggregate in the calculation of $\underline{P}(x)$ by means of (4.1). When $\gamma = 1$, r equals $|N|$ and vector W_L^{add} is retrieved.

The comprehensive experimental evaluation of [361] on a corpus of 102 binary imbalanced datasets concluded that weight combinations $\mathcal{W}_4$ and $\mathcal{W}_6$ provide the best classification results. It should be clear that $\mathcal{W}_4$ treats the two classes symmetrically and uses exponential weights to compute both $\underline{P}(x)$ and $\underline{N}(x)$. Regardless of this observation, the actual weight vectors W_L^{exp} used in the aggregations of $\underline{P}(x)$ and $\underline{N}(x)$ will be different due to a possibly large difference in aggregation length. From (4.1) and the fact that only two classes are present in the dataset, it should be clear that $\underline{P}(x)$ is an aggregation over a set of size $|N|$, while $\underline{N}(x)$ is derived based on only $|P|$ values. In an imbalanced dataset, the former value can be far larger than the latter. Configuration $\mathcal{W}_6$ uses distinct weight vector definitions for $\underline{P}(x)$ and $\underline{N}(x)$, which directly implies different actions on the two classes. As does $\mathcal{W}_4$, it uses exponential OWA weights to derive value $\underline{N}(x)$. The leading zeroes in $W_L^{add,\gamma}$ within the calculation of $\underline{P}(x)$ imply that the contribution of certain instances $y \in N$ is actively discarded in (4.1). Only the r smallest values $1 - R(x, y)$ are used, which originate from the instances $y \in N$ most similar to x. In [361], it was concluded that 0.1 is an appropriate value for the γ parameter. Looking back at its definition, this implies that many of the weights in $W_L^{add,\gamma}$ are zero and relatively few (somewhat more than $|P|$) are non-negligible.

4.2 Multi-class Imbalance

A skewed class distribution can naturally occur in datasets with more than two classes as well. Application domains wherein such multi-class imbalance is encountered include bioinformatics (e.g. [302, 475, 499]) and medical diagnosis (e.g. [92, 381, 480]). This classification task is even more challenging than its binary cousin [148]. Firstly, the higher number of classes implies a higher number of decision boundaries to learn. A second problem is that the additional data-intrinsic characteristics listed in Sect. 4.1.1 can be more strongly expressed when more classes are present. Instead of having a single minority and majority class, multi-minority and multi-majority situations can present themselves [438]. As a consequence, the imbalance ratio (1.1) can be insufficient to represent the degree of imbalance present in a dataset with more than two classes [340].

In general multi-class classification, a *decomposition scheme* can be applied to reduce the multi-class problem to several sub-problems with a lower number of classes (often, two). Two traditional examples are the *one-versus-one (OVO)* and *one-versus-all (OVA)* schemes. The OVO procedure considers each pair of classes, while the OVA scheme creates binary problems comparing one class to the union of all others. The former setting has been shown to be preferable in the presence of a skewed class distribution (e.g. [148]). By contrasting each class with the remainder of the dataset, an OVA decomposition inherently induces a larger imbalance in its binary classification tasks. In Sect. 4.2.1, we recall the one-versus-one decomposition scheme for general multi-class classification in more detail. Section 4.2.2 discusses the classifier competence issue related with the OVO procedure.

Similar to the approaches listed in Sect. 4.1.2, a variety of methods have been proposed to deal with multi-class imbalance with or without using a decomposition scheme. Section 4.2.3 presents an overview.

4.2.1 The One-Versus-One Decomposition Scheme

The OVO decomposition scheme is a traditional procedure to transform a multi-class problem to a number of two-class problems. When the dataset contains m classes, a total number of $\frac{m(m-1)}{2}$ sub-tasks are created, one for each pair of classes. A classifier is trained on each sub-problem using only the training elements belonging to the corresponding two classes as input to learn from. When classifying a new instance x, each binary classifier is fired and computes an output for this target. All classifier outputs are collected in a score-matrix $R(x)$ in the form of

$$R(x) = \begin{pmatrix} - & r_{12} & \cdots & r_{1m} \\ r_{21} & - & \cdots & r_{2m} \\ \vdots & & & \vdots \\ r_{m1} & r_{m2} & \cdots & - \end{pmatrix}, \tag{4.3}$$

with $r_{ij} \in [0, 1]$. Matrix entries r_{ij} and r_{ji} are provided by the binary classifier that discerns between classes C_i and C_j. The former represents the confidence that element x belongs to class C_i and not C_j, while the latter is interpreted as the reverse. When the classifier does not provide both values, the missing one is defined by relation $r_{ji} = 1 - r_{ij}$.

To derive an exact prediction for x, the information in $R(x)$ needs to be aggregated. Various aggregation procedures have been proposed in the literature to derive the prediction $l(x)$ from score-matrix $R(x)$ [173]:

- **Voting, binary voting, Max-Wins rule (VOTE, [163])**: each binary classifier votes for (at most) one class, namely the one for which the computed confidence is strictly highest. The class with the highest number of votes is selected as prediction. In particular,

$$l(x) = \underset{i=1,\ldots,m}{\arg\max} \left(\sum_{1 \le j \ne i \le m} I(r_{ij} > r_{ji}) \right),$$

  with $I(\cdot)$ the standard indicator function which evaluates to one when its argument is true and to zero otherwise.
- **Weighted voting (WV, [235])**: instead of casting a definitive vote for a single class, each classifier provides weighted votes for both its classes using its derived confidence degrees. The class for which the sum of the confidence degrees in its favour is largest is predicted. In particular,

$$l(x) = \underset{i=1,\ldots,m}{\arg\max} \left(\sum_{1 \le j \ne i \le m} r_{ij} \right).$$

- **Pairwise coupling (PC, [208])**: based on the values in the score-matrix, this method approximates the class posterior probabilities by means of an optimization procedure and sets $l(x)$ to the class corresponding to the highest probability value.
- **Decision directed acyclic graph (DDAG, [347])**: this method constructs a rooted binary acyclic graph. Each node in the graph corresponds to a list of classes and a binary classifier. At each split, the classifier is evaluated on two of the classes in the list and only the predicted class is retained. Prediction $l(x)$ is set to the final class remaining in the list.
- **Learning valued preference for classifiers (LVPC, [233, 234])**: three fuzzy relations (strict preference, conflict, ignorance) are derived from the score-matrix. The class corresponding to the largest value of a linear combination of the degrees of preference, conflicts and ignorance is predicted.
- **Non-dominance criterion (ND, [144])**: this method interprets the score-matrix as a fuzzy preference relation and derives, for each class, the degree to which it is dominated by none of the other classes (degree of non-dominance). The class attaining the highest value for this measure is used for $l(x)$.

- **Binary tree of classifiers (BTC, [142])**: similar to the DDAG method, a tree graph is constructed, wherein each node corresponds to a list of classes and a binary classifier. The DDAG method trains the classifier on two classes and removes the class that is not predicted. BTC proceeds in a different way and allows the removal of multiple classes in each level. The final class remaining in the list is used for $l(x)$.
- **Nesting OVO (NEST, [293, 294])**: this procedure sets up an iterative, nested OVO evaluation. Elements in an unclassifiable region (that is, elements for which no single class is predicted strictly most) are processed with a second OVO decomposition focusing on this region. The process is repeated until all elements have been classified.
- **Probability estimates (PE, [449])**: the method performs the same actions as the PC procedure, but uses a different optimization objective to derive the posterior class probabilities.

Review paper [173] showed that the simple and intuitive WV strategy yields a competitive prediction performance compared to the more sophisticated aggregation alternatives and can be favoured in terms of its robustness as well.

4.2.2 OVO Decomposition and the Classifier Competence Issue

As stated above, each classifier within the OVO decomposition is trained on the elements of only two classes. As a consequence, the classifier trained on data from classes C_i and C_j can perform sub-optimally when classifying a test instance of class C_k ($k \neq i, j$), since it had no knowledge of or information on class C_k at training time. This problem is called the *classifier competence issue* [172]. When classifying a test instance x within an OVO setting, the scores or predictions of all binary classifiers are taken into account to derive the final class assignation. Classifiers that were not trained on the true class of x may not hold any relevant information for its classification and can consequently hinder the aggregation procedure.

This issue has been addressed by the development of dynamic aggregation procedures. Two recent proposals are the Dynamic OVO [175] and DRCW-OVO [179] methods. Both approaches consider the local neighbourhood of a test instance and modify the score-matrix in order to reduce the contribution of binary classifiers trained on irrelevant classes, that is, classes that do not appear in the evaluated neighbourhood. They differ from each other in the way the score-matrix is treated:

- **Dynamic OVO**: the $3m$ nearest neighbours of the target instance are computed, with m the number of classes. Only classifiers for which both training classes appear in this neighbourhood are considered. To achieve this effect, the score-matrix is filtered and the confidence values of irrelevant classifiers are put to zero to cancel out their contribution. The WV procedure is used to derive a prediction from the modified score-matrix.

- **DRCW-OVO**: rather than setting some confidence values to zero, the score-matrix is modified by multiplying every entry with a weight representing the relevance of the binary classifiers in the neighbourhood of the target instance. The weights are defined as the relative classifier competence and are based on the distance of the target to its k nearest neighbours of the classes under consideration. This method has very recently been modified to the DRCW-ASEG alternative in [495], incorporating a synthetic element generation step before the weight calculations.

Another example is the similarity based aggregation technique proposed in [178]. It is computationally less attractive than Dynamic OVO or DRCW-OVO, because it requires the internal optimization of $\frac{m(m-1)}{2}$ parameters by means of a genetic algorithm. We note that dynamic OVO aggregation procedures are different from dynamic classifier selection (e.g. [320]), where only one binary classifier is selected per test instance. They are related to the dynamic ensemble selection framework (e.g. [494]) however, where a subset of classifiers is selected.

A related recent proposal is the NMC method from [176], which uses an aggregation transformation strategy that aims to benefit from the non-competent classifiers instead of reducing their importance. In essence, this procedure applies a nearest neighbour classification on the score-matrices rather than directly aggregating the score-matrix of a test instance to a final prediction. The authors considered an internal static instance and classifier pruning step as well, but showed that their method yields competitive results even without this optimization. The selection step, based on a genetic algorithm, leads to a static rather than dynamic pruning, that is, the final prediction is always based on the same subset of base classifiers.

4.2.3 Dealing with Multi-class Imbalance

OVO decomposition is a general approach to transform a multi-class classification task to a set of binary problems. It can be applied in the context of multi-class imbalance to allow the application of the methods discussed in Sect. 4.1.2. Substantial research has been conducted in the area of binary imbalanced classification and the OVO decomposition permits one to take advantage of these efforts in the context of multi-class imbalance as well.

A direct application of this technique was evaluated in [148], where the OVO decomposition was combined with the data level approaches to binary class imbalance (resampling). The OVO procedure is used to decompose the multi-class problem into several binary problems, one for each pair of classes. This reduces the overall imbalance of the problems, as the IR in a binary problem is at most equal to the total IR of the dataset (as defined in (1.1)) but often noticeably smaller. For each sub-problem, a binary resampling technique is applied on the two classes under consideration. The study of [148] put forward the popular SMOTE method as a well-performing option. Binary classifiers are trained on the resampled sub-problems and are used to construct the score-matrix $R(x)$ when classifying an instance x.

Although the above setting can be considered a data level approach to multi-class imbalance, it does not correspond to the development of new resampling methods to directly deal with imbalanced datasets of more than two classes. Examples of the latter are encountered in the literature as well and can be considered as true multi-class data level approaches. The GlobalCS method [508] achieves its cost-sensitive effect by randomly replicating instances in each minority class until they are the same size as the majority class. The Static-SMOTE algorithm [149] is a modification of the SMOTE method that oversamples each class with the SMOTE technique to attain a better overall balance. Another SMOTE-inspired multi-class oversampling method is the synthetic minority oversampling method SMOM from [513]. The Mahalanobis distance oversampling procedure (MDO) proposed in [1] was inspired by the Mahalanobis distance [314]. This method generates artificial instances with the guarantee that their Mahalanobis distance to the class mean equals that of the seed element from which they were constructed. In [463], an extension of the MDO technique was proposed. This method is called AMDO and is based on a generalized singular value decomposition.

Multi-class algorithm level approaches to an imbalanced class distribution have been proposed as well. The AdaBoost.NC ensemble classifier [438] remains a state-of-the-art standard in the field of multi-class imbalanced classification. It is based on a binary AdaBoost ensemble incorporating negative correction learning [435]. In addition to using them to better recognize misclassified elements in later iterations, the instance weights are also used to enhance the diversity within the boosting ensemble. The multi-class extension requires the application of random oversampling to improve the recognition of minority class instances. Recently, the EFIS-MOEA method was proposed in [145]. This ensemble method combines feature and instance selection using multi-objective optimization. In the contributions of [203, 468], ensemble classifiers for multi-class imbalanced data were evaluated in conjunction with feature selection as well. The authors of [493] evaluated ensembles developed for binary imbalanced data in the multi-class setting using the OVO scheme.

4.3 FROVOCO: Novel Algorithm for Multi-class Imbalanced Problems

We propose an extension of the IFROWANN method ([361], Sect. 4.1.3) to the multi-class imbalance setting. Our method is called FROVOCO, short for Fuzzy Rough OVO Combination. It uses the OVO decomposition procedure (see Sect. 4.2.1) to transform the multi-class problem to a set of binary sub-problems to which IFROWANN can be applied. To improve the competitiveness of our proposal with the methods discussed in Sect. 4.2.3, we develop two new components:

- **IFROWANN-$\mathcal{W}_{\text{IR}}$**: as discussed in [361], the preferred combination of OWA weights used within the IFROWANN classifier depends on the imbalance present in the binary problem at hand. The binary datasets generated by the OVO scheme can have highly different IR values. We propose an adaptive version of the IFROWANN classifier (called IFROWANN-$\mathcal{W}_{\text{IR}}$) that selects the OWA weights based on the IR value. We introduce this method in Sect. 4.3.1.
- **WV-FROST aggregation**: we modify the WV aggregation scheme to include two global summary terms. These additional terms assess the global affinity of target instances with the decision classes and complement the local information derived from the OVO decomposition. The inclusion of the summary terms is a way to deal with the classifier competence issue (Sect. 4.2.2). Our new aggregation procedure is discussed in Sect. 4.3.2.

To summarize the above, Sect. 4.3.3 presents an overview of our complete FROVOCO method.

4.3.1 Binary Classifier Within OVO: IFROWANN-$\mathcal{W}_{\text{IR}}$

In our discussion on the IFROWANN classifier in Sect. 4.1.3, we listed the six OWA weight combinations proposed and evaluated in [361]. The authors showed that versions $\mathcal{W}_4$ and $\mathcal{W}_6$ yield the best overall classification results. In this work, we refer to these settings as $\mathcal{W}_e$ and $\mathcal{W}_\gamma$ respectively, such that their name allows to more easily recollect their definition. The experimental evaluation of [361] indicated that $\mathcal{W}_e$ performs best for mildly imbalanced datasets (IR up to nine), while the $\mathcal{W}_\gamma$ setting is preferred when the imbalance exceeds this level. As noted in Sect. 4.1.1, an IR value of nine is often used as a threshold above which the dataset is considered highly imbalanced.

Our FROVOCO method uses the OVO decomposition procedure, which means that it generates a binary classification problem for each pair of classes in the multi-class dataset. Some classes may have similar sizes (implying a low IR value), while the sizes of others may be highly different (high IR). Consequently, keeping the conclusions of [361] in mind, it is not prudent to fix the IFROWANN weight configuration beforehand. We therefore propose a new adaptive weight setting $\mathcal{W}_{\text{IR}}$. When the IR of the binary problem under consideration is at most nine, $\mathcal{W}_{\text{IR}}$ coincides with $\mathcal{W}_e$. When the IR exceeds this threshold, $\mathcal{W}_{\text{IR}}$ coincides with $\mathcal{W}_\gamma$. The binary classifier using this setting is called IFROWANN-$\mathcal{W}_{\text{IR}}$. Apart from its adaptive choice of OWA weights for the two classes, it coincides with the IFROWANN method discussed in Sect. 4.1.3.

To explicitly motivate the choice of threshold on the IR, we recall the experimental results of [361] in Fig. 4.2. The authors used a collection of 102 binary datasets with IR values ranging from 1.82 to 129.44. In Fig. 4.2, we plot the difference in obtained AUC values of IFROWANN classifiers using $\mathcal{W}_\gamma$ and $\mathcal{W}_e$. Positive values indicate that the former outperforms the latter. Figure 4.2a contains the results of all

102 datasets. The datasets are ordered according to increasing IR. The only definite conclusion that can be drawn from this plot is that for a very highly imbalanced binary dataset (IR above 65), combination $\mathcal{W}_\gamma$ clearly leads to the best classification results. In Fig. 4.2b, we consider the datasets with an IR of at most 20. The plot indicates that the benefits of $\mathcal{W}_e$ are only present for mildly imbalanced datasets. Figure 4.2c, d are based on the 88 datasets with an IR at most 65. We include these figures to decide on the threshold above which $\mathcal{W}_\gamma$ can be preferred. They were constructed by averaging over consecutive points in order to obtain a smoother plot. In Fig. 4.2c, each point was obtained as an average across four observations. For example, the leftmost point is computed as the mean value of the four least imbalanced datasets in the study. In Fig. 4.2d, the same procedure was followed by taking averages across eight observations. In this way, an even smoother figure is obtained. We observe that our threshold of nine certainly seems appropriate. When the IR of a dataset exceeds this value, the use of $\mathcal{W}_\gamma$ results in a better classification performance of IFROWANN than $\mathcal{W}_e$. In the other case, the latter is preferred. Our choice of threshold follows from a tradition in class imbalanced learning and the empirical validation conducted in [361] and visually recalled in Fig. 4.2.

Our IFROWANN-$\mathcal{W}_{\text{IR}}$ classifier is used within the OVO decomposition scheme. For each pair of classes, it treats the smallest class as positive and the largest class as negative. As described in Sect. 4.1.3, whether a class is considered positive or negative affects the choice of weights used in its lower approximation calculations.

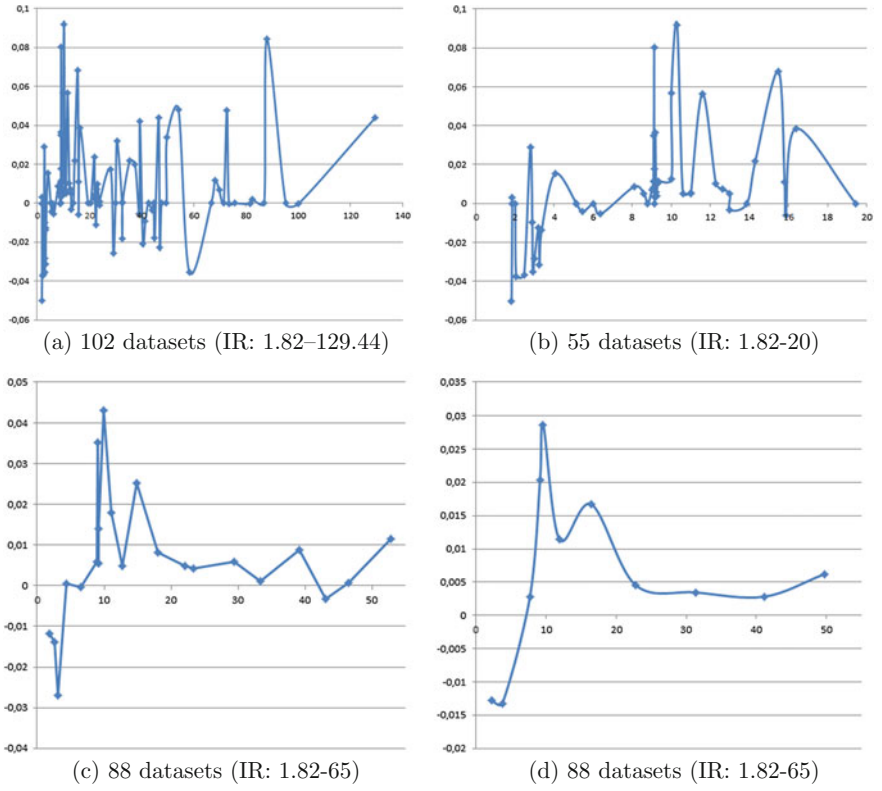
Finally, in order to adequately construct score-matrix (4.3), the classifier should output class confidence values rather than a strict prediction. When classifying x based on classes C_i and C_j, our IFROWANN-$\mathcal{W}_{\text{IR}}$ within OVO computes the score in favour of C_i as $\frac{\underline{C_i}(x)}{\underline{C_i}(x)+\underline{C_j}(x)}$ and that in favour of C_j as $\frac{\underline{C_j}(x)}{\underline{C_i}(x)+\underline{C_j}(x)}$.

4.3.2 New OVO Aggregation Scheme: WV-FROST

Based on the WV aggregation scheme (Sect. 4.2.1), we present a novel OVO aggregation scheme. Our proposal is called WV-FROST, which stands for Weighted Voting with Fuzzy ROugh Summary Terms. As a traditional OVO aggregator, WV captures the local information derived from the binary classification problems. WV-FROST complements this aspect with a global evaluation in the form of two summary terms. The inclusion of the global summary addresses the classifier competence issue (Sect. 4.2.2) by counteracting the information loss induced by the binary decomposition step.

Let x be the instance to classify and $R(x)$ the score-matrix constructed with the OVO scheme. As recalled in Sect. 4.2.1, the WV method assigns x to the class with the highest combined weighted vote in its favour. Its decision process relies on a vector V_x with

$$V_x(C_i) = \frac{1}{m} \sum_{1 \leq j \leq m} r_{ij},$$



(a) 102 datasets (IR: 1.82–129.44) (b) 55 datasets (IR: 1.82-20)

(c) 88 datasets (IR: 1.82-65) (d) 88 datasets (IR: 1.82-65)

Fig. 4.2 Motivation for the definition of $\mathcal{W}_{\text{IR}}$. The horizontal axis represents the IR of the dataset and the vertical axis the difference in the obtained AUC values between $\mathcal{W}_\gamma$ and $\mathcal{W}_e$, based on the results on the 102 binary datasets in [361]. Positive values indicate that $\mathcal{W}_\gamma$ performs better than $\mathcal{W}_e$. Lines between points where drawn for the sake of visual clarity

where the sum of the confidence degrees in favour of class C_i is divided by m for scaling purposes. WV assigns x to the class corresponding to the maximal value in V_x. Our proposed WV-FROST method modifies the V_x vector prior to the class prediction step. Two additional measures are added to each value $V_x(C_i)$ $(i = 1, \ldots, m)$:

- **Positive affinity term** $\boldsymbol{mem(x, C_i)}$: the membership degree of x to class C_i based on the full training set. The membership degree is set to the average of the membership degrees to the fuzzy rough lower and upper approximations of C_i.
- **Negative affinity term** $\boldsymbol{mse_n(x, C_i)}$: a signature vector of expected membership degrees of instances of class C_i to all classes can be constructed based on the $mem(y, \cdot)$ for all training instances $y \in C_i$. A similar vector can be constructed for x, namely consisting of its $mem(x, \cdot)$ values for all classes. The distance of this vector to the signature of C_i is penalized.

We discuss the two summary terms in detail in the following paragraphs.

Positive affinity For class C_i, the summary term $mem(x, C_i)$ represents the globally evaluated affinity of instance x with class C_i. This measure is defined as the average membership degree of x to the fuzzy rough lower and upper approximation of C_i, namely

$$mem(x, C_i) = \frac{\underline{C_i}(x) + \overline{C_i}(x)}{2}. \tag{4.4}$$

The values are directly derived from the full dataset and not from the binary subproblems, such that $mem(x, C_i)$ corresponds to a global evaluation of the affinity of x with C_i. It should be clear that definition (4.4) relates to the decision procedure used by the FRNN classifier ([242], Sect. 4.1.3). We use the OWA based fuzzy rough set model to compute the $\underline{C_i}(x)$ and $\overline{C_i}(x)$ values with weight vectors related to our adaptive weight setting $\mathcal{W}_{\text{IR}}$. The sizes of the sets to aggregate in $\underline{C_i}(x)$ and $\overline{C_i}(x)$ are $|co(C_i)|$ and $|C_i|$ respectively, with $co(\cdot)$ the set complement function. According to $\mathcal{W}_{\text{IR}}$, when the IR between C_i and its complement does not exceed nine, exponential OWA weights are used in both the lower and upper approximation calculations (combination $\mathcal{W}_e$). In the other case, when combination $\mathcal{W}_\gamma$ is followed, the shortest weight vector uses the exponential definition and the longest weight vector is constructed with (4.2), replacing $|P|$ and $|N|$ by $\min(|C_i|, |co(C_i)|)$ and $\max(|C_i|, |co(C_i)|)$ respectively in its definition. Instead of simply adding the $mem(x, C_i)$ values to the V_x vector, we replace each position $V_x(C_i)$ by $\frac{V_x(C_i)+mem(x,C_i)}{2}$. In this way, local evaluation $V_x(C_i)$ is replaced by the average of the local and global measures.

Negative affinity The second global summary term measures the instance-to-class affinity at a higher level and evaluates how strongly the $mem(x, \cdot)$ values resemble the expected values for instances belonging to particular classes. In order to do so, a signature vector S_{C_i} $(i = 1, \ldots, m)$ consisting of these expected values is constructed for each class. The class signature corresponds to a decision template [271]. Vector S_{C_i} has size m and position $S_{C_i}(C_j)$ corresponds to the average membership value $mem(y, C_j)$ of instances $y \in C_i$. In particular

$$\begin{aligned} S_{C_i} &= \left\langle S_{C_i}(C_1), S_{C_i}(C_2), \ldots, S_{C_i}(C_m) \right\rangle \\ &= \left\langle \frac{1}{|C_i|} \sum_{y \in C_i} mem(y, C_1), \frac{1}{|C_i|} \sum_{y \in C_i} mem(y, C_2), \ldots, \frac{1}{|C_i|} \sum_{y \in C_i} mem(y, C_m) \right\rangle. \end{aligned}$$

The $mem(x, \cdot)$ values can be grouped in a similar vector S_x and compared to the S_{C_i} vectors. The distance between S_x and the class signatures is measured by the mean squared error as

$$mse(x, C_i) = \frac{1}{m} \sum_{i=1}^{m} (mem(x, C_j) - S_{C_i}(C_j))^2$$

and expresses to what extent x is dissimilar to the training instances of class C_i. The dissimilarity property implies that a negative class affinity is evaluated by this measure. A high $mse(x, C_i)$ value indicates a large distance between S_x and the expected class membership values for instances in class C_i. The inclusion of the second affinity term is motivated by the following example. Consider a dataset with three classes, of which classes C_1 and C_2 have a high overlap in feature space. Since the definitions of the fuzzy rough lower and upper approximations strongly rely on instance similarity values, the membership values $mem(x, C_1)$ and $mem(x, C_2)$ can be expected to be close together. As a consequence, $mem(x, C_1)$ and $mem(x, C_2)$ may not suffice to decide between classes. The signature vectors S_{C_1} and S_{C_2} contain the expected membership values of instances of classes C_1 and C_2 to all classes and comparing $mem(x, C_3)$ to $S_{C_1}(C_3)$ and $S_{C_2}(C_3)$ can provide a vital clue in the class decision process. Before including the $mse(x, \cdot)$ values in the V_x vector, we scale them by dividing them by their total sum, defining

$$mse_n(x, C_i) = \frac{mse(x, C_i)}{\sum_{j=1}^{m} mse(x, C_j)}.$$

The $mse_n(x, \cdot)$ values are used as summary terms. Since they measure a negative class affinity, we subtract them from the values in V_x with weight $\frac{1}{m}$. This factor is inversely proportional to the number of classes in the dataset. We can expect the information measured by the $mse_n(x, \cdot)$ values to be less reliable when the number of classes increases. For larger values of m, the size of S_C increases and the constituent membership degrees become more similar. As a result, the class distinction power of the mean squared error is reduced.

Summary WV-FROST modifies the V_x vector constructed by WV by including two summary terms. For each class C_i, value $V_x(C_i)$ is replaced by the affinity based alternative

$$AV_x(C_i) = \frac{V_x(C) + mem(x, C_i)}{2} - \frac{1}{m} mse_n(x, C_i), \tag{4.5}$$

representing the aggregated score for class C_i. Figure 4.3 visually presents the internal construction of the $AV_x(\cdot)$ values by WV-FROST. As part of our experimental study, we show that the inclusion of both summary terms leads to superior classification results. The final predicted class label for test instance x is obtained as the class corresponding to the maximum $AV_x(\cdot)$ value, that is,

$$l(x) = \underset{i=1,\ldots,m}{\arg\max}(AV_x(C_i)).$$

The inclusion of the global summary terms $mem(x, \cdot)$ and $mse_n(x, \cdot)$ results in a dynamic aggregation procedure combating the classifier competence problem. WV-FROST differs from the Dynamic OVO and DRCW-OVO methods described in Sect. 4.2.2, because it leaves the score-matrix unchanged. We do not modify the matrix, but instead change the aggregated values by adding more information to them.

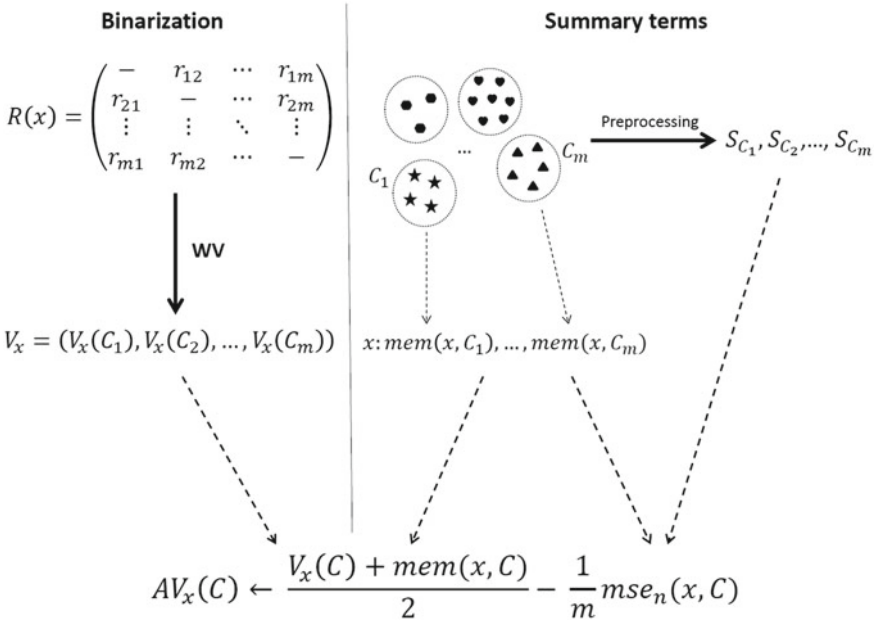


Fig. 4.3 Computation of the class scores $AV_x(\cdot)$ by WV-FROST

4.3.3 Overview of the FROVOCO Proposal

Our complete proposal for multi-class imbalanced classification is called FROVOCO, which stands for Fuzzy Rough OVO COmbination. It uses the IFROWANN-$\mathcal{W}_{\text{IR}}$ classifier within the OVO scheme combined with our WV-FROST aggregation. Fuzzy rough set theory is used in both stages of the proposal, the binary classification and prediction aggregation. The combined benefits are clearly shown in our experimental analysis conducted in the remainder of this chapter. Figure 4.4 presents an overview of our complete proposal. To classify a test instance x, the following steps are performed:

1. In the OVO decomposition phase, element x is sent to all IFROWANN-$\mathcal{W}_{\text{IR}}$ classifiers, each using a pair of classes as its training set.
2. The IFROWANN-$\mathcal{W}_{\text{IR}}$ methods generate class confidence scores.
3. The obtained scores are grouped in a score-matrix of the form (4.3).
4. WV-FROST is applied to aggregate the score-matrix to the AV_x vector using expression (4.5). This procedure is presented in Fig. 4.3. Instance x is assigned to the class for which the $AV_x(\cdot)$ value is largest.

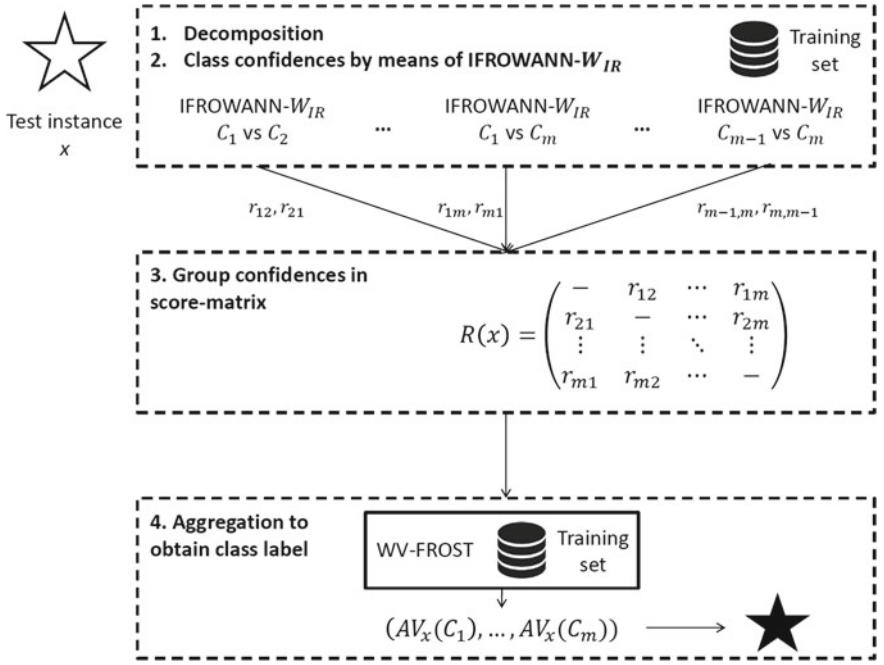


Fig. 4.4 Overview of the actions of our full FROVOCO method in the classification of instance x

4.4 Experimental Study

In this section, we experimentally validate our FROVOCO proposal. We first present our experimental set-up in Sect. 4.4.1. In Sect. 4.4.2, we evaluate our proposed adaptive weight combination $\mathcal{W}_{\text{IR}}$. To further motivate its definition, Sect. 4.4.3 compares the WV-FROST component to WV and other partially constructed models. We compare WV-FROST to other dynamic OVO aggregation procedures in Sect. 4.4.4. Finally, our full FROVOCO method as proposed in Sect. 4.3 is compared to the state-of-the-art in multi-class imbalanced classification in Sect. 4.4.4.

4.4.1 Experimental Set-Up

We use a collection of 18 datasets in the experimental evaluation conducted in this chapter. The datasets and their properties are listed in Table 4.1. We list the number of features (numeric and nominal) and classes. As an indication of the imbalance present in the dataset, we list the class distribution as well as the minimum, average and maximum IR values between pairs of classes. In all datasets, the minimum IR is low, which implies that there are at least two classes with relatively similar sizes.

Table 4.1 The 18 multi-class imbalanced datasets used in the experimental study of Chap. 4. The number of features is divided between numeric and nominal ones, e.g. the automobile dataset has 15 numeric features and 10 nominal features

Dataset	# inst	# feat	m	Min IR	Av. IR	Max IR	Distribution
automobile	150	25 (15/10)	6	1.04	4.90	16.00	3/20/48/46/29/13
balance	625	4 (4/0)	3	1.00	4.25	5.88	288/49/288
cleveland	297	13 (13/0)	5	1.03	3.87	12.62	164/55/36/35/13
contraceptive	1473	9 (9/0)	3	1.23	1.55	1.89	629/333/511
dermatology	358	34 (34/0)	6	1.00	2.17	5.55	111/60/71/48/48/20
ecoli	336	7 (7/0)	8	1.00	15.27	71.50	143/77/2/2/ 35/20/5/52
glass	214	9 (9/0)	6	1.09	3.60	8.44	70/76/17/13/9/29
led7digit	500	7 (7/0)	10	1.00	1.16	1.54	45/37/51/57/52/ 52/47/57/53/49
lymphography	148	18 (3/15)	4	1.33	18.30	40.50	2/81/61/4
newthyroid	215	5 (5/0)	3	1.17	3.48	5.00	150/35/30
pageblocks	5472	10 (10/0)	5	1.32	31.65	175.46	4913/329/28/87/115
satimage	6435	36 (36/0)	6	1.01	1.73	2.45	1533/703/1358/ 626/707/1508
shuttle	58000	9 (9/0)	7	1.30	561.92	558.60	45586/49/171/8903/ 3267/10/13
thyroid	7200	21 (21/0)	3	2.22	20.16	40.16	166/368/6666
wine	178	13 (13/0)	3	1.20	1.30	1.48	59/71/48
winequality-red	1599	11 (11/0)	6	1.07	18.83	68.10	10/53/681/638/ 199/18
winequality-white	4898	11 (11/0)	7	1.07	61.08	439.60	20/163/1457/2198/ 880/175/5
yeast	1484	8 (8/0)	10	1.08	11.65	92.60	244/429/463/44/51/ 163/35/30/20/5

The maximum IR expresses the largest imbalance encountered between any class pair in the dataset. This value varies greatly across the included datasets and is often very high. For most datasets, the average IR of the class pairs is relatively moderate, although a few extreme values are present.

In our evaluation, we use the ten-fold DOB-SCV validation scheme, as recommended for imbalanced data classification in [307] and recalled in Sect. 2.3.2. The prediction performance of the included methods is evaluated by means of the balanced accuracy and MAUC measures (see Sect. 2.3.1.2). We include both measures to capture two different aspects of the prediction behaviour of the classifiers. The balanced accuracy considers the actual output of a method and measures how well it recognizes the different classes. The MAUC, on the other hand, expresses the ability of an algorithm to separate pairs of classes (see e.g. [438]).

As part of our experimental validation, we compare our FROVOCO method to the state-of-the-art in multi-class imbalanced classification. We include the following methods discussed in Sect. 4.2.3:

- **OVO combined with binary resampling** [148]: we use the SMOTE method as resampling method in combination with the C4.5 classifier with the same parameter settings as in [148]. The use of decision tree learners like C4.5 in ensembles has been highlighted in [365]. The score-matrix is aggregated using the WV scheme.
- **GlobalCS** [508]: we combine this cost-sensitive approach with the C4.5 classifier.
- **Static-SMOTE** [149]: we combine this oversampling method with the C4.5 classifier.
- **MDO oversampling** [1]: we combine this method with the C4.5 classifier. We note that there is a small error in the pseudo-code method description of MDO in [1], which makes the implementation presented therein invalid. We have fixed this by slightly modifying Algorithm 3 from [1]. Instead of choosing the value r from the interval $[-\sqrt{AlphaV(j)}, \sqrt{AlphaV(j)}]$, we divide both boundaries by $|\mathcal{A}| - 1$, where $\mathcal{A}$ is the feature set. This fix is required to guarantee that a solution can be found in line 14. Leaving the algorithm as it was presented in [1] results in failures.
- **AdaBoost.NC** [438]: the penalty strength λ is set to two, as done in e.g. [148, 438]. The number of classifiers in the ensemble was set to ten, which is a lower value than the one used in the referenced studies. In a preliminary evaluation, we observed that this value provides better average results on our selected datasets. It has been used in ensemble classifiers for imbalanced data in earlier studies as well (e.g. [306]).
- **EFIS-MOEA** [145]: we use the implementation as provided by the authors in their online repository as well as the parameter settings recommended there.

4.4.2 *Evaluation of IFROWANN-$\mathcal{W}_{\text{IR}}$*

In this section, we assess the performance of the IFROWANN method within an OVO set-up using the nine traditional OVO aggregation schemes recalled in Sect. 4.2.1. The IFROWANN classifier is evaluated with three different weight combinations: the two original settings $\mathcal{W}_e$ and $\mathcal{W}_\gamma$ proposed in [361] and our adaptive version $\mathcal{W}_{\text{IR}}$.

We present the average accuracy and MAUC results of this evaluation in Table 4.2. These values were taken as averages and standard deviations across the 18 datasets listed in Table 4.1. The benefit of the adaptive combination $\mathcal{W}_{\text{IR}}$ over $\mathcal{W}_e$ and $\mathcal{W}_\gamma$ is clear. This is particularly reflected in the balanced accuracy measure, where substantial differences can be observed. For each aggregation scheme, the use of $\mathcal{W}_{\text{IR}}$ leads to the highest accuracy. We also note that the results of $\mathcal{W}_e$ are better than those of $\mathcal{W}_\gamma$. This can be explained based on the dataset description in Table 4.1 and the conclusions drawn from Fig. 4.2. The pairwise IR between classes is often less than nine, a situation in which $\mathcal{W}_e$ yields better results than $\mathcal{W}_\gamma$. Considering the MAUC evaluation, smaller performance differences between $\mathcal{W}_e$, $\mathcal{W}_\gamma$ and $\mathcal{W}_{\text{IR}}$ are observed. For five out of nine aggregation methods, $\mathcal{W}_{\text{IR}}$ has the best performance. In the four cases where it does not, the differences with the best performing scheme are small. Setting $\mathcal{W}_e$ mostly outperforms $\mathcal{W}_\gamma$ with respect to the MAUC as well.

In summary, when fixing the IFROWANN weight combination regardless of the IR, $\mathcal{W}_e$ yields better results than $\mathcal{W}_\gamma$ does. Our adaptive setting $\mathcal{W}_{\text{IR}}$ further improves the performance. The largest improvement is observed for the balanced accuracy measure, which implies that a high rate of correct classifications requires the use of $\mathcal{W}_{\text{IR}}$. The power to separate pairs of classes (as evaluated by the MAUC measure) is more or less comparable for both $\mathcal{W}_e$ and $\mathcal{W}_{\text{IR}}$. Having selected the $\mathcal{W}_{\text{IR}}$ combination and taking both evaluation measures into account, we can select the WV procedure as favoured aggregation scheme. It attains the highest balanced accuracy result and among the highest MAUC values. Furthermore, its robustness has been demonstrated in [173, 235]. In the following section, we further improve the performance of IFROWANN-$\mathcal{W}_{\text{IR}}$ within OVO by replacing the WV aggregation by our WV-FROST proposal.

4.4.3 Evaluation of IFROWANN-WV-FROST

In this section, we conduct an internal validation of our WV-FROST proposal and motivate our inclusion of the two global summary terms. In order to do so, we compare our aggregation scheme to partially constructed versions. We assess whether WV-FROST improves the performance of WV and whether partially constructed models would already yield similar (or better) results. Table 4.3 presents the results of partially constructed versions of WV-FROST. The classifier within the OVO decomposition is fixed to our IFROWANN-$\mathcal{W}_{\text{IR}}$ method. Table 4.3 includes the results of the following models:

- IFROWANN-$\mathcal{W}_{\text{IR}}$-WV: the best performing combination from Sect. 4.4.2. It uses the IFROWANN-$\mathcal{W}_{\text{IR}}$ algorithm in the OVO decomposition with WV aggregation.
- *mem*: this method does not perform a decomposition into binary problems. For a test instance x, the score of class C is set to value $mem(x, C)$. The class with the highest score is predicted.

Table 4.2 Results of the integration of IFROWANN in the OVO setting with the traditional aggregation methods. For each method and each evaluation measure, we print the result of the best performing weight combination in boldface

Balacc			
Method	$\mathcal{W}_e$	$\mathcal{W}_\gamma$	$\mathcal{W}_{\text{IR}}$
VOTE	69.4460±19.6554	61.6819±20.7889	**70.4035±20.5736**
WV	69.4460±19.6554	63.1538±21.6848	**71.4921±19.5685**
PC	69.4959±19.6182	62.6440±21.0826	**71.3500±19.1160**
DDAG	69.9674±19.6337	59.4896±21.0593	**71.1942±19.9944**
LVPC	58.8251±20.3807	61.2524±20.2090	**63.2167±19.5304**
ND	69.5269±19.5921	58.5540±23.3192	**70.2541±21.2630**
BTC	69.4497±19.7924	59.0936±21.4492	**70.5591±20.5641**
NEST	69.5686±19.6920	56.2790±23.1138	**70.0260±21.2530**
PE	69.4785±19.7172	62.1607±21.3814	**71.1413±19.6598**
MAUC			
Method	$\mathcal{W}_e$	$\mathcal{W}_\gamma$	$\mathcal{W}_{\text{IR}}$
VOTE	0.8566±0.1227	0.8208±0.1379	**0.8613±0.1204**
WV	**0.8921±0.1120**	0.8910±0.1025	0.8895±0.1120
PC	**0.8958±0.1062**	0.8935±0.1058	0.8939±0.1067
DDAG	0.8070±0.1243	0.7366±0.1384	**0.8143±0.1274**
LVPC	**0.8932±0.1110**	0.8919±0.1043	0.8910±0.1107
ND	0.8779±0.1065	0.8457±0.1229	**0.8794±0.1092**
BTC	0.8035±0.1259	0.7341±0.1413	**0.8105±0.1304**
NEST	0.8572±0.1228	0.8208±0.1380	**0.8613±0.1204**
PE	**0.8952±0.1065**	0.8875±0.1089	0.8927±0.1077

- mem-mse_n: this method does not perform a decomposition into binary problems. For a test instance x, the score of class C is set to value $mem(x, C) - \frac{1}{m} mse_n(x, C)$. The class with the highest score is predicted.
- IFROWANN-$\mathcal{W}_{\text{IR}}$-WV-mem: the IFROWANN-$\mathcal{W}_{\text{IR}}$ method is used in an OVO decomposition. The WV-mem aggregation scheme is similar to WV-FROST, but includes only one global summary term. The affinity based class scores $AV_x(\cdot)$ are set to $AV_x(C) = \frac{V_x(C)+mem(x,C)}{2}$.
- IFROWANN-$\mathcal{W}_{\text{IR}}$-WV-mse_n: the IFROWANN-$\mathcal{W}_{\text{IR}}$ method is used in an OVO decomposition. The WV-mse_n aggregation scheme is similar to WV-FROST, but includes only one global summary term. The affinity based class scores $AV_x(\cdot)$ are set to $AV_x(C) = V_x(C) - \frac{1}{m} mse_n(x, C)$.
- IFROWANN-$\mathcal{W}_{\text{IR}}$-WV-FROST: our complete proposal, the FROVOCO method.

The second and third methods have been included to verify whether our global summary terms are strong enough on their own or whether the binarization is truly

Table 4.3 Results of IFROWANN-$\mathcal{W}_{\text{IR}}$-WV-FROST and partially constructed versions. The best results are printed in boldface

Method	Balacc	MAUC
IFROWANN-$\mathcal{W}_{\text{IR}}$-WV	71.4921±19.5685	0.8895±0.1120
mem	67.6477±18.8233	0.8810±0.1069
mem-mse$_n$	69.2093±18.3121	0.8958±0.1022
IFROWANN-$\mathcal{W}_{\text{IR}}$-WV-*mem*	71.5351±19.3465	0.8946±0.1065
IFROWANN-$\mathcal{W}_{\text{IR}}$-WV-*mse*$_n$	71.8868±19.2429	0.8984±0.0987
IFROWANN-$\mathcal{W}_{\text{IR}}$-WV-FROST	**72.6327±19.3379**	**0.9018±0.0982**

required to obtain correct predictions. Both models correspond to generalizations of the original binary IFROWANN method (Sect. 4.1.3) to multi-class classification.

Table 4.3 clearly shows the superiority of our full proposal over all partially constructed models. Considering the results in more detail, we can observe the following:

- IFROWANN-$\mathcal{W}_{\text{IR}}$-WV-FROST outperforms IFROWANN-$\mathcal{W}_{\text{IR}}$-WV on 12 out of the 18 datasets in terms of the balanced accuracy and on 11 out of 18 based on the MAUC. This proves the worth of the inclusion of the summary terms in the aggregation process.
- Placing the results of models *mem* and IFROWANN-$\mathcal{W}_{\text{IR}}$-WV-*mem* next to each other, which differ in their use of the binarization or not, the benefit of the OVO decomposition is clear, in particular in the evaluation by the balanced accuracy. The relatively high MAUC result of the *mem* model indicates that the fuzzy rough membership degrees form an adequate tool to separate between pairs of classes. However, this does not necessarily imply correct classification results, as only pairwise comparisons between classes are used in the MAUC calculation. The classification behaviour of *mem* is clearly inferior, as reflected in its lower balanced accuracy value. The *mem* method corresponds to the most straightforward extension of IFROWANN to multi-class classification without applying any binarization. By also incorporating the pairwise comparison between classes, the IFROWANN-$\mathcal{W}_{\text{IR}}$-WV-*mem* can make more accurate predictions.
- Comparing *mem* to *mem-mse*$_n$ on the one hand and IFROWANN-$\mathcal{W}_{\text{IR}}$-WV-*mem* to IFROWANN-$\mathcal{W}_{\text{IR}}$-WV-FROST on the other, the improvement after the inclusion of the *mse*$_n$ term is clear.
- The performance difference between IFROWANN-$\mathcal{W}_{\text{IR}}$-WV-FROST and IFROWANN-$\mathcal{W}_{\text{IR}}$-WV-*mse*$_n$ shows that it is not sufficient to solely include the *mse*$_n$ measure and that both fuzzy rough summary terms carry complementary information needed to improve the baseline performance of IFROWANN-$\mathcal{W}_{\text{IR}}$-WV.
- In a statistical comparison between IFROWANN-$\mathcal{W}_{\text{IR}}$-WV-FROST and IFROWANN-$\mathcal{W}_{\text{IR}}$-WV by means of the Wilcoxon test, the p-values for the balanced accuracy and MAUC evaluations were 0.16736 ($R^+ = 118.0$, $R^- = 53.0$) and 0.08866 ($R^+ = 113.0$, $R^- = 40.0$) respectively.

Table 4.4 Mean balanced accuracy and MAUC values for different IR thresholds and γ values used within the $\mathcal{W}_{\text{IR}}$ component of WV-FROST. When varying the threshold, we set $\gamma = 0.1$. When different γ values are compared, the threshold is fixed to nine. For each column, the highest value is printed in boldface

IR	Balacc	MAUC	γ	Balacc	MAUC
5	72.4545±19.4768	0.9022±0.0979	0.0	72.6250±19.1890	**0.9050±0.0972**
6	**72.9003±19.3202**	0.9014±0.0979	0.1	**72.6327±19.3379**	0.9018±0.0982
7	72.8630±19.3377	0.9016±0.0981	0.2	72.1390±19.9091	0.9006±0.0990
8	72.7518±19.3471	0.9015±0.0983	0.3	71.7953±19.8429	0.8995±0.0997
9	72.6327±19.3379	0.9018±0.0982	0.4	71.6738±19.7787	0.8989±0.1000
10	72.5917±19.4183	0.9018±0.0982	0.5	71.3594±19.7781	0.8986±0.1001
11	72.5870±19.4929	0.9018±0.0981	0.6	71.2676±19.8256	0.8983±0.1002
12	72.6394±19.5677	0.9020±0.0982	0.7	70.9449±19.8065	0.8980±0.1003
13	72.7787±19.4216	0.9038±0.0974	0.8	70.6060±19.9011	0.8977±0.1005
14	72.6289±19.4657	0.9039±0.0974	0.9	70.0413±20.1123	0.8973±0.1008
15	72.7019±19.5113	**0.9041±0.0975**	1.0	69.8500±19.7312	0.8971±0.1010

The above results allow us to conclude the validity of our proposal. Aside from the comparisons to the partial models, we also study the effect of varying two internal parameters of the $\mathcal{W}_{\text{IR}}$ weight combination within FROVOCO. The $\mathcal{W}_{\text{IR}}$ setting is used by the IFROWANN classifiers as well as in the calculation of the fuzzy rough membership values (4.4) in the two summary terms. The internal parameters are the IR threshold above which the weight combination changes from $\mathcal{W}_e$ to $\mathcal{W}_\gamma$ and the γ value used internally in $\mathcal{W}_\gamma$. We followed the conclusions and guidelines of [361] and have set the IR threshold to 9 and γ to 0.1. However, we need to verify whether a different choice would largely impact the classification results of our proposal. We do so in Table 4.4. We vary the IR threshold between 5 and 15 and both the balanced accuracy and MAUC results indicate the small sensitivity of our proposal to these values. This holds for both the mean values as well as the results per dataset. A different behaviour is observed when varying the γ parameter. Both evaluation metrics indicate the preference for a low γ value. A similar evaluation was conducted in the experimental study of [361], with the same conclusion. Based on these results, we decide to retain the parameter settings as introduced in Sect. 4.3.

4.4.4 WV-FROST Versus Other Dynamic Approaches

Our next step is to compare the WV-FROST step to existing dynamic OVO aggregation approaches. We select the Dynamic OVO [175], DRCW-OVO [179] and NMC [176] methods discussed in Sect. 4.2.3. With regard to the latter, we select the version without the static instance and classifier selection step, such that this method

Table 4.5 Results of FROVOCO and the combination of IFROWANN-$\mathcal{W}_{\text{IR}}$ with three other dynamic aggregation methods

Method	Balacc	MAUC
IFROWANN-$\mathcal{W}_{\text{IR}}$+Dynamic OVO	71.7930±19.8270	0.7894±0.1151
IFROWANN-$\mathcal{W}_{\text{IR}}$+DRCW-OVO	70.8782±20.1452	0.8916±0.1097
IFROWANN-$\mathcal{W}_{\text{IR}}$+NMC	68.2483±21.4451	0.8907±0.1087
FROVOCO	**72.6327±19.3379**	**0.9018±0.0982**

Table 4.6 Pairwise statistical comparisons by means of the Wilcoxon test, accompanying the results of Table 4.5. Lines preceded by 'B' correspond to the evaluation by the balanced accuracy, while those starting with 'M' are related to the evaluation by MAUC. The second column lists the number of wins (W) and losses (L) corresponding to WV-FROST

Comparison	W/L	R^+	R^-	p
B: WV-FROST versus Dynamic OVO	9/9	100.0	71.0	0.52773
B: WV-FROST versus DRCW-OVO	12/6	129.0	42.0	0.05994
B: WV-FROST versus NMC	11/7	135.0	36.0	0.03036
M: WV-FROST versus Dynamic OVO	18/0	171.0	0.0	7.63E-6
M: WV-FROST versus DRCW-OVO	11/6	100.0	53.0	0.26595
M: WV-FROST versus NMC	11/6	111.0	42.0	0.10888

performs a nearest neighbour comparison of the score-matrices. Each aggregation scheme is combined with the IFROWANN-$\mathcal{W}_{\text{IR}}$ classifier within an OVO set-up. These combinations are compared to our full FROVOCO method, which uses our WV-FROST aggregation.

The mean results and standard deviations of this evaluation are presented in Table 4.5. Table 4.6 contains the accompanying statistical comparison of FROVOCO to the other methods by means of the Wilcoxon test. We observe that our method outperforms all others for both the balanced accuracy and MAUC. Furthermore, the computed rank sums of the Wilcoxon tests are always in favour of our proposal as well and it generally attains more wins than its competitors. WV-FROST significantly outperforms Dynamic OVO at the 5% significance level for the MAUC evaluation. This is due to the strict exclusion of some candidate classes by Dynamic OVO. The corresponding class probabilities are set to zero, a drastic action which results in lower MAUC values. The preference of WV-FROST over DRCW-OVO is clearest for the balanced accuracy. With respect to NMC, WV-FROST significantly outperforms this method for the balanced accuracy. From the results in Tables 4.5 and 4.6, we can conclude that IFROWANN-$\mathcal{W}_{\text{IR}}$ within the OVO decomposition scheme interacts better with WV-FROST than with the existing dynamic aggregation methods proposed in [175, 176, 179]. Since our binary classifier and aggregation scheme are both based on fuzzy rough set theory, their superior synergy is not an unexpected outcome. In fact, it motivated the use of fuzzy rough sets in our proposal of WV-FROST.

4.4.5 FROVOCO Versus State-of-the-Art Classifiers

As a final step in the experimental validation of FROVOCO, we compare our method to the state-of-the-art in multi-class imbalanced classification. This comparison includes the methods as listed in Sect. 4.4.1: OVO combined with SMOTE resampling and the WV aggregation scheme (OVO-SMT), GlobalCS combined with C4.5 (GlobalCS), Static-SMOTE combined with C4.5 (St-SMT), MDO oversampling combined with C4.5 (MDO), the AdaBoost.NC method (Ada) and the EFIS-MOEA method (EFIS).

Table 4.7 lists the full results for the balanced accuracy and MAUC measures. For each dataset, the highest attained result is printed in boldface. With respect to the balanced accuracy, FROVOCO yields the best result on 10 out 18 datasets as well as the highest value on average. For the MAUC, our method wins on 14 out of 18 datasets as well as based on the mean performance. The AdaBoost.NC method can be considered the closest competitor and attains the highest balanced accuracy result on four datasets and the highest MAUC on three. The remaining wins are attained by EFIS-MOEA, which comes in at first place on two datasets (*contraceptive* and *led7digit*) for both evaluation measures. These results demonstrate that our FROVOCO proposal, which integrates the IFROWANN classifier in an OVO scheme with the $\mathcal{W}_{\text{IR}}$ weight combination and the WV-FROST aggregation step, has a better performance than the state-of-the-art in this domain with respect to the balanced accuracy and the MAUC.

The accompanying statistical analysis by means of the Friedman and Wilcoxon tests is presented in Tables 4.8 and 4.9. The Friedman test detects significant performance differences between the seven methods for both evaluation measures. In both cases, our method is assigned the lowest rank (confirming its strong performance) and is used as control method in the Holm post-hoc procedure. FROVOCO is shown to significantly outperform the Static-SMOTE-C4.5 and MDO-C4.5 combinations for both evaluation measures and the OVO-SMOTE-C4.5-WV and GlobalCS-C4.5 methods for the MAUC. To study the differences in performance between FROVOCO and the other methods, we accompany the Friedman test with a Wilcoxon test comparing our proposal to each of the other algorithms. Table 4.9 lists the rank sums R^+ and R^- and p-values associated with the Wilcoxon test as well as the number of wins and losses for FROVOCO in each comparison. It is plain that our method dominates all others. In each comparison, it attains the highest number of wins and the highest rank sum R^+. The tests evaluating the MAUC performance all conclude that FROVOCO significantly outperforms its competitors. With respect to the balanced accuracy, statistically significant differences are only detected in the comparison with the MDO-C4.5 combination. Nevertheless, from its high mean balanced accuracy and high number of wins in each comparison, we can still confidently conclude the superior performance of our proposal. The explanation why no more significant differences are detected for the balanced accuracy lies with the *thyroid* dataset. On this single dataset, our proposal performs very poorly compared to all others (as do all other evaluated versions of IFROWANN within OVO). This performance differ-

Table 4.7 Full balanced accuracy and MAUC results for the state-of-the-art classifiers and our FROVOCO proposal. For each dataset, for each measure, the highest value is printed in bold

Balanced accuracy							
Data	OVO-SMT	GlobalCS	St-SMT	MDO	Ada	EFIS	FROVOCO
automobile	80.6444	**82.9444**	81.6444	76.4778	79.9444	72.3111	77.1556
balance	55.2701	56.3966	55.4163	56.4819	65.8900	60.1749	**78.8514**
cleveland	26.1917	29.5250	31.5750	29.0417	26.8750	30.1417	**33.7833**
contraceptive	51.7246	52.2288	51.6691	48.7521	47.9522	**54.2195**	47.4449
dermatology	96.2096	95.1751	94.0951	95.3709	94.6845	95.6578	**97.1553**
ecoli	71.4609	68.5556	69.7810	71.1481	76.2654	67.8465	**77.2723**
glass	**75.1885**	66.3492	66.7837	63.0437	71.5516	32.6508	67.0694
led7digit	63.5466	63.8680	64.9089	64.1728	54.3621	**65.2918**	64.7918
lymphography	72.2222	72.1825	76.3442	73.2093	72.4355	65.5109	**86.2401**
newthyroid	91.3889	90.7222	89.8333	90.4444	**94.7222**	91.8333	91.1111
pageblocks	89.2398	91.5234	87.2636	83.7520	**91.9105**	91.8161	90.0399
satimage	85.2928	84.4995	84.0159	84.7142	87.5570	33.8754	**89.4955**
shuttle	96.8439	98.4794	97.8236	91.3154	**98.4803**	98.2324	91.8527
thyroid	99.2688	98.9774	98.2533	97.9360	**99.4186**	97.8544	66.4500
wine	95.2698	93.8214	94.3254	92.9881	94.7579	95.5992	**98.2143**
winequality-red	34.1986	35.9825	33.9332	31.8371	39.6884	16.6667	**43.7544**
winequality-white	39.3455	44.8675	36.5483	39.3391	47.6684	15.4762	**47.8895**
yeast	51.8083	51.8429	51.4764	53.3381	49.0789	55.8139	**58.8178**
Mean	70.8397	70.9967	70.3162	69.0757	71.8468	63.3874	**72.6327**
MAUC							
Data	OVO-SMT	GlobalCS	St-SMT	MDO	Ada	EFIS	FROVOCO
automobile	0.9299	0.9133	0.8870	0.8928	0.9370	0.9340	**0.9633**
balance	0.5901	0.6068	0.6656	0.6768	0.8609	0.8245	**0.8854**
cleveland	0.5772	0.5171	0.5723	0.5610	0.5834	0.6307	**0.6981**
contraceptive	0.6560	0.6567	0.6375	0.6529	0.6669	**0.7387**	0.6485
dermatology	0.9861	0.9722	0.9646	0.9727	0.9857	0.9944	**0.9966**
ecoli	0.8990	0.7850	0.8203	0.8556	0.9162	0.9191	**0.9304**
glass	0.9204	0.8430	0.7997	0.8534	0.9246	0.7814	**0.9325**
led7digit	0.9134	0.8385	0.8050	0.8780	0.7640	**0.9207**	0.9189
lymphography	0.7717	0.7845	0.8022	0.8231	0.7847	0.8106	**0.9108**
newthyroid	0.9563	0.9414	0.9238	0.9276	0.9972	0.9811	**0.9981**
pageblocks	0.9739	0.9434	0.9204	0.9446	**0.9876**	0.9859	0.9736
satimage	0.9619	0.9063	0.9041	0.9214	**0.9817**	0.7385	**0.9817**
shuttle	0.9979	0.9920	0.9873	0.9600	0.9911	0.9983	**0.9987**
thyroid	0.9965	0.9944	0.9869	0.9894	**0.9998**	0.9948	0.8494
wine	0.9788	0.9593	0.9574	0.9482	0.9818	0.9938	**1.0000**
winequality-red	0.7495	0.6323	0.6036	0.6432	0.7581	0.5000	**0.8342**
winequality-white	0.7772	0.7116	0.6249	0.6811	0.7856	0.5000	**0.8309**
yeast	0.8472	0.7209	0.7288	0.7693	0.8279	0.8544	**0.8810**
Mean	0.8602	0.8177	0.8106	0.8306	0.8741	0.8389	**0.9018**

Table 4.8 Results of the Friedman test and Holm post-hoc procedure. P-values implying statistically significant differences at the 5% significance level are printed in bold

Method	Balacc		MAUC	
	Rank	p_{Holm}	Rank	p_{Holm}
FROVOCO	2.7222 (1)	–	1.8333 (1)	–
Ada	3.3333 (2)	0.396066	2.8889 (2)	0.14268
OVO-SMT	3.9444 (3)	0.192233	3.7222 (4)	**0.026136**
GlobalCS	4.0556 (4)	0.192233	5.2778 (6)	**0.000009**
EFIS	4.2778 (5)	0.123014	3.2778 (3)	0.089725
St-SMT	4.6111 (6)	**0.04356**	6.0556 (7)	$\leq$ **0.000001**
MDO	5.0556 (7)	**0.007162**	4.9444 (5)	**0.000062**
$p_{Friedman}$	**0.028848**		$\leq$ **0.000001**	

Table 4.9 Results of the Wilcoxon test comparing FROVOCO to the six state-of-the-art classifiers. Lines preceded by 'B' relate to the evaluation by the balanced accuracy, while those starting with 'M' correspond to the MAUC performance. P-values implying statistically significant differences at the 5% significance level are printed in bold. The second column lists the number of wins (W) and losses (L) corresponding to WV-FROST

Comparison	W/L	R^+	R^-	p
B: FROVOCO versus OVO-SMOTE-C4.5-WV	12/6	116.0	55.0	0.19638
B: FROVOCO versus GlobalCS-C4.5	13/5	117.0	54.0	0.18146
B: FROVOCO versus Static-SMOTE-C4.5	13/5	124.0	47.0	0.09874
B: FROVOCO versus MDO-C4.5	16/2	148.0	23.0	**0.004746**
B: FROVOCO versus AdaBoost.NC	11/7	108.0	63.0	0.32714
B: FROVOCO versus EFIS-MOEA	12/6	130.0	41.0	0.05386
M: FROVOCO versus OVO-SMOTE-C4.5-WV	15/3	149.0	22.0	**0.004006**
M: FROVOCO versus GlobalCS-C4.5	16/2	156.0	15.0	**0.0010452**
M: FROVOCO versus Static-SMOTE-C4.5	17/1	157.0	14.0	**0.0008392**
M: FROVOCO versus MDO-C4.5	16/2	155.0	16.0	**0.0012894**
M: FROVOCO versus AdaBoost.NC	15/3	139.0	32.0	**0.018234**
M: FROVOCO versus EFIS-MOEA	14/4	137.0	34.0	**0.02368**

ence is assigned the highest rank in favour of the competing methods and contributes greatly to R^-. As recalled in Sect. 4.4.1, the balanced accuracy and MAUC measures capture different aspects of the classification performance. The former solely focuses on the number of correctly and incorrectly predicted elements, while the latter takes the prediction confidence of the classifier into account. Based on the analysis presented in Table 4.9, we can stress that the prediction confidences of FROVOCO are significantly more reliable than those of its competitors.

As a final aspect, we consider the influence of the actual imbalance on the prediction results. To this end, we combine the results in Table 4.7 with the IR information

in Table 4.1. The datasets on which FROVOCO performs sub-optimally in comparison with the other algorithms are mainly those with a high average IR. In particular, these are the *pageblocks*, *shuttle* and *thyroid* datasets, for which the high average IR value is due to the presence of one very large majority class. When we investigate these results in more detail, we observe that the accuracy of our proposal on the single majority class is notably lower than the accuracy on this class obtained by the other methods. This low value results in an inferior balanced accuracy. The fact that there is only one majority class as well as its absolute size explain why FROVOCO has a relatively poor performance in this situation. Firstly, the method aims to boost the performance on the minority classes to such a degree that the classification performance on the single majority class is negatively affected. The combined effect of all minority classes against the single majority class leads to a decrease in balanced accuracy. Secondly (and perhaps most importantly), when the majority class is very large, the OWA based fuzzy rough approximation operators are hindered in their recognition ability (see Chap. 3). This is due to the effect of the aggregation length on weight vector (4.2), which is used when the IR between a pair of classes is high. When the absolute size of one of these classes is high, the length of the weight vector increases and the weights on the non-zero positions flatten out towards an average. We have discussed this property of the additive weights at length in Chap. 3. As a consequence, the desirable characteristics are lost. On the *winequality-white* dataset, which has an average IR of 61.08, FROVOCO yields the best balanced accuracy result. This does not contradict our analysis above. This dataset contains two majority classes, of which the sizes do not differ strongly. Moreover, the size of the largest class is also not that great compared to that of the *pageblocks*, *shuttle* and *thyroid* datasets.

In summary, as demonstrated by the balanced accuracy and MAUC results in Table 4.7 and the statistical analysis in Tables 4.8 and 4.9, our FROVOCO method outperforms its competitors in multi-class imbalanced classification. It combines local (OVO decomposition) and global (WV-FROST aggregation) views of the data, thereby improving the recognition of minority classes. Only in the particular case when a single massive majority class is present, the user may prefer to apply the AdaBoost.NC method in the classification process instead. For reasons discussed in this section, FROVOCO is less suited to handle this one specific type of problem. This situation can easily be checked for by a user before the application of a multi-class imbalanced classification method.

4.5 Conclusion

This chapter reviewed the domain of imbalanced data classification. An uneven distribution of observations across the decision classes can increase the difficulty to adequately discern between classes. To resolve this issue, a myriad of approaches have been proposed in the literature. Data level techniques modify the training set during preprocessing to obtain a better (sometimes perfect) balance between classes.

Algorithm level methods are imbalance-resistant classifiers, which take the possible presence of class imbalance into account in their learning phase. At its inception, research on class imbalanced data focused on binary problems with one minority and one majority class. Nevertheless, class imbalance presents itself in applications with more than two classes as well.

We have introduced our FROVOCO proposal that extends the binary IFROWANN method from [361] to the multi-class imbalanced setting relying on the OVO decomposition scheme. The latter is a traditional way to reduce a multi-class problem to several two-class sub-problems, on which binary methods can be applied. Within this decomposition setting, we use an adaptive version of the IFROWANN method, which chooses its weight settings based on the imbalance present in a particular subproblem. The OVO step does not yield an immediate prediction for a target instance. Instead, it collects classifier confidence degrees and generates a score-matrix. An aggregation procedure is required to extract one class prediction from this matrix. We have proposed a new such aggregator, called WV-FROST, that further complements the local information grouped in the score-matrix with global information extracted from the full multi-class dataset.

We have extensively evaluated our proposal in the experiments conducted in this chapter. By means of an empirical validation of both components of our FROVOCO method (our adaptive IFROWANN-$\mathcal{W}_{\text{IR}}$ method and our WV-FROST aggregator), we can confidently conclude their strength. The latter is a dynamic aggregation procedure, for which alternatives have already been proposed in the literature. We have shown that, within our FROVOCO method, WV-FROST results in a superior performance compared to the existing approaches. As a final step, we have compared FROVOCO to the state-of-the-art in multi-class imbalanced classification and have again been able to show the dominance of our proposal based on both the balanced accuracy and MAUC.

Another aspect that can be assessed in future research is the interaction of the WV-FROST aggregation with other classifiers in the OVO decomposition. Within FROVOCO, the fuzzy rough operators are used in both the binary classifiers and global affinity terms and nicely complement each other in terms of local and global information respectively. When a different classifier is used, this synergy may be lost or, equally likely, the WV-FROST affinity terms can still provide a sufficient assessment of the global relation of instances with classes to benefit the OVO score-matrix. This remains to be verified.

Chapter 5
Fuzzy Rough Set Based Classification of Semi-supervised Data

In several classification applications, obtaining labelled training instances is costly or difficult. While the feature values of observations can be relatively easy to collect, sufficient resources and expert knowledge are required to (manually) assign these elements to their correct classes. This phenomenon is addressed by the introduction of semi-supervised classification, in which a prediction model is derived from a training set consisting of both labelled and unlabelled data. Information in both the labelled and unlabelled parts of the training set can be used in the classification process.

In this chapter, we evaluate the performance of fuzzy rough set based methods in semi-supervised classification. In Sect. 5.1, we first introduce this domain with a specific focus on the self-labelling approach. Self-labelling methods extend the labelled part of the training set by predicting class labels for unlabelled training instances and accepting the most confident predictions among them. This intuitive approach to semi-supervised classification has been adopted by many researchers. In Sect. 5.2, we evaluate our fuzzy rough classifiers from Chap. 3 for this classification task. We combine them with a variety of self-labelling techniques in order to study the interaction between the fuzzy rough classification mechanism and self-labelling. Our experiments show that the self-labelling step is not necessary and that our fuzzy rough methods significantly outperform existing semi-supervised classification methods by only relying on the originally labelled training elements. Finally, our overall conclusions are presented in Sect. 5.3.

5.1 Semi-supervised Classification

Semi-supervised learning is located on the midpoint between supervised and unsupervised learning and incorporates aspects of both settings [72, 514]. A semi-supervised training set T consists of a labelled part L and unlabelled part U, where the size

S. Vluymans, *Dealing with Imbalanced and Weakly Labelled Data in Machine Learning using Fuzzy and Rough Set Methods*, Studies in Computational Intelligence 807, https://doi.org/10.1007/978-3-030-04663-7_5

of the former is usually considerably smaller than that of the latter. The information contained in the full training set, both labelled and unlabelled instances, can be exploited. Two prominent sub-categories are semi-supervised classification and semi-supervised clustering. The goal in a *semi-supervised classification* task is to construct a strong classification model based on the information present in both L and U. This setting leans more toward the supervised learning paradigm with its general aim to predict the outcome of unseen elements. *Semi-supervised clustering* (also, constrained clustering) adheres more closely to unsupervised learning. Similar to the traditional clustering task, the goal is to detect well-separated groups of similar elements. The extension to semi-supervised clustering comes with the constraints represented by the labelled data corresponding to must-links (elements that should be in the same cluster) and cannot-links (elements that should be in different clusters) (e.g. [27, 452]). We focus on the classification setting in this chapter, as we do elsewhere in the book.

It should be noted that the classification behaviour of a semi-supervised classification algorithm can be decomposed into two components. The *transductive learning phase* relates to the prediction performance on the elements in U, of which the feature values are available during training but the labels are not. The evaluation on a separate test set corresponds to the *inductive learning phase*.

In this chapter, we focus on self-labelling methods to perform semi-supervised classification. Sect. 5.1.1 recalls a recently proposed taxonomy for these techniques and lists several well-performing methods. Semi-supervised classification approaches other than self-labelling are discussed in Sect. 5.1.2. We conclude this section with a brief discussion on recent application papers in the semi-supervised classification domain in Sect. 5.1.3.

5.1.1 Self-Labelling Techniques

One particularly intuitive semi-supervised classification approach is self-labelling. Generally put, these methods have an iterative nature and try to predict the missing class labels in the training set, such that the final classification phase can rely on a larger set of labelled instances. In each iteration, class predictions for the elements in U are derived based on the labelled elements in L. The most confident predictions are accepted, increasing the size of L and the information contained in it.

A self-labelling technique extends the set L to one or more supersets L'. A taxonomy for these methods was proposed in the recent review paper [406]. The authors note that the traditional distinction between self-training [466], co-training [51] and its generalization to multi-view learning [389] does not suffice to capture every defining difference between existing algorithms. Their proposed taxonomy consists of the following four categories:

- **Addition mechanism**: in constructing L' from L, several search directions can be followed. An *incremental* algorithm adds elements to L step by step, increasing

its size in each iteration. A *batch* approach first marks all elements to be labelled and added to L and only effectively adds them after this assessment process has been completed. The third option is to implement an *amending* procedure, where instances can be added to or removed from L' in each iteration. This allows the method to undo previous labelling decisions.

- **Number of classifiers**: a distinction is made between *single-classifier* and *multi-classifier* systems. The former uses a single classifier to predict the labels of elements in U, while the latter aggregates the predictions of multiple classifiers to derive these class decisions.
- **Number of learning algorithms**: when multiple classifiers are used, they can either use the same learning algorithm (*single-learning*) or multiple different learning algorithms (*multi-learning*). A single-classifier method uses single-learning by definition, while multi-classifier systems can opt to follow either the single-learning or multi-learning approach.
- **Type of view**: several existing self-labelling methods perform one or more vertical splits of the set L, that is, they construct several smaller datasets based on feature subsets. The intuition behind this *multi-view* approach is to use redundant and conditionally independent views of the same data to improve the classification ability of the method. When no splits are applied and only the full set L is used in the learning process, the method is denoted as *single-view*.

Aside from these main defining taxonomic characteristics, the following additional properties are highlighted in [406]:

- **Confidence measure**: to decide which elements from U to add to L, a confidence degree on the class prediction is computed. Different types of confidence measures have been developed. *Simple* measures are directly derived from the prediction process from the single classifier used in the labelling step. *Agreement and combination* measures are used within multi-classifier methods. One option is to derive the confidence degree from the agreement between the predictions of the classifiers, e.g. by a majority voting scheme. An alternative is to aggregate the confidence degrees provided by the different classifiers.
- **Teaching approach**: this property relates to multi-classifier algorithms. They can perform *mutual-teaching*, where the classifiers exchange their predictions and increase each other's custom training sets. The final complete set L' is constructed by combining the sets processed and enlarged by the different classifiers. The *self-teaching* approach works on a single labelled set and uses the combined information provided by all classifiers to extend it in each iteration.
- **Stopping criteria**: at a certain point, the construction of L' is halted. It is not necessary for all elements in U to be labelled, several alternative stopping criteria have been proposed in existing self-labelling methods.

Aside from the taxonomic division recalled above, the contribution of [406] also lies with a thorough experimental comparison of 15 self-labelling methods in combination with different base classifiers. The authors used a collection of 64 datasets and varied the percentage of labelled elements in the training sets. Their evaluation allowed them to put forward the following strong performing algorithms:

- Standard co-training [51] with SMO (support vector machines with sequential minimal optimization, [346]) as base classifier. The co-training procedure is an incremental, multi-classifier, single-learning, multi-view (concretely, two views) method that performs mutual-teaching. The labelled set L is split into two parts based on two feature subsets. A classifier is trained on each of these sets and, in later iterations, on their enlarged versions. The most confident predictions of one classifier are added to the training set of the other.
- TriTraining [506] with C4.5 as base classifier. This is an incremental, multi-classifier, single-learning, single-view algorithm, that determines the most confident predictions based on three classifiers learned from bootstrapped versions of L.
- Democratic co-learning (DemCo, [503]). As an incremental, multi-classifier, multi-learning, single-view method, DemCo relies on three classifiers (3NN, C4.5, Naive Bayes). The mutual learning approach is followed, where confident predictions for unlabelled elements are added to the labelled training set of those classifiers that predicted a class different than the majority.
- Co-bagging [201, 202] with C4.5 as base classifier. This method sits in the same place of the taxonomy as TriTraining and is an incremental, multi-classifier, single-learning, single-view method. A classifier committee is constructed by training multiple classifiers following the standard bagging procedure ([58], Sect. 2.2.7).

Finally, we note that the SEGSSC framework was recently proposed in [405]. Any self-labelling technique (including the ones listed above) can be integrated within it. SEGSSC addresses the small size and sparsity of L by generating synthetic elements. The acronym stands for Synthetic Examples Generation for Self-labelled Semi-supervised Classification. The generation process is based on oversampling (Sect. 4.1.2) and position adjustment techniques. The experimental evaluation conducted in [405] showed the improvement in prediction performance when using a self-labelling method within the SEGSSC framework.

5.1.2 Other Semi-supervised Classification Techniques

Aside from the self-labelling approach discussed in the preceding section, three other groups of semi-supervised classification techniques can be listed [514]. For the sake of completeness, we briefly discuss them here.

- **Generative models and cluster-then-label methods**: as discussed in Sect. 2.2.6, a generative model approximates the joint distribution of $P(X, Y)$ from the provided training set. A semi-supervised generative model uses both labelled and unlabelled elements to learn this distribution. Examples of this approach can be found in e.g. [3, 167, 199, 261, 304, 395]. Cluster-then-label methods are a related setting, where the training set is first clustered and then labelled using the labelled elements within each constructed cluster (e.g. [10, 116, 119, 156, 269]).

- **Semi-supervised support vector classification**: semi-supervised support vector machines have been proposed as an extension of the traditional support vector machine classifier (see Sect. 2.2.3) to handle partly labelled training data (e.g. [4, 39, 73, 251, 397]). The inclusion of unlabelled elements in the learning process results in a non-convex optimization problem. The intuition behind clustering is incorporated: instances that are close in the feature space should have similar outcomes. The learned SVM decision boundary should (i) sufficiently separate classes and (ii) not cross dense regions of unlabelled elements.
- **Graph-based methods**: these techniques formulate the semi-supervised classification task as a graph min-cut problem [50]. A weighted graph representation of the dataset is generated, in which each training instance (labelled or unlabelled) corresponds to a node. Weights of edges are determined based on the similarity between the instances they connect. A minimum cut of the graph is determined in order to partition the classes. Graph-based semi-supervised classification methods have been proposed in e.g. [33, 70, 252, 387, 432, 451].

5.1.3 Applications

Several application domains can be listed in which partially labelled and partially unlabelled datasets present themselves. In this section, we collect a number of recently published application papers that use semi-supervised methods to solve specific prediction problems.

A first domain of note is that of natural language processing [254, 316]. Its core task revolves around language recognition, which can be in the form of speech analysis, text analysis or text generation. Semi-supervised classification algorithms have recently been developed for speech analysis [300], sentiment analysis [114, 502], big social data analysis [237] and personality prediction [288]. The book review [384] provides an overview of semi-supervised algorithms for natural language processing and how this setting allows to deal with the inherent data sparsity of such problems.

Research in bioinformatics often encounters semi-supervised data as well. Technological advances have led to a relatively easy data acquisition, while the labelling task remains costly and commonly requires a human expert. Semi-supervised datasets are a necessary compromise. Examples of recent work focus on phenotype classification [131], gene regulatory inference [345], the taxonomic assignment of metagenomic reads [412] and protein sub-cellular localization prediction [457].

A third domain in which researchers have made ample use of semi-supervised techniques is image recognition and classification. Recent proposals can be found for general image classification (e.g. [115, 197, 310]) with specific examples of glaucoma detection [30], remote sensing image classification [236] and hyperspectral image classification [430]. The face recognition task has been addressed with semi-supervised methods as well (e.g. [185, 186, 427]).

Some miscellaneous application examples include intrusion detection (e.g. [20, 238, 336]), vegetation classification (e.g. [398]) and medical diagnosis (e.g. [481, 496, 512]).

5.2 Fuzzy Rough Set Based Classifiers and Self-Labelling

The aim of this chapter is to assess how our proposed fuzzy rough classifiers from Chap. 3 with our optimized weighting scheme selection strategies perform on semi-supervised data and how they interact with self-labelling techniques. Section 5.2.1 studies the performance of the OWA based fuzzy rough set model (in the form of the classifiers proposed in Chap. 3) on semi-supervised data. We evaluate their prediction strength on datasets where a substantial part of the training set is unlabelled. In Sect. 5.2.2, we combine the fuzzy rough classification step with several self-labelling approaches to analyse whether the former benefits from the latter. Section 5.2.3 presents a comparison of the performance of our fuzzy rough approach and existing semi-supervised classification methods relying on self-labelling. Lastly, Sect. 5.2.4 discusses the results and outlines directions for future research in this area.

5.2.1 OWA Based Fuzzy Rough Classifiers on Semi-supervised Data

As demonstrated in Chap. 3, selecting an appropriate weighting scheme for the OWA based fuzzy rough operators based on the data at hand further increases the strength of the OWA based fuzzy rough set model in the classification task. In this section, we use our proposed weighting scheme selection strategy and study the prediction performance of this model on semi-supervised data. At this stage, we only use the elements in L to derive class predictions. We do not first extend L to a superset L' with additional labelled elements, that is, we do not perform any self-labelling at this point. As a result, although we report the performance on both, there is no fundamental difference between the evaluation on the unlabelled instances in U (transductive phase) and those in the test set (inductive phase). The calculations in the prediction step are independent of both.

We use 30 datasets provided by [406] and apply the ten-fold cross validation scheme. The percentages of labelled instances range from 10 to 40%. These datasets were constructed by randomly removing the labels of a certain percentage of training elements, ensuring that at least one labelled instance is present for each class. The remaining dataset characteristics are presented in Table 5.1. We use the balanced accuracy to measure the prediction performance of all classifiers. We compare their performance to the base classifiers used in [406], namely the 1NN algorithm, the decision tree classifier C4.5 and the support vector machine method SMO. The

Table 5.1 The 30 datasets used in the experimental study of Sects. 5.2.1–5.2.3. The number of features is divided between numeric and nominal ones, e.g. the abalone dataset has 7 numeric features and 1 nominal feature

Dataset	# inst	# feat	# classes	Dataset	# inst	# feat	# classes
Abalone	4174	8 (7/1)	28	Lymphography	148	18 (3/15)	4
Appendicitis	106	7 (7/0)	2	Mammographic	830	5 (5/0)	2
Australian	690	14 (8/6)	2	Monk-2	432	6 (6/0)	2
Automobile	159	25 (15/10)	6	Movement_libras	360	90 (90/0)	15
Banana	5300	2 (2/0)	2	Pima	768	8 (8/0)	2
Bupa	345	6 (6/0)	2	Saheart	462	9 (8/1)	2
cleveland	303	13 (13/0)	5	Segment	2310	19 (19/0)	7
Contraceptive	1473	9 (9/0)	3	Sonar	208	60 (60/0)	2
Crx	653	15 (6/9)	2	Spambase	4597	57 (57/0)	2
Dermatology	358	34 (34/0)	6	Spectfheart	267	44 (44/0)	2
Ecoli	336	7 (7/0)	8	Titanic	2201	3 (3/0)	2
German	1000	20 (7/13)	2	Vehicle	846	18 (18/0)	4
Glass	214	9 (9/0)	6	Vowel	990	13 (13/0)	11
Haberman	306	3 (3/0)	2	Wisconsin	683	9 (9/0)	2
Heart	270	13 (13/0)	2	Yeast	1484	8 (8/0)	10

parameters of these methods are set to the values considered in [406], as we are using the same datasets and partitions as this study as well.

We include three OWA based fuzzy rough classifiers. Version 'Lower' uses the lower approximation as predictor (see Sect. 3.3), while version 'Upper' relies on the upper approximation to make class predictions (see Sect. 3.4). The third version 'Both' combines the lower and upper approximations in the class score calculations (see Sect. 3.5.3). For each of these, the OWA weighting schemes are selected according to our guidelines derived in Chap. 3. Table 5.2 presents the mean transductive and inductive balanced accuracy across the 30 datasets in Table 5.1. The leftmost column lists the percentage of labelled instances in the training set. For the inductive balanced accuracy, we include the evaluation with a fully labelled training set as well. The results allow us to conclude the following:

- On average, the fuzzy rough methods provide better balanced accuracy results than the 1NN, C4.5 and SMO classifiers. This holds for both the transductive and inductive results and for all evaluated percentages of labelled training instances.
- The performance differences between the three fuzzy rough methods are minor and we can consequently opt to limit ourselves to the OWA based lower approximation method.
- In Table 5.3, this classifier is compared to 1NN, C4.5 and SMO by means of the Wilcoxon test. For the former two methods, the difference in performance is always found to be significant and this in favour of our fuzzy rough method.

Table 5.2 Mean transductive and inductive balanced accuracy results for the fuzzy rough methods and the 1NN, C4.5 and SMO classifiers

	Transductive					
	Lower	Upper	Both	1NN	C4.5	SMO
10%	0.6396	0.6397	0.6425	0.6122	0.5923	0.6239
20%	0.6807	0.6727	0.6774	0.6400	0.6338	0.6440
30%	0.6987	0.6923	0.7001	0.6561	0.6521	0.6688
40%	0.7142	0.7053	0.7139	0.6677	0.6694	0.6776
	Inductive					
	Lower	Upper	Both	1NN	C4.5	SMO
10%	0.6447	0.6443	0.6470	0.6102	0.6035	0.6261
20%	0.6847	0.6750	0.6805	0.6481	0.6402	0.6572
30%	0.7024	0.6921	0.6997	0.6664	0.6643	0.6744
40%	0.7208	0.7090	0.7166	0.6812	0.6815	0.6900
100%	0.7463	0.7423	0.7456	0.7047	0.7157	0.7147

With respect to the comparison with the SMO method, the OWA based fuzzy rough lower approximation significantly outperforms this classifier for almost all settings as well.

The above study reaffirms the strength of our weighting scheme selection guidelines proposed in Chap. 3. In particular, the evaluation with 100% labelled training data shows that the basic lower approximation classifier outperforms the standard 1NN, C4.5 and SMO algorithms. Our method upholds this strong performance even when the percentage of labelled training instances is drastically decreased. This indicates its strong robustness against a lack of data. As noted in Chap. 3 may be particularly well suited for classification with small training sets and this is confirmed here. No large or massive amounts of training observations are required to guarantee a solid classification performance of our fuzzy rough set based methods.

5.2.2 *Interaction with Self-Labelling Schemes*

As a next step in our evaluation, we integrate the OWA based lower approximation classifier with our weighting scheme selection guidelines from Sect. 3.3 in the well-performing self-labelling schemes discussed in Sect. 5.1.1. This step is performed to evaluate whether our classifier benefits from self-labelling at all or whether it suffices to only explicitly use the information in L to derive confident predictions. We include the following classification models:

Table 5.3 Results of the Wilcoxon tests comparing the OWA based fuzzy rough lower approximation classifier to 1NN, C4.5 and SMO methods on semi-supervised data. P-values that imply statistically significant differences at the 5% significance level are printed in boldface

	Transductive								
	Lower vs 1NN			Lower vs C4.5			Lower vs SMO		
	R^+	R^-	p	R^+	R^-	p	R^+	R^-	p
10%	371.0	94.0	**0.00425**	390.0	75.0	**0.001155**	284.0	181.0	0.28482
20%	397.0	68.0	**0.000689**	379.0	86.0	**0.002498**	322.0	113.0	**0.023181**
30%	407.0	58.0	**0.000319**	394.5	70.5	**0.000805**	336.0	129.0	**0.032427**
40%	403.0	62.0	**0.000421**	400.0	65.0	**0.000549**	335.0	130.0	**0.034129**
	Inductive								
	Lower vs 1NN			Lower vs C4.5			Lower vs SMO		
	R^+	R^-	p	R^+	R^-	p	R^+	R^-	p
10%	370.0	95.0	**0.004431**	394.0	71.0	**0.000862**	290.0	175.0	0.231681
20%	403.0	62.0	**0.000436**	412.0	53.0	**0.000214**	330.0	135.0	**0.043832**
30%	395.0	70.0	**0.000800**	373.5	91.5	**0.003524**	340.0	125.0	**0.026325**
40%	404.0	61.0	**0.000404**	380.0	85.0	**0.002334**	311.0	154.0	0.104184
100%	429.0	36.0	**0.000051**	357.0	108.0	**0.010139**	346.5	118.5	**0.018219**

- Lower: the method evaluated in the previous section. It uses the OWA based fuzzy rough lower approximation to classify instances. Within this operator, the method uses our weighting scheme selection strategy proposed in Sect. 3.3.
- SelfTr(FR), CoTr(FR), TriTr(FR), CoBag(FR): the standard self-training [466], co-training [51], TriTraining [506] and CoBagging [201, 202] methods using the above fuzzy rough classifier as base classifier during self-labelling as well as final classifier.
- CoTr(SMO)+FR, TriTr(C45)+FR, CoBag(C45)+FR: during the self-labelling phases, these methods coincide with the ones listed in Sect. 5.1.1, namely the co-training algorithm with SMO as base classifier, the TriTraining method with C4.5 as base classifier and the CoBagging method with C4.5 as base classifier respectively. The final classification of U and the test set is performed by the fuzzy rough classifier.
- FR-SSL: the study of [311] proposed a naive fuzzy rough self-labelling method that labels all instances in U by using the fuzzy rough approximation operators. This is a straightforward fuzzy rough set based self-labelling approach, which always results, as opposed to the other self-labelling methods listed above, in a fully labelled training set. We use our classifiers from Chap. 3 in these labelling steps in order to verify how they interact with this algorithm.

For the self-labelling methods apart from FR-SSL, we use the same parameter settings as in [406]. The combinations CoTr(SMO), TriTr(C45) and CoBag(C45) were put forward as the best-performing self-labelling alternatives in that study. In this evaluation, we modify these settings in two ways. In a first version, we replace their

Table 5.4 Mean balanced accuracy results of the included self-labelling methods and classifiers. For algorithms containing random components, we report the standard deviation of the mean balanced accuracy across ten runs. For each setting, the highest mean value is printed in boldface

	Transductive			
	10%	20%	30%	40%
Lower	**0.6396**	**0.6807**	**0.6987**	**0.7142**
SelfTr(FR)	0.6197	0.6602	0.6831	0.6972
CoTr(FR)	0.6182 ± 0.0106	0.6611 ± 0.0094	0.6838 ± 0.0079	0.6975 ± 0.0082
CoTr(SMO)+FR	0.6170 ± 0.0107	0.6530 ± 0.0095	0.6730 ± 0.0078	0.6860 ± 0.0089
TriTr(FR)	0.6302 ± 0.0072	0.6726 ± 0.0054	0.6942 ± 0.0045	0.7086 ± 0.0049
TriTr(C45)+FR	0.6280 ± 0.0100	0.6672 ± 0.0083	0.6919 ± 0.0059	0.7044 ± 0.0060
CoBag(FR)	0.6156 ± 0.0091	0.6546 ± 0.0111	0.6820 ± 0.0082	0.7008 ± 0.0079
CoBag(C45)+FR	0.6227 ± 0.0099	0.6587 ± 0.0109	0.6860 ± 0.0081	0.7038 ± 0.0076
FR-SSL	0.5123	0.5466	0.5715	0.5930
	Inductive			
	10%	20%	30%	40%
Lower	**0.6447**	**0.6847**	**0.7024**	**0.7208**
SelfTr(FR)	0.6234	0.6577	0.6841	0.7004
CoTr(FR)	0.6231 ± 0.0136	0.6644 ± 0.0103	0.6890 ± 0.0094	0.7040 ± 0.0092
CoTr(SMO)+FR	0.6233 ± 0.0145	0.6609 ± 0.0115	0.6854 ± 0.0088	0.6969 ± 0.0094
TriTr(FR)	0.6323 ± 0.0093	0.6742 ± 0.0076	0.6965 ± 0.0070	0.7139 ± 0.0052
TriTr(C45)+FR	0.6379 ± 0.0124	0.6769 ± 0.0107	0.6976 ± 0.0077	0.7111 ± 0.0072
CoBag(FR)	0.6211 ± 0.0127	0.6607 ± 0.0124	0.6854 ± 0.0107	0.7062 ± 0.0107
CoBag(C45)+FR	0.6282 ± 0.0138	0.6647 ± 0.0124	0.6887 ± 0.0105	0.7087 ± 0.0110
FR-SSL	0.5050	0.5521	0.5672	0.5915

base classifier by the 'Lower' method (represented by CoTr(FR), TriTr(FR) and CoBag(FR)). The second version corresponds to the CoTr(SMO), TriTr(C45) and CoBag(C45) methods themselves, in which we have only replaced the final classifier by our fuzzy rough method. In the next section, we will compare the results of the fuzzy rough methods with the original methods used in [406] as well. Note that we do not include the SEGSSC framework from [405] at this point, since we first wish to study the precise interplay of our fuzzy rough classifier with the pure self-labelling techniques without the results being affected by the further interaction between SEGSSC and self-labelling. We also do not test the third possible modification of the self-labelling methods evaluated in [406], wherein we would use the fuzzy rough method as base classifier and maintain SMO or C4.5 as final classifier, since this evaluation carries no information on the effect of self-labelling on the fuzzy rough set based classifier.

The mean results are reported in Table 5.4 and the accompanying statistical analysis can be found in Table 5.5. We use the Wilcoxon test to compare the performance of our OWA based fuzzy rough lower approximation classifier with and without

Table 5.5 Results of the Wilcoxon test comparing our fuzzy rough classifier 'Lower' to the other algorithms in Table 5.4 in the format 'R^+/R^-/p'. The R^+ value always corresponds to the fuzzy rough method. P-values implying statistically significant differences at the 5% significance level are printed in boldface

	Transductive			
	10%	20%	30%	40%
SelfTr(FR)	346.5/118.5/**0.018219**	382.0/83.0/**0.00198**	376.5/88.5/**0.002813**	367.5/97.5/**0.005088**
CoTr(FR)	398.5/66.5/**0.000616**	406.5/58.5/**0.000332**	392.5/42.5/**0.000136**	431.0/34.0/**0.000043**
CoTr(SMO)+FR	426.0/39.0/**0.000066**	435.0/30.0/**0.00003**	435.0/30.0/**0.00003**	412.0/53.0/**0.000214**
TriTr(FR)	384.0/81.0/**0.00177**	416.0/49.0/**0.000154**	385.0/80.0/**0.00165**	414.0/51.0/**0.000182**
TriTr(C45)+FR	297.0/168.0/0.180119	319.5/145.5/0.071174	240.5/194.5/0.610386	275.0/16.0/0.202373
CoBag(FR)	446.0/19.0/**0.000011**	463.0/2.0/**0.000002**	455.0/10.0/**0.000005**	453.0/12.0/**0.000005**
CoBag(C45)+FR	353.0/112.0/**0.012819**	418.0/47.0/**0.00013**	417.0/48.0/**0.000142**	407.0/58.0/**0.000319**
FR-SSL	436.0/29.0/**0.000027**	438.0/27.0/**0.000023**	438.0/27.0/**0.000023**	408.0/57.0/**0.000295**
	Inductive			
	10%	20%	30%	40%
SelfTr(FR)	375.0/90.0/**0.003269**	372.0/63.0/**0.000803**	337.0/98.0/**0.009465**	372.5/62.5/**0.000748**
CoTr(FR)	372.5/62.5/**0.000748**	354.0/81.0/**0.002974**	363.5/71.5/**0.001447**	386.5/48.5/**0.000247**
CoTr(SMO)+FR	390.0/75.0/**0.001155**	389.0/76.0/**0.001241**	388.0/77.0/**0.001334**	418.0/47.0/**0.00013**
TriTr(FR)	402.0/63.0/**0.000471**	333.0/132.0/**0.037764**	313.0/152.0/0.095706	414.0/51.0/**0.000182**
TriTr(C45)+FR	309.5/155.5/0.110008	293.0/172.0/0.206079	242.5/192.5/0.57828	308.5/156.5/0.114673
CoBag(FR)	449.0/16.0/**0.000008**	442.0/23.0/**0.000016**	451.0/14.0/**0.000007**	444.0/21.0/**0.000013**
CoBag(C45)+FR	377.0/88.0/**0.00286**	415.0/50.0/**0.000167**	422.0/43.0/**0.000093**	411.0/54.0/**0.000232**
FR-SSL	432.0/33.0/**0.000039**	426.0/39.0/**0.000066**	416.0/49.0/**0.000154**	403.0/62.0/**0.000436**

self-labelling, as we wish to verify whether it benefits from a self-labelling step. The results in Table 5.4 indicate that this may not be the case. For both the transductive and inductive performance, for all evaluated percentages of labelled instances in the training set, the highest average result is obtained by the classifier without self-labelling. It only uses the information in L to classify the instances in U and the test set and outperforms other methods that have extended the set L by labelling some additional training instances. The statistical analysis in Table 5.5 confirms that no increase in classification performance of our fuzzy rough method is obtained after self-labelling. Instead, almost all pairwise comparisons show that a statistically significant performance drop follows from trying to extend the set of labelled training instances. Only for the TriTr(C45)+FR combination do we not observe significant differences, although the high R^+ values for the classifier without self-labelling again express that the TriTraining step does more harm than good. We also note that the performance of the FR-SSL method is particularly poor. It labels the full set U, which is clearly too extreme an option and results in a considerable and significant decrease in prediction performance.

Starting from overall sparse training data, a self-labelling process creates denser regions of same-class elements. The average similarity of labelled elements with

other labelled instances of the same class increases, while the average similarity with labelled elements in opposite classes decreases. As our fuzzy rough classifier heavily relies on instance similarity values in its class predictions, the creation of such class islands in feature space can result in misclassification errors in sparser regions. Our method performs better on the original datasets, where the sparsity is distributed more evenly across the feature space. In fact, the question whether or not knowledge of unlabelled training instances truly improves the performance of a classifier trained on semi-supervised data is an important and ongoing topic of discussion in the literature. Several studies have shown that genuine semi-supervised classifiers can be outperformed by supervised methods trained on the labelled data only (e.g. [37, 82, 266, 283, 285, 383, 490]). In a way, this is not altogether surprising, as the self-labelling process shows a close relation to imputation and the true signal present in the labelled part of the training set may be diminished or averaged out in this process, rendering prediction more cumbersome.

5.2.3 Comparison with Other Classifiers

Based on the above analysis, we can claim that our classifier from Sect. 3.3 can be used to adequately classify semi-supervised data as is, that is, without first self-labelling the training set. It remains to be verified whether its classification strength can rival that of existing semi-supervised classification algorithms that do rely on self-labelling, in particular methods exhibiting a strong classification performance in earlier studies. We compare our classifier to the four best performing algorithms from the extensive comparative study conducted in [406]. As discussed in Sect. 5.1.1, these are the co-training algorithm with SMO as base classifier (CoTr(SMO)), the TriTraining method with C4.5 as base classifier (TriTr(C45)), democratic co-learning (DemCo) and co-bagging with C4.5 as base classifier (CoBag(C45)). For all included algorithms, we use the same parameter settings as considered in [406]. Aside from these methods, we also evaluate the latter three within the SEGSSC framework from [405] (SEGSSC-TriTr(C45), SEGSSC-DemCo, SEGSSC-CoBag(C45)).

The experimental results, averaged across the datasets in Table 5.1, are reported in Table 5.6. Based on their mean performance, our fuzzy rough classifier stands strong against all other included methods, outperforming all of them with respect to the classification of both the unlabelled training instances in U (transductive performance) and the independent test set (inductive performance). This holds for all percentages of labelled training elements. The performance difference with the standard self-labelling algorithms used in [406] is substantial. These methods do provide better results when used within the SEGSSC framework, but without reaching the level of our OWA based lower approximation predictor with the weighting scheme selection strategy from Sect. 3.3.

We report the results of the statistical analysis in Tables 5.7 and 5.8. The former compares the performance of the included classifiers by means of the Friedman test. For each setting, the p-value of this test is smaller than 0.05 and indicates that sig-

Table 5.6 Mean balanced accuracy results of our fuzzy rough classifier and existing self-labelling techniques. For algorithms containing random components, we report the standard deviation of the mean balanced accuracy across ten runs. For each setting, the highest mean value is printed in boldface

	Transductive			
	10%	20%	30%	40%
Lower	**0.6396**	**0.6807**	**0.6987**	**0.7142**
CoTr(SMO)	0.5940 ± 0.0113	0.6323 ± 0.0099	0.6589 ± 0.0084	0.6710 ± 0.0083
TriTr(C45)	0.6023 ± 0.0106	0.6372 ± 0.0078	0.6583 ± 0.0084	0.6748 ± 0.0068
DemCo	0.5934 ± 0.0002	0.6298 ± 0.0002	0.6552 ± 0.0002	0.6816 ± 0.0002
CoBag(C45)	0.5935 ± 0.0133	0.6310 ± 0.0115	0.6523 ± 0.0099	0.6680 ± 0.0097
SEGSSC-TriTr(C45)	0.6263 ± 0.0082	0.6616 ± 0.0072	0.6788 ± 0.0080	0.6918 ± 0.0075
SEGSSC-DemCo	0.6284 ± 0.0077	0.6602 ± 0.0077	0.6728 ± 0.0082	0.6859 ± 0.0080
SEGSSC-CoBag(C45)	0.6273 ± 0.0047	0.6603 ± 0.0040	0.6702 ± 0.0044	0.6840 ± 0.0043
	Inductive			
	10%	20%	30%	40%
Lower	**0.6447**	**0.6847**	**0.7024**	**0.7208**
CoTr(SMO)	0.5962 ± 0.0168	0.6458 ± 0.0121	0.6696 ± 0.0102	0.6753 ± 0.0096
TriTr(C45)	0.6116 ± 0.0157	0.6497 ± 0.0118	0.6701 ± 0.0109	0.6862 ± 0.0104
DemCo	0.6002 ± 0.0003	0.6368 ± 0.0003	0.6750 ± 0.0003	0.6875 ± 0.0001
CoBag(C45)	0.6039 ± 0.0177	0.6441 ± 0.0165	0.6647 ± 0.0153	0.6780 ± 0.0147
SEGSSC-TriTr(C45)	0.6328 ± 0.0121	0.6727 ± 0.0127	0.6847 ± 0.0131	0.6980 ± 0.0131
SEGSSC-DemCo	0.6322 ± 0.0142	0.6676 ± 0.0138	0.6818 ± 0.0135	0.6943 ± 0.0129
SEGSSC-CoBag(C45)	0.6314 ± 0.0093	0.6700 ± 0.0087	0.6831 ± 0.0092	0.6967 ± 0.0092

nificant differences in performance exist between the algorithms. Our fuzzy rough classifier always receives the lowest Friedman rank and is therefore used as control method in every Holm post-hoc procedure. The associated p_{Holm} values show that our method often outperforms the competitor algorithms with statistical significance. Only for the SEGSSC-TriTr(C45) combination can we never conclude that the performance of the fuzzy rough classifier is significantly superior. In Table 5.8, we perform a pairwise comparison of our method to the seven other algorithms included in Table 5.6 by means of the Wilcoxon test. This mostly confirms our findings from the group analysis in Table 5.7 and even more statistically significant differences are detected in favour of our proposal.

Note that we have not evaluated our fuzzy rough method within the SEGSSC framework. When doing so, we could, based on the reported results in Table 5.6, possibly expect a small increase in the classification performance relative to that of the combination methods in Table 5.4. However, the SEGSSC framework requires the algorithm to first be integrated within a self-labelling procedure (e.g. TriTraining

Table 5.7 Results of the Friedman test comparing the algorithms from Table 5.6. P-values implying statistically significant differences at the 5% significance level are printed in boldface

	Transductive			
	10%		20%	
	Rank	p_{Holm}	Rank	p_{Holm}
Lower	3.0333 (1)	-	2.7833 (1)	-
CoTr(SMO)	5.5000 (7)	**0.000577**	4.9667 (5)	**0.002225**
TriTr(C45)	4.8000 (5)	**0.020866**	5.2000 (7)	**0.000797**
DemCo	5.2333 (6)	**0.002521**	5.1500 (6)	**0.000913**
CoBag(C45)	6.1333 (8)	**0.000007**	5.8333 (8)	**0.000010**
SEGSSC-TriTr(C45)	3.6000 (2)	0.740528	3.5000 (2)	0.257151
SEGSSC-DemCo	3.6000 (2)	0.740528	4.0667 (3)	0.084980
SEGSSC-CoBag(C45)	4.1000 (4)	0.275071	4.5000 (4)	**0.019925**
$p_{Friedman}$	**0.000001**		**0.00001**	
	30%		40%	
	Rank	p_{Holm}	Rank	p_{Holm}
Lower	2.7000 (1)	-	2.8667 (1)	-
CoTr(SMO)	4.4333 (3)	**0.012264**	4.4000 (3)	**0.030666**
TriTr(C45)	5.1000 (7)	**0.000887**	5.0833 (6)	**0.002284**
DemCo	4.9333 (5)	**0.001655**	4.5833 (4)	**0.019925**
CoBag(C45)	5.6333 (8)	**0.000025**	5.2000 (8)	**0.001574**
SEGSSC-TriTr(C45)	3.5667 (2)	0.170587	3.9000 (2)	0.102292
SEGSSC-DemCo	4.6000 (4)	**0.007989**	4.8000 (5)	**0.008946**
SEGSSC-CoBag(C45)	5.0333 (6)	**0.001124**	5.1667 (7)	**0.001657**
$p_{Friedman}$	**0.000057**		**0.002493**	
	Inductive			
	10%		20%	
	Rank	p_{Holm}	Rank	p_{Holm}
Lower	3.0333 (1)	-	2.9000 (1)	-
CoTr(SMO)	5.5000 (8)	**0.000673**	4.6000 (5)	**0.028758**
TriTr(C45)	5.2333 (5)	**0.002017**	5.4333 (7)	**0.000371**
DemCo	5.4667 (7)	**0.000716**	5.2833 (6)	**0.000822**
CoBag(C45)	5.4000 (6)	**0.000913**	5.7333 (8)	**0.000052**
SEGSSC-TriTr(C45)	3.4667 (2)	0.858391	3.4667 (2)	0.370264
SEGSSC-DemCo	3.5333 (3)	0.858391	4.3167 (4)	0.075283
SEGSSC-CoBag(C45)	4.3667 (4)	0.105045	4.2667 (3)	0.075283
$p_{Friedman}$	**0.00004**		**0.00002**	
	30%		40%	
	Rank	p_{Holm}	Rank	p_{Holm}
Lower	3.3167 (1)	-	2.8167 (1)	-
CoTr(SMO)	4.1000 (3)	0.431018	4.6500 (4)	**0.011239**
TriTr(C45)	5.1500 (7)	**0.022479**	4.9333 (6)	**0.004088**
DemCo	4.3667 (4)	0.290625	4.7333 (5)	**0.009765**
CoBag(C45)	5.7333 (8)	**0.000930**	5.6000 (8)	**0.000075**
SEGSSC-TriTr(C45)	3.8000 (2)	0.444738	3.7667 (2)	0.133076
SEGSSC-DemCo	4.5333 (5)	0.290625	4.5333 (3)	**0.013284**
SEGSSC-CoBag(C45)	5.0000 (6)	**0.038887**	4.9667 (7)	**0.013284**
$p_{Friedman}$	**0.003332**		**0.000664**	

or CoBagging) and our method clearly does not interact well with these proposals (see Sect. 5.2.2). We therefore decided to retain our classifier in its clean, basic form. Its classification procedure is easy to understand and plainly performs well.

In conclusion, the benefit of using our OWA based fuzzy rough lower approximation classifier proposed in Sect. 3.3 on semi-supervised data is apparent. In fact, comparing the results in Table 5.6 to the ones reported in Table 5.2, we observe that the CoTr(SMO), TriTr(C45) and CoBag(C45) methods do not clearly outperform their C4.5 and SMO base classifiers. Only when integrated within the SEGSSC framework is their performance sufficiently lifted. However, the latter operation comes at a cost due to the synthetic element generation steps that rely on differential evolution [350]. We have shown that our method is already sufficiently strong when using only the originally labelled training data in its class predictions.

5.2.4 Discussion

The above experimental results have shown the strong performance of our OWA fuzzy rough set based classifier from Chap. 3 relative to existing semi-supervised classification methods. However, it is important to note that we have only included semi-supervised classifiers performing self-labelling in the comparison in Sect. 5.2.3. Alternative approaches (see Sect. 5.1.2) are not represented. The reason for this is that our 'Lower' method is in fact not a true semi-supervised classifier, in the sense that it does not use the elements in U to derive its class predictions. We have shown in Sect. 5.2.2 that our fuzzy rough set based method does not benefit from self-labelling in the classification of semi-supervised data. We only wished to verify whether its performance consequently sits below that of algorithms that do use self-labelling to their advantage. This is not the case.

We do not claim that there exists no possibility other than self-labelling to further enhance the performance of our fuzzy rough classifier on semi-supervised data by exploiting the information in U in its prediction mechanism. One possible area of future research lies with the modification of the definitions of the fuzzy rough approximation operators (3.7-3.8) according to the presence of unlabelled training instances. In their present form, the lower approximation of a class C relies on labelled training instances not in C and the upper approximation of C relies on labelled training instances in C. In other words, the former relies on elements in $L \cap co(C)$ and the latter on elements in $L \cap C$. On a semi-supervised dataset, these sets could be extended with certain elements in U, for instance by including unlabelled elements near x when computing $\underline{C}(x)$ and $\overline{C}(x)$ and weighing their contribution according to how strongly they relate to $co(C)$ or C respectively. Inspiration for the latter could be found with the techniques listed in Sect. 5.1.2. This approach stays close to self-labelling, which iteratively adds elements to L and thereby naturally extends the sets $L \cap co(C)$ and $L \cap C$, but differs from it by (i) only performing one 'labelling' iteration and (ii) using an adaptive set of instances in U for the prediction of each element x (namely, only those sufficiently near x).

Table 5.8 Results of the Wilcoxon test comparing our fuzzy rough classifier 'Lower' to the other algorithms in Table 5.6 in the format 'R^+/R^-/p'. The R^+ value always corresponds to the fuzzy rough method. P-values implying statistically significant differences at the 5% significance level are printed in boldface

	Transductive	
	10%	20%
CoTr(SMO)	399.0/66.0/**0.000593**	386.5/78.5/**0.001443**
TriTr(C45)	369.0/96.0/**0.004834**	376.0/89.0/**0.003058**
DemCo	372.0/93.0/**0.003982**	360.0/75.0/**0.001987**
CoBag(C45)	386.0/79.0/**0.001537**	388.0/77.0/**0.001334**
SEGSSC-TriTr(C45)	316.0/149.0/0.084035	342.0/123.0/**0.023665**
SEGSSC-DemCo	287.0/178.0/0.257946	345.5/119.5/**0.019257**
SEGSSC-CoBag(C45)	285.0/180.0/0.275659	339.0/126.0/**0.027749**
	30%	40%
CoTr(SMO)	373.0/92.0/**0.00373**	363.0/102.0/**0.00705**
TriTr(C45)	386.0/79.0/**0.001537**	363.0/72.0/**0.001594**
DemCo	372.0/93.0/**0.003982**	374.0/91.0/**0.003492**
CoBag(C45)	390.0/75.0/**0.001155**	387.0/78.0/**0.001432**
SEGSSC-TriTr(C45)	316.0/149.0/0.084035	338.0/127.0/**0.029239**
SEGSSC-DemCo	287.0/178.0/0.257946	355.5/79.5/**0.002674**
SEGSSC-CoBag(C45)	285.0/180.0/0.275659	368.0/97.0/**0.005153**
	Inductive	
	10%	20%
CoTr(SMO)	394.0/71.0/**0.000862**	368.0/97.0/**0.005153**
TriTr(C45)	380.0/85.0/**0.002274**	387.0/78.0/**0.001432**
DemCo	377.0/88.0/**0.002789**	375.0/90.0/**0.003269**
CoBag(C45)	389.5/75.5/**0.001163**	397.0/68.0/**0.000689**
SEGSSC-TriTr(C45)	298.0/167.0/0.174619	329.0/136.0/**0.046029**
SEGSSC-DemCo	297.0/168.0/0.181242	340.5/124.5/**0.025259**
SEGSSC-CoBag(C45)	281.0/184.0/0.312281	325.0/140.0/0.055767
	30%	40%
CoTr(SMO)	347.0/118.0/**0.018013**	340.0/95.0/**0.007822**
TriTr(C45)	347.5/87.5/**0.004662**	376.0/89.0/**0.003058**
DemCo	324.5/140.5/0.056459	362.5/102.5/**0.007122**
CoBag(C45)	388.0/77.0/**0.001334**	387.0/78.0/**0.001432**
SEGSSC-TriTr(C45)	320.0/145.0/0.070294	349.0/116.0/**0.016106**
SEGSSC-DemCo	359.0/106.0/**0.008821**	370.0/95.0/**0.004534**
SEGSSC-CoBag(C45)	352.0/113.0/**0.013549**	373.0/92.0/**0.00373**

Two other components are present in definitions (3.7-3.8), namely the instance similarity relation $R(x, \cdot)$ and the OWA weight vectors W_L and W_U. The former could be replaced by an alternative directly incorporating the information in U for example in a similar way as semi-supervised support vector machine classifiers use the unlabelled training elements. As described in Sect. 5.1.2, these methods avoid letting the decision boundary cross dense regions of unlabelled elements. An option for our fuzzy rough set based methods would be to penalize the similarity between labelled instances separated by a dense unlabelled region. Very informally put, this corresponds to considering two people standing on either bank of a 50 meter wide river as more distant to each other as when they were standing on either end of a 50 meter long meadow. It is more difficult to cross the former than the latter. Another option is the integration of the training set label sparsity in the definitions of W_L and W_U. The weight definitions discussed in Chap. 3 could be updated to accommodate both labelled and unlabelled elements, for instance by interweaving weight vectors of different schemes with each other (e.g. W_L^{add} for L and W_L^{exp} for U).

5.3 Conclusion

In this chapter, we have studied the topic of semi-supervised classification. In a semi-supervised training set, only part of the observations are associated with a class label, while the remainder is unlabelled. The classification task can be split up in two aspects, the transductive and inductive performances. The former refers to the prediction of class labels for unlabelled training elements, while the latter evaluates the predictions for unseen test elements.

We have evaluated our fuzzy rough classifiers proposed in Chap. 3 for this task using a set of 30 semi-supervised datasets with 10, 20, 30 and 40% of labelled training instances (that is, 30 datasets for each of these percentages). These are fairly basic and easy-to-understand algorithms: to classify an instance x, they compute its membership degree to the OWA based fuzzy rough lower or upper approximation of all decision classes and assign x to the class for which this value is largest. The weighting schemes used within the lower and upper approximation calculations are chosen according to our strategy proposed in Chap. 3.

In a first step of our evaluation, we were able to show that our fuzzy rough classifiers retain a strong performance even when only a small portion of the training set is labelled. Using the relatively limited amount of information available in the labelled part of the training set, our methods clearly outperform other base classifiers used in the experimental study of [406]. Only small performance differences are observed between our methods using the lower approximation, upper approximation or both and we decided to focus on the former. Secondly, we combined this OWA based fuzzy rough lower approximation classifier with existing strong self-labelling techniques and showed that its performance is not improved by extending the labelled part of the training set. Instead, statistically significant decreases in performance were observed. We concluded that the most prudent course of action is to only allow

our fuzzy rough method to use the originally labelled training instances to derive its predictions. In the final part of our experiments, we compared the classification performance of our classifier to existing semi-supervised classifiers that do rely on self-labelling. We selected well-performing methods from recent studies [405, 406]. Our method outperforms all included algorithms and often so with statistical significance.

The reader will recognize that this has been an atypical chapter, in which we showed that our existing algorithm from Chap. 3 already performed very well and is only hindered by making modifications for this specific classification setting. As should be clear from our discussion in Chap. 1, the difference between supervised and semi-supervised data is of a different order than the difference between traditional data and multi-instance or multi-label data (Chaps. 6 and 7). Semi-supervised data does not imply a new structure of the observations, only a smaller labelled training set. As we will discuss in our overall conclusion in Chap. 8, fuzzy rough set based methods are particularly suited for small classification problems, that is, problems with small to moderately-sized training sets. Aside from this aspect, our optimized weighting scheme selection strategy from Chap. 3 has allowed our OWA based lower approximation predictor to use the limited information in the training set to its full advantage. As discussed in Sect. 5.2.2, extending the labelled part of the training set by means of self-labelling can result in a relatively more challenging training set to learn from (although containing more labelled data) and the adverse effect is clear on our fuzzy rough classifier due to its strong dependence on instance similarity values.

Our main conclusion from this study is twofold, namely (i) our OWA based fuzzy rough lower approximation classifier proposed in Chap. 3 performs strongly even on semi-supervised data and (ii) the method does not benefit from any self-labelling. We acknowledge that the second conclusion has been derived using only (variants of) established self-labelling methods reviewed in [406]. However, while studying the challenge of semi-supervised classification, we did test many other ideas, both based on fuzzy rough set theory and not, but no improvements over the baseline performance of our method could be obtained. As shown in the experimental study, the performance of the naive fuzzy rough self-labelling method proposed in [311] is far below that of the other methods reported in this chapter as well. The inherent characteristic of self-labelling remains that these methods can intuitively be expected to create several dense areas of same-class instances (in which confident class predictions were made) separated by sparser regions (where prediction confusion is present). The resulting self-labelled training sets do not lend themselves as appropriate training data for our fuzzy rough classification algorithm in its current form. Lessons learnt from the development of so-called *safe* semi-supervised classifiers (e.g. [283, 285]), which is an active topic of current research that ensures that including the unlabelled instances in the prediction process causes no harm, can further assist us in the study of a true semi-supervised fuzzy rough set based classification method.

The research question has been whether our fuzzy rough classifier benefits from self-labelling. We have shown that it does not, but wish to stress that this conclusion should not be regarded as a negative result. Instead, we have once again confirmed the strength of our weighting scheme selection strategy proposed in Chap. 3. Its

strong performance compared to the existing semi-supervised classifiers as evaluated in Sect. 5.2.3 furthermore suggests an efficiency gain for the classification of this type of data. Our classifier does not require a self-labelling step, which is usually an iterative procedure in which many classifiers are constructed, and only uses the small labelled training set in its predictions. As a consequence, the classification procedure is both swift and relatively accurate.

Chapter 6
Multi-instance Learning

The domain of multi-instance learning (MIL) deals with datasets consisting of compound data samples. Instead of representing an observation as an instance described by a single feature vector, each observation (called a bag) corresponds to a set of instances and, consequently, a set of feature vectors. The instances within a bag can represent different parts or alternative representations of the same object. Initially proposed in [127], the MIL domain has developed into a mature learning paradigm with many real-world applications. A comprehensive overview can be found in the recent book [217].

In this chapter, we propose multi-instance classifiers based on fuzzy and fuzzy rough set theory. Our methods classify unseen bags using either instance-level or bag-level information. We first provide an introduction to MIL in general in Sect. 6.1 and to multi-instance classification in Sect. 6.2. Our fuzzy multi-instance classifiers are described in Sect. 6.3, while Sect. 6.5 defines our fuzzy rough multi-instance methods developed for class imbalanced multi-instance data. These two sections provide a complete overview of our algorithms and their proposed internal parameter settings.

We conduct a thorough experimental validation of our proposals. A high number of experimental results are included and, for the sake of clarity, we divide the experimental study into two main parts. Sections 6.4 and 6.6 compare the different internal settings of our fuzzy and fuzzy rough set methods respectively and explain why certain choices for the given parameters are more appropriate than others. We report these after the sections introducing our frameworks, such that Sects. 6.3 and 6.4 both consider our fuzzy methods, while Sects. 6.5 and 6.6 are on the fuzzy rough set based classifiers. In Sect. 6.7, with lessons learnt from Sects. 6.4 and 6.6, we compare our methods to existing multi-instance classifiers on both balanced and imbalanced multi-instance datasets. Finally, Sect. 6.8 sums up our conclusions.

We have aimed to retain a clear structure in the large amount of material included in this chapter, which consists of four parts (followed by the conclusion in Sect. 6.8), summarized as follows:

S. Vluymans, *Dealing with Imbalanced and Weakly Labelled Data in Machine Learning using Fuzzy and Rough Set Methods*, Studies in Computational Intelligence 807, https://doi.org/10.1007/978-3-030-04663-7_6

1. Sections 6.1 and 6.2 serve as introduction to the topic of multi-instance classification.
2. Sections 6.3 and 6.4 introduce our framework of multi-instance classifiers based on fuzzy set theory. The former describes the classification process of our methods in great detail, including illustrative examples and a discussion on their computational complexity. The latter performs an internal experimental comparison of these methods to gain a better insight into their behaviour.
3. Sections 6.5 and 6.6 considers our fuzzy rough set based multi-instance classifier framework. As before, the former chapter provides a detailed description of our methods and the latter presents a thorough experimental comparison of their parameter settings.
4. Section 6.7 presents the concluding part of the experimental study conducted in this chapter. Based on the observations made in Sects. 6.4 and 6.6, we compare our proposed algorithms to existing multi-instance classifiers.

6.1 Introduction to Multi-instance Learning

We first provide a brief introduction to the domain of MIL. Its origins are presented in Sect. 6.1.1, where we discuss the original proposal [127] by Dietterich et al. initiating research in this field. Having provided the intuition behind multi-instance data, its formal description is given in Sect. 6.1.2. Several prominent application areas are listed in Sect. 6.1.3. This section (as well as Sect. 6.2) is based on the recent book [217] and we refer the interested reader to that work for a more detailed survey of the field.

6.1.1 Origin

The seminal paper by Dietterich et al. [127] introduced and formalized the concept of multi-instance data. The authors provided the toy problem of staff member key chains. Every member has a number of keys, one of which opens the supply room of the department. However, different supply room keys have been handed out. Some of them only open the actual supply room, while others can be used for other rooms (e.g. the cafeteria) as well. The task of a lock smith would be to deduce the shape of the key required to open the supply room door based on the key chains of all staff members without them actually showing which one of their keys opens this door. The lock smith therefore only knows that one of the keys per chain opens the required door, but not which one.

Section 1.1.3 already recalled the second (more realistic) multi-instance data application presented in [127], namely the drug activity prediction task. This biochemistry application aims to ascertain which drug (typically a small molecule) binds to a certain target, thereby producing the desired biological or medical effect. Since

a molecule is made up of several atoms, it can appear in different shapes (called conformations or molecular structures) due to internal rotations. Its binding ability and strength can depend on the particular shape. These alternative representations result in a more complex data format, where each observation (molecule) corresponds to several instances (conformations). When at least one of its conformations binds to the target, the molecule is considered a good drug molecule. On the other hand, when none of its conformations result in a binding, it belongs to the negative class and should not be used in this particular drug synthesis process.

Multi-instance learning is a generalization of traditional (single-instance) learning. Whereas each observation could traditionally be represented by a single feature vector, a multi-instance observation corresponds to a bag of instance vectors. In the examples above, an observation corresponds to a key chain containing several keys or a drug molecule with different conformations. The increased complexity in the data format, which we formally specify in the next section, requires custom learning algorithms to extract the information contained in it. By formulating the above problems in this manner, the field of MIL was born.

6.1.2 Structure of Multi-instance Data

An example multi-instance dataset was presented in Table 1.4. In the present context, an observation is referred to as a *bag* containing a number of *instances*. Each instance x can be represented as a d-dimensional feature vector, in which the ith position corresponds to the value of x for the ith descriptive feature. A bag B contains n_B instances. As the index indicates, the size of each bag in the dataset can be different. Furthermore, duplicate instances within the same bag are allowed and different bags can contain copies of the same instance(s) as well. In supervised multi-instance learning, the bag is associated with a class label. We should stress that this label is only known for the bag as a whole and not for its constituent instances. Note that, as is common in the MIL literature, we use lower-case letters to refer to instances and upper-case letters to refer to bags.

The general structure of a labelled multi-instance dataset is presented in Table 6.1. The dataset contains M bags B_1 to B_M. Each bag B_i in turn consists of n_{B_i} instances x_{i1} to $x_{in_{B_i}}$. Each of these instances is described by its values for the d features. Value $a_k(x_{ij})$ corresponds to the kth feature value of instance x_{ij} $(k = 1, \ldots, d)$. The outcome y_i associated with a bag B_i can be discrete (a class label) or continuous (regression outcome).

6.1.3 Application Areas

The multi-instance data format presented in Table 6.1 is naturally encountered in a variety of applications. In general, we can list the following sources of multi-instance data with an intrinsic ambiguity in observation representation.

Table 6.1 General structure of a multi-instance dataset with M labelled bags and descriptive features $a_1, a_2, \ldots, a_d$

Bag	a_1	a_2	…	a_d	Outcome
B_1	$a_1(x_{11})$	$a_2(x_{11})$	…	$a_d(x_{11})$	y_1
	$a_1(x_{12})$	$a_2(x_{12})$	…	$a_d(x_{12})$	
	…	…	…	…	
	$a_1(x_{1n_{B_1}})$	$a_2(x_{1n_{B_1}})$	…	$a_d(x_{1n_{B_1}})$	
B_1	$a_1(x_{21})$	$a_2(x_{21})$	…	$a_d(x_{21})$	y_2
	$a_1(x_{22})$	$a_2(x_{22})$	…	$a_d(x_{22})$	
	…	…	…	…	
	$a_1(x_{2n_{B_2}})$	$a_2(x_{2n_{B_2}})$	…	$a_d(x_{2n_{B_2}})$	
…	…	…	…	…	…
B_M	$a_1(x_{M1})$	$a_2(x_{M1})$	…	$a_d(x_{M1})$	y_M
	$a_1(x_{M2})$	$a_2(x_{M2})$	…	$a_d(x_{M2})$	
	…	…	…	…	
	$a_1(x_{Mn_{B_M}})$	$a_2(x_{Mn_{B_M}})$	…	$a_d(x_{Mn_{B_M}})$	

- **Alternative representations**: as described in Sect. 6.1.1, the drug activity prediction task considers molecular data, where each bag corresponds to a set of conformations of the same molecule. This fits in this first group of multi-instance data sources, where a number of different descriptions or views of the same object are included.
- **Compound objects**: when an observation consists of several integral parts, each of them can be represented by an instance in the bag corresponding to the observation. A traditional example is the image classification task, where an image (bag) is divided into several regions of interest (instances) by means of an image segmentation technique. The example image in Fig. 6.1 would be labelled as a 'full English breakfast'. Its constituent instances are the image segments corresponding to the breakfast ingredients (poached egg, sausage, beans, cup of tea and so on).
- **Evolving objects**: the third source of multi-instance data corresponds to time-series problems, where the same object is recorded at several time steps. The bag corresponds to the monitored object and the instances to the recording at each time point. An example dataset is one consisting of medical measurements of a patient on each day of a hospital stay.

Application areas where the above sources of multi-instance data are encountered include bioinformatics (e.g. [5, 56, 127, 166, 274, 282, 326, 456, 500]), image retrieval and classification (e.g. [71, 94, 143, 220, 256, 318, 351]), web mining and text classification (e.g. [369, 478, 479, 505]), object detection and tracking (e.g. [21, 434, 484]), medical diagnosis and imaging (e.g. [134, 348, 388, 436]) and acoustic event detection (e.g. [268]). The study of [12] stressed the possibly

Fig. 6.1 A full English breakfast

considerable differences in the multi-instance characteristic of different application areas, even when they all yield datasets in which observations are represented as bags of instances.

6.2 Multi-instance Classification

In multi-instance classification, the goal is to predict the class label of a previously unseen bag based on a classification model learned from a training set of labelled bags. The difference with traditional single-instance classification (see Chap. 2) lies with the data format used in the learning process. As described in Sect. 6.1.2, a bag is a collection of instances. The learner needs to handle these compound objects and derive sufficient information from them.

In Sect. 6.2.1, we first discuss a number of existing multi-instance hypotheses that decide when a bag is considered positive in two-class multi-instance problems with one positive and one negative class. Section 6.2.2 recalls a taxonomy of multi-instance classifiers that divides these algorithms based on how they approach the multi-instance nature of their training data. Several example methods are listed as well. Section 6.2.3 defines imbalanced multi-instance classification.

Aside from classification, multi-instance regression and unsupervised MIL exist as learning paradigms as well [217]. One multi-instance task that is not present in single-instance learning is instance annotation. In this setting, aside from the prediction of the bag label, a class is derived for each instance as well. The set of possible instance classes is not required to coincide with the set of possible bag classes.

6.2.1 Multi-instance Hypotheses

The most common and traditional setting within multi-instance classification is that of binary class prediction with one *positive* and one *negative* class. Multi-instance hypotheses or assumptions indicate when a bag is considered positive based on the instances it contains. They represent the link between the bag label and the constituent instances. In particular, the multi-instance hypothesis maps a set of instance labels to a bag label. This relation is often implicit in the development of a multi-instance classifier.

The so-called *standard multi-instance assumption* relates back to the original introduction of multi-instance classification in [127]. It states that a bag is positive if and only if it contains at least one positive instance. This interpretation suits the drug activity prediction task, where a bag (molecule) is positive when at least one of its instances (molecular conformations) binds to the target. The presence of one positive instance directly determines the bag label. When a bag contains only negative instances, it is labelled as negative. This instance-to-bag relation may not hold in all binary classification application areas. Other, more general, multi-instance hypotheses have therefore been introduced as well [159].

The *presence-based, threshold-based and count-based multi-instance assumptions* have been proposed in [444]. They are based on the general setting where instance labels and bag labels (positive, negative) do not need to coincide. Instances can belong to various concepts (classes) other than 'positive' or 'negative'. For a bag to be positive, instances from several concepts need to be present. The presence-based assumption expects the presence of at least one instance of every required concept, while the threshold-based assumption states that the number of instances of each of these concepts should exceed a corresponding threshold. The count-based assumption of [444] uses a lower and upper threshold on the number of instances of each required concept.

The *collective multi-instance hypothesis* [160, 454] assumes that each instance contributes equally to the label of a bag. The bag class probability function is based on the instance likelihoods of belonging to or being associated with the bag class labels. Later extensions of this setting include the *weighted collective assumption* [157] and the *product rule and sum rule assumptions* [284]. The study of [284] also considered the *mixture distribution assumption* (including product rule, sum rule and γ-rule assumptions), which models the mixture distribution of instances across a concept and a non-concept. Finally, the *soft bag assumption* [281] generalizes the

standard multi-instance assumption and allows for both positive and negative bags to be soft, meaning that a negative bag can contain some positive instances and a positive bag can contain some negative instances.

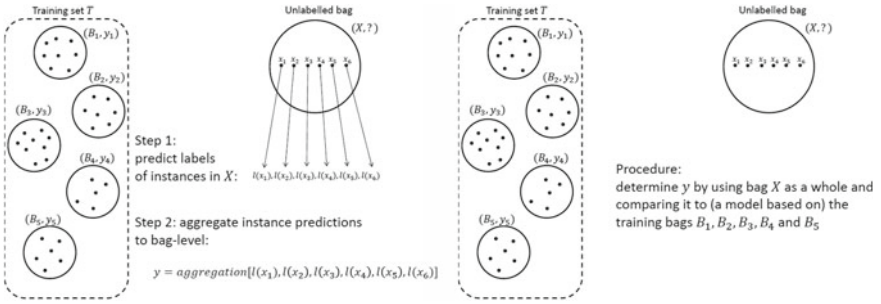
6.2.2 *Taxonomy of Multi-instance Classifiers*

A taxonomy of multi-instance classification methods has been proposed in [13]. Based on their internal approach to the multi-instance data problem, the algorithms are divided into three groups.

1. **Instance space paradigm**: the multi-instance classifier relies on information at the level of instances. The classification model is derived in such a way that it optimally (according to some criterion) separates instances belonging to bags of different classes. The class label of a bag is determined based on the derived labels of its instances, which correlates with a multi-instance hypothesis mapping. Example multi-instance classifiers within this paradigm are the miSVM method [17], the axis-parallel rectangle method [127], the MIWrapper method [160] and the diverse density method [317].
2. **Bag space paradigm**: a bag space classifier considers the bags as whole entities, the true union of their instances. These methods commonly rely on distance or similarity comparisons between bags. The class discriminant function is determined at the level of the bags. Multi-instance classifiers embodying this idea include the CitationKNN [433] and MI-Graph [509] methods.
3. **Embedded space paradigm**: these algorithms transform the multi-instance data format to a single-instance representation. A mapping is defined to transform each bag to a single feature vector. In this induced space, any existing single-instance classifier can be applied. Methods adhering to this paradigm include MILES [93], DD-SVM [94], the SimpleMI [130], YARDS [157] and BARTMIP [487] methods.

Review work [217] further divides each group of methods based on the traditional single-instance classification ideas they incorporate or extend, such as support vector machines or neural networks. We can also refer to the study of [99] for a more general taxonomy depending on whether bags of instances are present in the learning phase, testing phase, both or not at all.

In the remainder of this chapter, we introduce multi-instance classifiers adhering to the first and second paradigms. Figure 6.2 illustrates the difference between these two approaches. To predict the label y of a new bag X, instance-based classifiers (Fig. 6.2a) first derive class predictions $l(x)$ for all instances $x \in X$ and then aggregate these values to a class label for the whole bag. Bag-based classifiers (Fig. 6.2b) process bag X as a whole and predict its label based on its similarity with the training bags.



(a) Instance space paradigm (b) Bag space paradigm

Fig. 6.2 Illustration of the instance space and bag space paradigms in multi-instance classification

6.2.3 Imbalanced Multi-instance Classification

Imbalanced multi-instance data is defined in a similar way as done in Sect. 1.1.1 and Chap. 4 for single-instance datasets. The imbalance refers to the class label, namely that more observations are present for some classes than for others and that this discrepancy may be large. Multi-instance class imbalance is consequently defined at the level of bags. Compared to the research output on single-instance imbalanced classification, relatively little work has been done in the area of multi-instance class imbalance. Nonetheless, as the problem is for both settings defined at the class level, existing single-instance solutions to class imbalance could often be adapted to deal with the multi-instance version in a straightforward way. Examples include the methods proposed in [321, 322, 439, 440] that consider preprocessing techniques to obtain a better balance between the classes as well as a set of multi-instance cost-sensitive boosting algorithms. In Sect. 6.5, we address the class imbalance problem in multi-instance classification by means of our fuzzy rough set based methods that extend the binary IFROWANN method (see Sect. 4.1.3).

6.3 Fuzzy Multi-instance Classifiers

We develop two families of fuzzy set based multi-instance classifiers. They adhere to the instance space paradigm (instance-based fuzzy multi-instance classifiers, IFMIC) and the bag space paradigm (bag-based fuzzy multi-instance classifiers, BFMIC) in the taxonomy of [13]. Each class is regarded as a fuzzy set to which every bag has a certain membership degree. When classifying an unseen bag, our methods calculate its membership degree to each class and assign it to the class for which this value is largest. Section 6.3.1 introduces our proposed classifiers. In particular, we discuss the IFMIC family in Sect. 6.3.1.1 and the BFMIC family in Sect. 6.3.1.2. We present an

overview of the framework in Sect. 6.3.2, followed by some examples in Sect. 6.3.3 for illustration purposes. We conclude with a derivation of the theoretical complexity of our proposals in Sect. 6.3.4.

6.3.1 Proposed Classifiers

We define a fuzzy multi-instance classifier as a mapping

$$f : \mathbb{N}^{\mathcal{X}} \to \mathcal{C} : X \mapsto \arg\max_{C \in \mathcal{C}} [C(X)], \tag{6.1}$$

with $\mathcal{X}$ the instance space, $\mathbb{N}^{\mathcal{X}}$ the bag space (defined as multi-sets of instances) and $\mathcal{C}$ the set of possible classes. The bag X is assigned to the class C for which its membership degree $C(X)$ is largest. In case of a tie for the maximum value, one of the tied classes is randomly selected. We consider two approaches to derive the $C(X)$ values:

- Instance-based fuzzy multi-instance classifiers (IFMIC family, Sect. 6.3.1.1): value $C(X)$ is derived from the class membership degree $C(x)$ of instances x in bag X.
- Bag-based fuzzy multi-instance classifiers (BFMIC family, Sect. 6.3.1.2): value $C(X)$ is derived directly from bag information. In particular, it relies strongly on the similarity between X and the training bags.

Both classifier families require the definition of class membership degrees, either for instances or bags. These values are determined based on the training data and we introduce ways to do so in the sections below.

6.3.1.1 The IFMIC Family

Our instance-based methods derive the membership degree $C(X)$ in (6.1) from the membership degree of instances to class C. We therefore need to specify how to calculate the membership degree $C(x)$ of an instance x to a class C. Prior to this, we introduce different ways to determine the membership degree (or, perhaps more appropriately, the degree of affinity) $B(x)$ of instance x to training bag B. The calculation of $C(x)$ relies on the $B(x)$ values for training bags B belonging to class C. Interpreting bags B as fuzzy sets allows us to express how typical x is for this bag, that is, how strongly it relates to it. This differs from merely determining whether an instance belongs to a bag or not. We now discuss the different components of our IFMIC classifiers in their calculation of the $C(X)$ values. These are summarized in Table 6.2.

Membership $B(x)$ of instances to bags The definitions of the bag affinity values rely on an instance similarity relation $R_I(\cdot, \cdot)$. This fuzzy relation measures the degree of similarity between pairs of instances. Due to the high-dimensional nature

of many multi-instance datasets, we decide to work with two different relations. When the number of features is at most 20, $R_I(\cdot,\cdot)$ coincides with our standard instance similarity relation (3.13). When the number of features exceeds this threshold, we use the cosine similarity relation

$$R_I(x_1, x_2) = \frac{x_1 \cdot x_2}{||x_1|| \cdot ||x_2||} \tag{6.2}$$

and rescale this value to the unit interval. The cosine similarity has been shown to be the most appropriate similarity measure to use in high-dimensional multi-instance datasets related to text processing and image retrieval (e.g. [369]), examples of which are included in our experiments. The cut-off value of 20 features is based on the experimental study of [6].

Given relation $R_I(\cdot,\cdot)$, we can define $B(x)$ in several ways. Table 6.2 lists the five possibilities evaluated in this chapter. They represent several intuitive choices that one can make. A first option is to set $B(x)$ to the maximum instance similarity of x to one of the instances $b \in B$ (Max). Alternatively, we can compute $B(x)$ as the average of the $R_I(x, b)$ values (Avg). We also include three OWA aggregations whose actions place them between taking the maximum and average. We aggregate the $R_I(x, b)$ values (for $b \in B$) with W_U^{exp} (MaxExp), W_U^{invadd} (MaxInvadd) or W_U^{add} (MaxAdd). As shown in Sect. 3.2, these weight vectors correspond to softened maxima.

Membership $C(x)$ of instances to classes Having determined the values $B(x)$ for all training bags B, we consider the same five ways to aggregate these values to $C(x)$ membership degrees. We define the set T_C as the set containing all training bags belonging to class C. For each bag $B \in T_C$, the value $B(x)$ can be computed. Based on these results, $C(x)$ is determined as their maximum (Max), average (Avg) or OWA aggregation (MaxExp, MaxInvadd or MaxAdd).

Membership $C(X)$ of bags to classes To finally determine the membership degree $C(X)$ of bag X to class C in (6.1), the IFMIC methods aggregate the instance membership degrees $C(x)$ for $x \in X$ to one value using the same five alternatives as listed for the $B(x)$ and $C(x)$ calculations. In our experiments, we evaluate the use of the maximum (Max), softened maxima (MaxExp, MaxInvadd or MaxAdd) or the average (Avg).

6.3.1.2 The BFMIC Family

The bag-based definition of our fuzzy multi-instance classifiers relies on a bag similarity measure $R(\cdot,\cdot)$. The class membership degrees $C(X)$ are based on the similarity between X and training bags B of class C. All settings for the bag similarity relation and the class membership degree aggregation evaluated in this chapter are summarized in Table 6.3.

Bag-wise similarity $R(X, B)$ The similarity between bags X and B is based on the distance $\delta(\cdot,\cdot)$ between pairs of their instances, which we define based on the

Table 6.2 Settings for the IFMIC methods evaluated within our proposed framework. Set T_C contains all training bags belonging to class C

Code	$B(x)$	Code	$C(x)$
Max	$\max_{b\in B} R_I(x,b)$	Max	$\max_{B\in T_C} B(x)$
MaxExp	$\mathrm{OWA}_{W_U^{exp}}(\{R_I(x,b)\mid b\in B\})$	MaxExp	$\mathrm{OWA}_{W_U^{exp}}(\{B(x)\mid B\in T_C\})$
MaxInvadd	$\mathrm{OWA}_{W_U^{invadd}}(\{R_I(x,b)\mid b\in B\})$	MaxInvadd	$\mathrm{OWA}_{W_U^{invadd}}(\{B(x)\mid B\in T_C\})$
MaxAdd	$\mathrm{OWA}_{W_U^{add}}(\{R_I(x,b)\mid b\in B\})$	MaxAdd	$\mathrm{OWA}_{W_U^{add}}(\{B(x)\mid B\in T_C\})$
Avg	$\frac{1}{\lvert B\rvert}\sum_{b\in B} R_I(x,b)$	Avg	$\frac{1}{\lvert T_C\rvert}\sum_{B\in T_C} B(x)$

Code	$C(X)$
Max	$\max_{x\in X} C(x)$
MaxExp	$\mathrm{OWA}_{W_U^{exp}}(\{C(x)\mid x\in X\})$
MaxInvadd	$\mathrm{OWA}_{W_U^{invadd}}(\{C(x)\mid x\in X\})$
MaxAdd	$\mathrm{OWA}_{W_U^{add}}(\{C(x)\mid x\in X\})$
Avg	$\frac{1}{\lvert X\rvert}\sum_{x\in X} C(x)$

instance similarity relation. In particular,

$$(\forall x\in X)(\forall b\in B)(\delta(x,b)=1-R_I(x,b)).$$

We consider eight bag similarity functions, which can be divided into two groups of four. The first group (consisting of H, HExp, HInvadd and HAdd) is based on the Hausdorff distance [135] between two bags. The first option is to set the similarity between two bags to the complement to one of this distance measure (H). The three other versions replace the maximum and minimum operators in this definition by OWA operators using exponential (HExp), inverse additive (HInvadd) or additive (HAdd) weights. The second group is based on the average Hausdorff distance [487]. The bag similarity based on this measure is given by AvgH. As above, we replace the maximum and minimum operators by OWA aggregations in the three modified versions AvgHExp, AvgHInvadd and AvgHAdd.

Membership $C(X)$ of bags to classes The aggregation of the $R(X,\cdot)$ values to a $C(X)$ membership degree is handled by one of the same five alternatives as used in our IFMIC methods. We consider the use of the average similarity (Avg) as well as the maximum similarity (Max) of X with a bag B in class C. We also include the softened maxima alternatives MaxExp, MaxInvadd and MaxAdd.

We note that our proposed bag-based classifiers exhibit a link with a nearest neighbour approach to multi-instance classification. In case of version Max for the $C(X)$ calculations, the method computes the membership degrees of a test bag to the decision classes based on its most similar (nearest) training bag of each class. Since

Table 6.3 Settings for the BFMIC methods evaluated within our proposed framework. Set T_C contains all training bags belonging to class C

Code	$R(X, B)$
H	$1 - \max(\max_{x \in X} \min_{b \in B} \delta(x, b), \max_{b \in B} \min_{x \in X} \delta(x, b))$
HExp	$1 - \max[\text{OWA}_{W_U^{exp}}(\{\text{OWA}_{W_L^{exp}}(\{\delta(x, b) \mid b \in B\}) \mid x \in X\}),$
	$\text{OWA}_{W_U^{exp}}(\{\text{OWA}_{W_L^{exp}}(\{\delta(x, b) \mid x \in X\}) \mid b \in B\})]$
HInvadd	$1 - \max[\text{OWA}_{W_U^{invadd}}(\{\text{OWA}_{W_L^{invadd}}(\{\delta(x, b) \mid b \in B\}) \mid x \in X\}),$
	$\text{OWA}_{W_U^{invadd}}(\{\text{OWA}_{W_L^{invadd}}(\{\delta(x, b) \mid x \in X\}) \mid b \in B\})]$
HAdd	$1 - \max[\text{OWA}_{W_U^{add}}(\{\text{OWA}_{W_L^{add}}(\{\delta(x, b) \mid b \in B\}) \mid x \in X\}),$
	$\text{OWA}_{W_U^{add}}(\{\text{OWA}_{W_L^{add}}(\{\delta(x, b) \mid x \in X\}) \mid b \in B\})]$
AvgH	$1 - \frac{1}{\lvert X\rvert+\lvert B\rvert}\left(\sum_{x \in X} \min_{b \in B} \delta(x, b) + \sum_{b \in B} \min_{x \in X} \delta(x, b)\right)$
AvgHExp	$1 - \frac{1}{\lvert X\rvert+\lvert B\rvert}\left(\sum_{x \in X} \text{OWA}_{W_L^{exp}}(\{\delta(x, b) \mid b \in B\}) + \sum_{b \in B} \text{OWA}_{W_L^{exp}}(\{\delta(x, b) \mid x \in X\})\right)$
AvgHInvadd	$1 - \frac{1}{\lvert X\rvert+\lvert B\rvert}\left(\sum_{x \in X} \text{OWA}_{W_L^{invadd}}(\{\delta(x, b) \mid b \in B\}) + \sum_{b \in B} \text{OWA}_{W_L^{invadd}}(\{\delta(x, b) \mid x \in X\})\right)$
AvgHAdd	$1 - \frac{1}{\lvert X\rvert+\lvert B\rvert}\left(\sum_{x \in X} \text{OWA}_{W_L^{add}}(\{\delta(x, b) \mid b \in B\}) + \sum_{b \in B} \text{OWA}_{W_L^{add}}(\{\delta(x, b) \mid x \in X\})\right)$
Code	$C(X)$
Max	$\max_{B \in T_C} R(X, B)$
MaxExp	$\text{OWA}_{W_U^{exp}}(\{R(X, B) \mid B \in T_C\})$
MaxInvadd	$\text{OWA}_{W_U^{invadd}}(\{R(X, B) \mid B \in T_C\})$
MaxAdd	$\text{OWA}_{W_U^{add}}(\{R(X, B) \mid B \in T_C\})$
Avg	$\frac{1}{\lvert T_C\rvert} \sum_{B \in T_C} R(X, B)$

the final class prediction is made using (6.1), the class label of the overall nearest bag is automatically predicted. Consequently, we can conclude that the use of Max reduces our BFMIC proposal to a one-nearest neighbour multi-instance classifier, where the distance measure is taken as the complement of the bag similarity $R(\cdot, \cdot)$. When one of the OWA based alternatives for $C(X)$ is selected instead, BFMIC takes a step away from the true nearest neighbour paradigm. For each class C, all training bags belonging to C contribute to the membership degree estimation of X to C. Their contribution not only depends on their similarity with the bag to classify, but also on the number of bags in C, since the class size determines the length of the OWA weight vector. As part of our experimental study, we show that our proposed methods outperform the most prominent nearest neighbour multi-instance classifier CitationKNN [433].

6.3.1.3 Discussion

From the descriptions given in the two preceding chapters and Tables 6.2 and 6.3, it should be clear that in various cases an aggregation step can be set to the maximum or average of a group of values. Both correspond to intuitive aggregation approaches. The strict maximum assigns weight one to a single value and weight zero to all others, while the average involves all values in its calculation, effectively assigning equal weight to all. We consider intermediate options by using the OWA based alternatives that model the trade-off between the maximum and average.

For the OWA weights, we use the weight vectors studied in Sect. 3.2.1. As discussed in Chap. 3, the selection of an OWA weight vector is not always straightforward. Several automatic procedures have been proposed for this purpose (see Sect. 3.2.2). Many of them are optimization methods, optimizing a certain objective function (e.g. entropy) for a user-specified $orness$ value. Within the scope of this study, fixing the $orness$ of the weight vector beforehand feels arbitrary and we therefore opt not to use such an optimization algorithm. We also avoid them in the interest of computational cost. For a given $orness$ value, existing optimization methods do not provide the user with a closed formula to determine the weights, but rather with a particular weight set of the specified length. As should be clear from the descriptions given in Tables 6.2 and 6.3, the vector lengths are not fixed in our methods. For instance, in the MaxAdd variant of $B(x)$ in Table 6.2 the length of the weight vector equals the size of bag B. Since all bags in a multi-instance dataset can contain a different number of instances, the length of all weight vectors can be different. The variable length implies the requirement of multiple runs of the optimization procedure, imposing a considerable cost. Furthermore, these methods become practically intractable when the vector lengths increase. The procedure proposed in [168], for instance, relies on the calculation of the roots of a polynomial equation with degree equal to the length of the weight vector. As shown in the experiments conducted later in this chapter, we can expect the vector lengths, and therefore the degree of these polynomial equations, to be high (100 and above).

We have decided to use the fixed OWA weight vectors recalled in Sect. 3.2.1, namely the exponential, inverse additive and additive weights. As discussed at length in Chap. 3, these weighting schemes exhibit sufficiently different characteristics to model the transition from the maximum to the average. Based on the proof of Theorem 3.2.1, we know that $orness(W_U^{add}) = \frac{2}{3}$ holds independent of the aggregation length. We find, based on the proofs of Theorems 3.2.2 and 3.2.3, that the $orness$ values of weight vectors W_U^{exp} and W_U^{invadd} do depend on the aggregation length p. This constitutes an important difference between the three OWA aggregations, which is illustrated in our experimental study. We do not include our *Mult* alternative introduced in Sect. 3.2.2, because it only showed clear advantages on datasets with very specific properties.

6.3.2 Overview of the Framework

Our proposed fuzzy set based multi-instance classifiers are divided into two families, which both require a number of settings to be specified. The BFMIC family computes $C(X)$ as an aggregation of $R(X, B)$ values derived from training bags B in class C. The IFMIC classifiers compute $B(x)$ values based on an instance similarity relation, aggregate these values to derive instance class membership degrees $C(x)$ and finally use these in the computation of $C(X)$. In naming our methods, we use the following conventions based on the abbreviations introduced in Tables 6.2 and 6.3:

- **Instance-based fuzzy multi-instance classifiers**: we list these methods as IFMIC-$B(x)$-$C(x)$-$C(X)$. As an example, our IFMIC-Max-Avg-MaxAdd method uses Max to calculate the $B(x)$ values, Avg to determine $C(x)$ and MaxAdd to finally derive $C(X)$.
- **Bag-based fuzzy multi-instance classifiers**: these methods are listed as BFMIC-$R(X, B)$-$C(X)$. As an example, our bag-based classifier BFMIC-H-Avg uses the Hausdorff similarity to compute the similarity between bags and Avg to calculate $C(X)$.

6.3.3 Worked Examples

To illustrate the workflow of our IFMIC and BFMIC classifiers, we detail the actions of some example algorithms on the toy problem presented in Table 6.4. The training set consists of five bags B_1, B_2, B_3, B_4 and B_5. The first three belong to class C_1, the last two to class C_2. We consider the classification of a new bag $X = \{x_1, x_2, x_3\}$. Table 6.4 lists the instance similarity values $R_I(\cdot, \cdot)$ between instances in the training bags and in bag X. We evaluate two of our proposed algorithms, the instance-based IFMIC-MaxInvadd-Avg-Max method and the bag-based BFMIC-AvgH-MaxAdd classifier. Below, we describe the calculations of these methods in their classification process of X.

Classification by IFMIC-MaxInvadd-Avg-Max For each instance x in the bag X, this method calculates its membership degree to the two classes. As an example, to compute $C_1(x_1)$ we select the bags belonging to class C_1 and first determine the affinity of x_1 with these bags based on MaxInvadd. We find

$$\begin{aligned} B_1(x_1) &= \mathrm{OWA}_{W_U^{invadd}}(\{R_I(x_1, b) \mid b \in B_1\}) \\ &= \frac{6}{11} \cdot 0.7 + \frac{3}{11} \cdot 0.5 + \frac{2}{11} \cdot 0.1 \\ &= \frac{59}{110}. \end{aligned}$$

Table 6.4 Two-class multi-instance toy dataset represented by its instance similarity values between the training instances and $x_1, x_2, x_3 \in X$

Bags of class C_1		$R_I(\cdot, x_1)$	$R_I(\cdot, x_2)$	$R_I(\cdot, x_3)$
B_1	x_{11}	0.1	0.2	0.2
	x_{12}	0.7	0.6	0.3
	x_{13}	0.5	0.0	0.1
B_2	x_{21}	0.4	0.4	0.5
	x_{22}	0.3	0.6	0.4
B_3	x_{31}	0.7	0.8	0.7
	x_{32}	0.5	0.3	0.9
	x_{33}	0.7	0.2	0.2
Bags of class C_2		$R_I(\cdot, x_1)$	$R_I(\cdot, x_2)$	$R_I(\cdot, x_3)$
B_4	x_{41}	1.0	0.8	0.9
	x_{42}	0.7	0.6	0.3
	x_{43}	0.5	0.5	0.9
	x_{44}	0.2	0.9	0.8
B_5	x_{51}	0.6	0.3	0.7
	x_{52}	0.9	0.8	0.8
	x_{53}	0.7	1.0	0.6

In the same way, we derive $B_2(x_1) = \frac{11}{30}$ and $B_3(x_1) = \frac{83}{110}$. The class membership degree $C_1(x_1)$ is set to the average of these values (Avg), such that

$$C_1(x_1) = \frac{1}{3}\left(\frac{59}{110} + \frac{11}{30} + \frac{83}{110}\right) = \frac{547}{990}.$$

Analogously, we find $C_1(x_2) = \frac{485}{990}$ and $C_1(x_3) = \frac{469}{990}$. The membership degree of X to class C_1 is determined as the maximum value obtained by one of its instances (Max), namely

$$C_1(X) = \max[C_1(x_1), C_1(x_2), C_1(x_3)] = \frac{547}{990}.$$

For the second class, we can follow the same procedure and derive $C_2(X) = \frac{879}{1100} > C_1(X)$, such that the method assigns X to class C_2.

Classification by BFMIC-AvgH-MaxAdd The prediction of this bag-based method is based on the similarity of X with the training bags, computed by the complement of the average Hausdorff distance. The similarity of X with bag B_1 is calculated as

$$
\begin{aligned}
R(X, B_1) &= 1 - \frac{1}{|X| + |B_1|} \left(\sum_{x \in X} \min_{b \in B_1} \delta(x, b) + \sum_{b \in B_1} \min_{x \in X} \delta(x, b) \right) \\
&= 1 - \frac{1}{3+3} \left((0.3 + 0.4 + 0.7) + (0.8 + 0.3 + 0.5) \right) \\
&= \frac{1}{2},
\end{aligned}
$$

where we remind the reader that the $\delta(\cdot, \cdot)$ values are computed as the complement of the $R_I(\cdot, \cdot)$ results in Table 6.4. Similarly, we find $R(X, B_2) = \frac{26}{50}$ and $R(X, B_3) = \frac{4}{5}$. Based on these values and the definition of MaxAdd in Table 6.3, we derive

$$
\begin{aligned}
C_1(X) &= \text{OWA}_{W_U^{add}}(\{R(X, B) \mid B \in T_{C_1}\}) \\
&= \frac{6}{12} \cdot \frac{4}{5} + \frac{4}{12} \cdot \frac{26}{50} + \frac{2}{12} \cdot \frac{1}{2} \\
&= \frac{197}{300}.
\end{aligned}
$$

For the second class, we find $C_2(X) = \frac{161}{180}$. As for the above IFMIC method, we find $C_2(X) > C_1(X)$, such that the method assigns bag X to class C_2.

6.3.4 Theoretical Complexity Analysis

In this section, we analyse the theoretical complexity of our proposed IFMIC and BFMIC methods. Below, n_B and n_X denote the sizes of bags B and X, M the number of training bags, n_{max} the size of the largest training bag and b_{max} the size of the largest class. We set $c = |\mathcal{C}|$ to the number of classes. When the instances in bags are described by d features, the cost of a similarity or distance calculation between a pair of instances is $\mathcal{O}(d)$ (see (3.13) and (6.2)).

IFMIC methods We consider the different steps performed by our instance-based methods:

- $\boldsymbol{B(x)}$: the affinity of an instance x with a bag B is measured by one of the alternatives listed in Table 6.2. The Max and Avg versions are linear in the number of instances in B and their complexity is therefore given as $\mathcal{O}(n_B \cdot d)$, taking into account the cost of the $R_I(\cdot, \cdot)$ calculations. The use of the OWA aggregation in MaxExp, MaxInvadd and MaxAdd implies the additional cost of the sorting operation (Definition 3.1.1). Consequently, their complexity with respect to the $B(x)$ calculation is $\mathcal{O}(n_B \cdot (\log(n_B) + d))$. Since the size of the training bags is limited to n_{max} (a very loose upper bound), we find that Max and Avg have complexity $\mathcal{O}(n_{max} \cdot d)$ and MaxExp, MaxInvadd and MaxAdd complexity $\mathcal{O}(n_{max} \cdot (\log(n_{max}) + d))$.

- $\boldsymbol{C(x)}$: secondly, when the $B(x)$ values have been computed, an IFMIC method determines the instance class membership degrees, for which the Max, MaxExp, MaxInvadd, MaxAdd and Avg options can again be used (see Table 6.2). The Avg and Max alternatives are obviously linear in the number of training bags and have complexity $\mathcal{O}(b_{max})$. For the OWA based settings, we have to account for the sorting step and their complexity is therefore $\mathcal{O}(b_{max}\log(b_{max}))$. These expressions ignore the cost of the previous step, that is, the cost of computing the affinity of instances with bags.
- $\boldsymbol{C(X)}$: finally, we determine the cost of calculating the class membership degrees of a test bag X, again momentarily ignoring the cost of the previous steps. As before, when using Max or Avg, we can derive a linear complexity $\mathcal{O}(n_X)$, while the MaxExp, MaxInvadd and MaxAdd settings come at $\mathcal{O}(n_X\log(n_X))$ cost.

As an example of the overall complexity of an IFMIC method, we consider the IFMIC-MaxAdd-MaxInvadd-MaxExp version. We select this classifier, as it has, for each step, an option with the highest computational complexity as discussed above. As a result, its complexity constitutes an upper bound on that of all included IFMIC methods. To classify a previously unseen bag X, this method computes, for each class C,

$$\begin{aligned} C(X) &= \mathrm{OWA}_{W_U^{exp}}(\{C(x) \mid x \in X\}) \\ &= \mathrm{OWA}_{W_U^{exp}}(\{\mathrm{OWA}_{W_U^{invadd}}(\{B(x) \mid B \in T_C\}) \mid x \in X\}) \\ &= \mathrm{OWA}_{W_U^{exp}}(\{\mathrm{OWA}_{W_U^{invadd}}(\{\mathrm{OWA}_{W_U^{add}}(\{R_I(x,b) \mid b \in B\}) \mid B \in T_C\}) \mid x \in X\}). \end{aligned}$$

For each class, this calculation has complexity

$$\begin{aligned} &\mathcal{O}(n_X \cdot (b_{max} \cdot (n_{max} \cdot (\log(n_{max}) + d)) + b_{max}\log(b_{max})) + n_X\log(n_X)) \\ &\qquad = \mathcal{O}(n_X \cdot (b_{max} \cdot (n_{max} \cdot (\log(n_{max}) + d) + \log(b_{max})) + \log(n_X))). \end{aligned}$$

If we take into account that this computation needs to be repeated for each class, the total classification complexity of this method (and therefore the upper bound on the complexity for our included IFMIC methods) is given as

$$\mathcal{O}(c \cdot n_X \cdot (b_{max} \cdot (n_{max} \cdot (\log(n_{max}) + d) + \log(b_{max})) + \log(n_X))). \tag{6.3}$$

BFMIC methods We consider the different steps performed by our bag-based methods:

- $\boldsymbol{R(X, B)}$: in its first step, a BFMIC method computes the similarity between bags. The Hausdorff and average Hausdorff similarity (H and AvgH) are based on distances between pairs of instances in the bags and consequently have complexity $\mathcal{O}(n_{max}^2 \cdot d)$. Their OWA modified versions (HExp, HInvadd, HAdd, AvgHExp, AvgHInvadd, AvgHAdd) require the pairwise distances to be sorted and have complexity $\mathcal{O}(n_{max}^2(\log(n_{max}^2) + d)) = \mathcal{O}(n_{max}^2(\log(n_{max}) + d))$.

- $\boldsymbol{C(X)}$: secondly, the method estimates the membership degree of a bag to the classes. As for the analogous instance membership degrees to the classes for the IFMIC family, we find that the Avg and Max settings have complexity $\mathcal{O}(b_{max})$ and that the MaxExp, MaxInvadd and MaxAdd alternatives have complexity $\mathcal{O}(b_{max} \log(b_{max}))$.

As an example, we determine the overall complexity of our BFMIC-AvgHInvadd-MaxInvadd method. As for the instance-based classifier above, this method is one that attains the highest complexity within the BFMIC family. Its complexity therefore forms an upper bound on that of the included bag-based methods. When classifying an unseen bag X, for each class C, the method calculates the value $C(X) = \text{OWA}_{W_U^{invadd}}(\{R(X, B) \mid B \in T_C\})$, with the $R(X, \cdot)$ calculated by means of the AvgHInvadd alternative listed in Table 6.3. For each class, we can derive that this calculation has a cost of

$$\begin{aligned}&\mathcal{O}(b_{max} \cdot n_X \cdot n_{max} \cdot (\log(n_X \cdot n_{max}) + d) + b_{max} \log(b_{max})) \\ &\qquad = \mathcal{O}(b_{max} \cdot (n_X \cdot n_{max} \cdot (\log(n_X \cdot n_{max}) + d) + \log(b_{max}))).\end{aligned}$$

Again considering that this computation is repeated for all classes, the total classification cost of BFMIC-AvgHInvadd-MaxInvadd of a bag X is

$$\mathcal{O}(c \cdot b_{max} \cdot (n_X \cdot n_{max} \cdot (\log(n_X \cdot n_{max}) + d) + \log(b_{max}))). \tag{6.4}$$

Summary Comparing the complexity bounds (6.3) and (6.4) for our instance-based and bag-based methods respectively, we note that they both have a log-linear dependency on b_{max} (the size of the majority class) and n_{max} (the size of the largest training bag). When we remove the common terms in expressions (6.3) and (6.4), we can derive that the former contains the additional $\mathcal{O}(c \cdot n_X \cdot b_{max} \cdot \log(b_{max}) + c \cdot n_X \cdot \log(n_X))$ terms and the latter the additional $\mathcal{O}(c \cdot b_{max} \cdot \log(b_{max}) + c \cdot b_{max} \cdot n_X \cdot n_{max} \cdot \log(n_X))$ terms. The first term, log-linear in b_{max}, is n_X times larger for the IFMIC family than it is for the BFMIC family. The second term, log-linear in n_X, is $n_{max} \cdot b_{max}$ times larger for the BFMIC methods compared to the IFMIC methods, a value linear in the size of the majority class and the size of the largest training bag. Overall, we can expect the second term to imply a larger difference between the costs of the two families. The IFMIC methods can therefore intuitively be expected to be less costly than their BFMIC relatives.

6.4 Experimental Study of Our Fuzzy Multi-instance Classifiers

In Sect. 6.3, we have presented a framework of fuzzy set based multi-instance methods. It consists of two families: instance-based fuzzy multi-instance classifiers (IFMIC methods) and bag-based fuzzy multi-instance classifiers (BFMIC methods).

In order to gain more insight into these algorithms, we conduct an internal evaluation of the two groups. We compare their different parameter settings and explain why certain alternatives are more appropriate than others. The datasets used in this experimental study are described in Sect. 6.4.1. Next, Sects. 6.4.2 and 6.4.3 consider our IFMIC and BFMIC methods respectively. Taking all possible combinations of the settings listed in Tables 6.2 and 6.3 into account, this implies the evaluation of 125 IFMIC methods and 40 BFMIC methods.

6.4.1 Datasets

In this evaluation, we apply our methods on the 33 more or less balanced multi-instance datasets described in Table 6.5. We list their number of features, number of bags and total number of instances. All datasets consist of two classes, of which class 0 is the negative class and class 1 the positive class. The class sizes are included in the table. As an additional information component, we also list the minimum, mean, median and maximum size of a bag in each dataset. All results reported in the remainder of this chapter are obtained via five-fold cross validation. As evaluation measures, we use the overall classification accuracy as well as the accuracy on the two classes separately. The use of the traditional classification accuracy is justified because there is little to no class imbalance in the datasets in Table 6.5.

6.4.2 The IFMIC Family

As described in Sect. 6.3.1.1, our instance-based fuzzy classifiers depend on three parameters, namely the ways to compute (i) the affinity $B(x)$ of an instance x with a bag B, (ii) the membership degree $C(x)$ of instance x to class C and (iii) the membership degree $C(X)$ of bag X to class C. Each can be set to one of five alternatives (Max, MaxExp, MaxInvadd, MaxAdd and Avg).

We use three evaluation measures: the overall accuracy, the accuracy on class 0 (negative class) and the accuracy on class 1 (positive class). Table 6.6 lists the five highest scoring IFMIC methods for each measure based on their average performance across the datasets in Table 6.5. To complement these results, the worst obtained mean values for these measures are included as well. A striking point is that each classifier in the top five for each evaluation measure uses the Max alternative for $B(x)$. With respect to the $C(x)$ step, the preference for a procedure closer to a maximum (Max or MaxExp) or to an average (Avg or MaxAdd) seems to depend on the particular class, as evident from the different high-ranking versions for the class 0 and class 1 accuracy. Based on the overall accuracy, MaxExp seems to be the preferred choice. A preference for one of the $C(X)$ alternatives is less clear-cut based on the results included in Table 6.6. We study and explain the differences in performance of the possible settings in the following sections.

Table 6.5 Balanced multi-instance datasets used in our experimental evaluation

Dataset	Class sizes					Bag sizes			
	nFeat	nBags	nInsts	nCl0	nCl1	Min	Mean	Median	Max
Musk1	166	92	476	45	47	2	5.17	4	40
Musk2	166	101	6598	62	39	2	65.33	14	1044
Atoms	10	188	1618	63	125	5	8.61	8	15
Bonds	16	188	3995	63	125	8	21.25	20	40
Chains	24	188	5349	63	125	8	28.45	27	52
Elephant	230	200	1391	100	100	2	6.96	7	13
Fox	230	200	1320	100	100	2	6.60	6	13
Tiger	230	200	1220	100	100	1	6.10	6	13
EastWest	24	20	213	10	10	4	10.65	9	16
WestEast	24	20	213	10	10	4	10.65	9	16
AntDrugs5	5	400	3728	202	198	5	9.32	9	14
AntDrugs10	10	400	3787	214	186	5	9.47	10	14
AntDrugs20	20	400	3736	212	188	5	9.34	9	14
TREC9Sel-1	320	400	3224	200	200	1	8.06	8	19
TREC9Sel-2	303	400	3344	200	200	1	8.36	8	19
TREC9Sel-3	324	400	3246	200	200	1	8.12	8	20
TREC9Sel-4	306	400	3391	200	200	1	8.48	8	20
TREC9Sel-7	300	400	3367	200	200	1	8.42	8	19
TREC9Sel-9	299	400	3300	200	200	1	8.25	8	19
TREC9Sel-10	303	400	3453	200	200	1	8.63	9	20
WIRSel-7	303	113	3423	58	55	4	30.29	24	200
WIRSel-8	303	113	3423	58	55	4	30.29	24	200
WIRSel-9	301	113	3423	58	55	4	30.29	24	200
Corel01vs02	9	200	838	100	100	2	4.19	4	11
Corel01vs03	9	200	794	100	100	2	3.97	3	11
Corel01vs04	9	200	1243	100	100	2	6.22	6	13
Corel01vs05	9	200	684	100	100	2	3.42	2	11
Corel02vs03	9	200	664	100	100	2	3.32	3	10
Corel02vs04	9	200	1113	100	100	2	5.57	5	13
Corel02vs05	9	200	554	100	100	2	2.77	2	10
Corel03vs04	9	200	1069	100	100	2	5.35	5	13
Corel03vs05	9	200	510	100	100	2	2.55	2	6
Corel04vs05	9	200	959	100	100	2	4.80	3	13

6.4.2.1 Instance-to-Bag Affinity $B(x)$

The mean results obtained by the five alternatives for $B(x)$ are listed in Table 6.7. These results were derived as averages over the datasets in Table 6.5 and all IFMIC methods using a particular setting. For example, the results for Avg are taken as the

Table 6.6 Five best performing IFMIC methods for each evaluation measure. The results are taken as averages over the datasets in Table 6.5. The bottom line lists the methods with the lowest obtained mean results for these measures

Classifier	Acc cl0	Classifier	Acc cl1	Classifier	Accuracy
Max-MaxExp-MaxInvadd	0.8068	Max-Avg-MaxAdd	0.8918	Max-MaxExp-MaxAdd	0.8168
Max-MaxExp-MaxAdd	0.8068	Max-Avg-MaxExp	0.8893	Max-MaxExp-MaxInvadd	0.8155
Max-MaxExp-Avg	0.8039	Max-Avg-MaxInvadd	0.8886	Max-MaxExp-Avg	0.8153
Max-Max-Avg	0.8022	Max-MaxAdd-MaxAdd	0.8808	Max-MaxInvadd-Avg	0.8128
Max-Max-MaxAdd	0.7991	Max-MaxAdd-MaxInvadd	0.8802	Max-MaxInvadd-MaxAdd	0.8121
Avg-Avg-Max	0.5024	Avg-Max-Avg	0.5971	Avg-Avg-Max	0.6502

Table 6.7 Setting rankings of the $B(x)$ alternatives of the IFMIC methods

$B(x)$	Acc cl0	$B(x)$	Acc cl1	$B(x)$	Accuracy
Max	0.7175±0.0789	Max	0.8324±0.0524	Max	0.7786±0.0304
MaxAdd	0.6904±0.0679	MaxExp	0.8034±0.0550	MaxExp	0.7537±0.0276
MaxInvadd	0.6839±0.0723	MaxInvadd	0.7795±0.0699	MaxInvadd	0.7459±0.0322
MaxExp	0.6803±0.0794	MaxAdd	0.7605±0.0667	MaxAdd	0.7381±0.0298
Avg	0.6729±0.0729	Avg	0.7248±0.0682	Avg	0.7121±0.0275

mean value across the 25 included IFMIC-Avg-*-* methods. The reported standard deviations are taken across these methods as well.

As was already evident from the results listed in Table 6.6, the Max setting is clearly preferred. It attains the highest mean accuracy, both overall as well as on the two classes separately. The performance of Avg is lowest for all measures and that of the OWA alternatives is found between Max and Avg. This indicates that there is no point in softening the strict maximum (and thereby bringing it closer to an average) in the calculation of $B(x)$.

The $B(x)$ step determines the membership of an instance x to a bag B. The results show that the most similar instance $y \in B$ contains the most information. Involving all instances in B in this calculation, either by assigning them all equal weights (Avg) or not (MaxAdd, MaxInvadd, MaxExp), deteriorates the performance. This phenomenon is explained as follows. In a multi-instance dataset, the variety between instances in a bag can be quite large. Indeed, considering the standard two-class multi-instance hypothesis that states that a bag is positive when at least one of its

instances belongs to the positive class, a positive bag can both contain instances affiliated with the positive concept and instances affiliated with the negative concept. Assume that we draw two instances x_1 and x_2 from bag B that are affiliated with the positive and negative class respectively and that, based on their feature values, their similarity is (unsurprisingly) low. If we involve value $R_I(x_1, x_2)$ in the calculation of $B(x_1)$ and $B(x_2)$, it would unjustifiably lower the results, even though x_1 and x_2 both belong to B and their affinity with the bag should be high. We also observe that the OWA approaches, as intermediate options between the average and maximum, do not provide an advantage over the strict maximum and we conclude that they are not useful for the estimation of $B(x)$.

6.4.2.2 Instance-to-Class Membership Degree $C(x)$

Table 6.8 lists the results for the five alternatives for the $C(x)$ calculations, obtained in the same way as described in the previous section. MaxInvadd has the highest mean accuracy. We observe that this is due to its more or less balanced performance on the two classes, that is, it does not obtain extremely good or poor results on one of them. The Max and Avg alternatives sit at the bottom of the accuracy table, which is due to their inferior performance on one class (low class 0 accuracy for Avg, low class 1 accuracy for Max). MaxInvadd attains the best trade-off between the maximum and average aggregations.

We need to explain why Avg works better for class 1, while class 0 prefers Max. The largest differences can be found on the TREC and WIR datasets from Table 6.5, which respectively belong to the text classification and web mining domains. On this group of ten datasets, the average class 0 accuracies are 0.3125 (Avg) and 0.7233 (Max), while the average class 1 accuracies are 0.9557 (Avg) and 0.7683 (Max). Based on these results, we only need to consider the behaviour of Avg, since Max achieves a more or less balanced performance on the two classes. It warrants an explanation why Avg assigns the class 1 label more easily than the class 0 label for these text datasets.

We momentarily fix $B(x)$ to Max (the favoured setting of this parameter as discussed in Sect. 6.4.2.1) and consider the distribution of the average $B(x)$ values. These values (and therefore the $C(x)$ values as computed by Avg) of instances belonging

Table 6.8 Setting rankings of the $C(x)$ alternatives of the IFMIC methods

$C(x)$	Acc cl0	$C(x)$	Acc cl1	$C(x)$	Accuracy
Max	0.7579±0.0301	Avg	0.8434±0.0342	MaxInvadd	0.7696±0.0308
MaxExp	0.7539±0.0331	MaxAdd	0.8272±0.0415	MaxExp	0.7555±0.0375
MaxInvadd	0.7142±0.0304	MaxInvadd	0.8010±0.0442	MaxAdd	0.7423±0.0317
MaxAdd	0.6383±0.0322	MaxExp	0.7332±0.0513	Max	0.7376±0.0358
Avg	0.5807±0.0319	Max	0.6957±0.0549	Avg	0.7235±0.0285

to class 0 bags are close together for bags B of class 0 and class 1 and are even a little higher for those of class 1. On the other hand, for instances belonging to class 1 bags, the opposite occurs: their average $B(x)$ values are higher for class 1 than class 0. As a result, we can expect the membership degree to class 1 to be higher for all instances, regardless of the bag label of their parent bag. Clearly, there is an attracting force in class 1 for the ten text datasets. Recall that class 1 is the positive class and the characteristics of the text datasets may lead to an attraction of this class when Avg is used to compute the membership degree of instances to classes. We can also note that these datasets further stand out as having a relatively low similarity of instances in different bags, that is, instances in a bag B are usually more similar to each other than they are to instances in other bags. We observe the same characteristics for the image datasets Elephant, Fox and Tiger, where the difference in class accuracy results is present as well, albeit less pronounced. Finally, we should note that the observed behaviour can not be solely due to our choice of instance similarity relation $R_I(\cdot, \cdot)$. All these datasets have far more than 20 features and are therefore processed with the cosine similarity, but other datasets using the cosine similarity do not exhibit this behaviour. Instead, the positive class of these datasets must have some sort of attracting property, which results in an overly easy assignment of the class 1 label by Avg.

The overall preference of MaxInvadd can be deduced from its $orness$ value, which places it between Max and Avg, but closer to the former. The observant reader will have noted that the first two columns in Table 6.8 rank the $C(x)$ settings according to their $orness$ value, namely in decreasing order for the class 0 accuracy and in increasing order for the class 1 accuracy. The MaxInvadd setting achieves the best trade-off between a high class 1 accuracy and an acceptable class 0 accuracy.

6.4.2.3 Bag-to-Class Membership Degree $C(X)$

Table 6.9 lists the results for the five alternatives for the $C(X)$ calculations, once again obtained by averaging the results of all IFMIC methods using a particular setting. The average-related options yield the best performance. In particular, the Max setting provides clearly inferior results compared to the other alternatives. It is interesting to note that the ranking obtained for the overall accuracy is exactly the

Table 6.9 Setting rankings of the $C(X)$ alternatives of the IFMIC methods

$C(X)$	Acc cl0	$C(X)$	Acc cl1	$C(X)$	Accuracy
Avg	0.7244±0.0647	MaxAdd	0.8002±0.0677	Avg	0.7680±0.0290
MaxAdd	0.7008±0.0721	MaxInvadd	0.7926±0.0705	MaxAdd	0.7624±0.0285
MaxInvadd	0.6942±0.0737	Avg	0.7916±0.0731	MaxInvadd	0.7551±0.0293
MaxExp	0.6774±0.0736	MaxExp	0.7773±0.0709	MaxExp	0.7387±0.0294
Max	0.6483±0.0735	Max	0.7388±0.0643	Max	0.7042±0.0261

opposite from the one observed for the $B(x)$ calculations in Table 6.7. The $C(X)$ membership degrees are aggregations of the $C(x)$ values for all instances $x \in X$. It is reasonable to expect that all instances in X should contribute (more or less) equally to this calculation for a proper class estimation. The experimental results confirm this.

6.4.2.4 Conclusion

Overall, we can advise the use of the IFMIC-Max-MaxInvadd-Avg classifier. From the settings evaluated in this chapter, Max for $B(x)$, MaxInvadd for $C(x)$ and Avg for $C(X)$ stand out as the best choices on average, for reasons discussed above. We include it in the global comparison of our proposed methods to state-of-the-art multi-instance classifiers in Sect. 6.7.

6.4.3 The BFMIC Family

As described in Sect. 6.3.1.2, our bag-based fuzzy classifiers depend on two parameters, namely the ways to compute (i) the similarity between two bags $R(X, B)$ and (ii) the membership degree $C(X)$ of a bag to a class. The former can be set to one of eight alternatives (H, HExp, HInvadd, HAdd, AvgH, AvgHExp, AvgHInvadd, AvgHAdd), while five choices are available for the latter (Max, MaxExp, MaxInvadd, MaxAdd, Avg).

Table 6.10 lists the five highest scoring BFMIC methods for the overall and class-wise accuracies based on their average performance across the datasets in Table 6.5. In particular for the class 0 accuracy, we observe a stand-out performance of the

Table 6.10 Five best performing BFMIC methods for each evaluation measure. The results are taken as averages over the datasets in Table 6.5. The bottom line lists the methods with the lowest obtained mean results for these measures

Classifier	Acc cl0	Classifier	Acc cl1	Classifier	Accuracy
AvgH-MaxExp	0.8045	AvgH-Avg	0.8867	AvgH-MaxInvadd	0.8204
AvgH-Max	0.7873	AvgHExp-Avg	0.8819	AvgH-MaxExp	0.8202
H-Max	0.7870	AvgHExp-MaxAdd	0.8807	AvgHExp-MaxInvadd	0.8114
HInvadd-MaxExp	0.7812	AvgH-MaxAdd	0.8788	AvgHExp-MaxExp	0.8112
AvgHExp-MaxExp	0.7785	AvgHInvadd-Avg	0.8690	HInvadd-MaxExp	0.8094
HAdd-Avg	0.5982	AvgHAdd-Max	0.7698	H-Avg	0.6986

BFMIC-AvgH-MaxExp method. For the other evaluation measures, the top results are closer together. In the following sections, we evaluate the performance of the eight $R(X, B)$ and five $C(X)$ alternatives.

6.4.3.1 Bag Similarity $R(X, B)$

Table 6.11 lists the mean results and standard deviations of the eight alternatives for $R(X, B)$. As we did for the IFMIC methods in the previous section, these results were obtained as averages over the datasets in Table 6.5 and based on all BFMIC methods using a particular setting. For example, the results for AvgH are derived from those of the five included BFMIC-AvgH-* methods.

The experiments show that the average Hausdorff similarity is the best choice for the bag similarity relation, attaining the best average result for both class-wise accuracies as well as the overall accuracy. The preference of the average over the regular Hausdorff distance was already expressed in [487]. Our modified versions using OWA aggregations (AvgHExp, AvgHInvadd, AvgHAdd) do not further improve its performance. This is in line with the observations with regard to the $B(x)$ step in our instance-based classifiers (Sect. 6.4.2.1). For the $B(x)$ parameter, the strict maximum proved superior to our OWA-softened versions and we could attribute this to the variability existing within bags. The same occurs here. The OWA modifications AvgHExp, AvgHInvadd and AvgHAdd of the average Hausdorff distance lie at the level of the instance-wise comparisons. Once again, we find that the least distant (most similar) instance carries the most information (Table 6.11).

6.4.3.2 Bag-to-Class Membership Degree $C(X)$

Table 6.12 lists the results for the five alternatives for $C(X)$. The rankings of these settings are in line with the ones obtained in Sect. 6.4.2.2 for the $C(x)$ step in our IFMIC methods. We observe a strong difference between Avg on the one hand and

Table 6.11 Setting rankings of the $R(X, B)$ alternatives of the BFMIC methods

$R(X, B)$	Acc cl0	$R(X, B)$	Acc cl1	$R(X, B)$	Accuracy
AvgH	0.7364±0.0639	AvgH	0.8514±0.0332	AvgH	0.7993±0.0190
AvgHExp	0.7140±0.0568	AvgHExp	0.8498±0.0340	AvgHExp	0.7920±0.0179
HInvadd	0.7129±0.0623	AvgHInvadd	0.8374±0.0316	AvgHInvadd	0.7802±0.0175
HExp	0.7091±0.0514	HInvadd	0.8304±0.0176	HInvadd	0.7784±0.0260
H	0.7054±0.0706	HAdd	0.8252±0.0239	HAdd	0.7715±0.0238
HAdd	0.7040±0.0622	HExp	0.8241±0.0125	HExp	0.7711±0.0260
AvgHInvadd	0.6976±0.0517	AvgHAdd	0.8233±0.0351	AvgHAdd	0.7697±0.0163
AvgHAdd	0.6904±0.0486	H	0.7975±0.0120	H	0.7544±0.0349

Table 6.12 Setting ranking of the $C(X)$ alternatives of the BFMIC methods

$C(X)$	Acc cl0	$C(X)$	Acc cl1	$C(X)$	Accuracy
MaxExp	0.7691±0.0199	MaxAdd	0.8522±0.0219	MaxExp	0.8007±0.0122
Max	0.7568±0.0218	Avg	0.8488±0.0316	MaxInvadd	0.7964±0.0166
MaxInvadd	0.7331±0.0154	MaxInvadd	0.8415±0.0172	Max	0.7787±0.0108
MaxAdd	0.6685±0.0108	MaxExp	0.8177±0.0132	MaxAdd	0.7682±0.0179
Avg	0.6160±0.0122	Max	0.7892±0.0098	Avg	0.7414±0.0213

the Max* versions on the other. On class 0, the performance of Avg is very low, while it ranks in second place for the class 1 accuracy. This is mainly due to the results on the TREC and WIR datasets, where its class 1 accuracy is very high and its class 0 accuracy relatively low. A similar behaviour was observed within the IFMIC methods in Sect. 6.4.2.2. The MaxAdd method, which is the one most related to Avg among the OWA alternatives, exhibits a comparable pattern, although it performs notably better on class 0 than Avg does. The remaining OWA approximations MaxExp and MaxInvadd yield a better balance between the classes. They also perform better (on average) than the strict maximum. As for $C(x)$ in IFMIC, these settings achieve a preferable trade-off between the characteristics of the maximum and average.

6.4.3.3 Conclusion

Based on the results reported and explanations given in Sects. 6.4.3.1 and 6.4.3.2, we can put forward our BFMIC-AvgH-MaxExp or BFMIC-AvgH-MaxInvadd methods as best performing ones among the evaluated set of BFMIC methods. We include the BFMIC-AvgH-MaxExp method in our global comparison conducted in Sect. 6.7.

6.5 Fuzzy Rough Classifiers for Class Imbalanced Multi-instance Data

Section 6.3 has focused on the development of general multi-instance classifiers based on fuzzy set theory, which have been compared among each other in Sect. 6.4. We now turn our attention to the challenges associated with two-class imbalanced multi-instance data (see Sect. 6.2.3). As discussed at length in Chap. 4, an imbalanced class distribution renders the classification task more difficult. We propose a framework of multi-instance classifiers based on fuzzy rough set theory. As for our fuzzy set based multi-instance classifiers described in Sect. 6.3, we present two families of methods: instance-based and bag-based classifiers. Both are based on the imbalance-resistant single-instance IFROWANN method ([361], Sect. 4.1.3).

We present our proposals in Sect. 6.5.1. Section 6.5.2 summarizes the proposed framework and Sect. 6.5.3 discusses the theoretical complexity of our methods.

6.5.1 Proposed Classifiers

Based on the IFROWANN method for single-instance binary imbalanced classification, we present a framework of fuzzy rough multi-instance classification algorithms. We define a fuzzy rough multi-instance classifier as a mapping

$$f : \mathbb{N}^{\mathcal{X}} \to \mathcal{C} : X \mapsto \arg\max_{C \in \mathcal{C}} [\underline{C}(X)], \tag{6.5}$$

where $\underline{C}(X)$ represents the membership degree of bag X to the fuzzy rough lower approximation of class C. In case of a tie for the highest value in (6.5), X is assigned to the smallest class, as done by the original IFROWANN method. We consider two approaches to compute the $\underline{C}(X)$ values, corresponding to two classifier families:

- Instance-based fuzzy rough multi-instance classifiers (IFRMIC family, Sect. 6.5.1.2): value $\underline{C}(X)$ is derived by aggregating the corresponding instance-wise values $\underline{C}(x)$ of instances x in bag X.
- Bag-based fuzzy rough multi-instance classifiers (BFRMIC family, Sect. 6.5.1.3): the definition of the fuzzy rough lower approximation is extended to the level of bags based on a bag similarity relation and bag class membership degrees.

As discussed in Sect. 6.3 on our fuzzy multi-instance classifiers, the constituent parts of our IFRMIC and BFRMIC methods can be obtained in several ways. Below, we describe the two fuzzy rough classifier families in more detail, but first recall the weighting schemes for the OWA based fuzzy rough lower approximation used in the IFROWANN method.

6.5.1.1 IFROWANN Weighting Schemes

As recalled in Sect. 4.1.3, the IFROWANN method [361] is a binary single-instance classifier based on fuzzy rough set theory that assigns an instance to the class for which its membership degree to the lower approximation is highest. The OWA based fuzzy rough set model is used and a variety of weighting scheme combinations (class dependent or not) has been studied. The definitions of the weight combinations provided in Sect. 4.1.3 do not literally coincide with the ones listed in [361]. We relied on expression (4.1) for the lower approximation calculation, which bases its result for $\underline{C}(x)$ on instances $y \notin C$. The authors of [361] achieved the same effect by basing the lower approximation on all training instances y and adding a number of leading zeroes in the weight vectors. The contributions of instances $y \in C$ are paired with

zero weights, effectively reducing the result to (4.1) with the weight combinations listed in Sect. 4.1.3.

The reduction to instances $y \notin C$ in (4.1) is a consequence of crisp class membership degrees in single-instance datasets. Instances are directly related to a class label. In multi-instance data, instances and classes are separated by the bag level, that is, only bags have an associated class label while their instances do not. Keeping this in mind, we use the OWA weight combinations as listed in the original [361] contribution, which are based on all training samples rather than reducing them to the members of one of the two classes. We consider

$$\begin{aligned}
\mathcal{W}_1 &= \langle W_{Min}^{add*}, W_{Maj}^{add*} \rangle \\
\mathcal{W}_2 &= \left\langle W_{Min}^{add*}, W_{Maj}^{exp*} \right\rangle \\
\mathcal{W}_3 &= \langle W_{Min}^{exp*}, W_{Maj}^{add*} \rangle \\
\mathcal{W}_4 &= \left\langle W_{Min}^{exp*}, W_{Maj}^{exp*} \right\rangle \\
\mathcal{W}_5 &= \left\langle W_{Min}^{add*,\gamma}, W_{Maj}^{add*} \right\rangle \\
\mathcal{W}_6 &= \left\langle W_{Min}^{add*,\gamma}, W_{Maj}^{exp*} \right\rangle \\
\mathcal{W}_7 &= \langle W_{Min}^{strict*}, W_{Maj}^{strict*} \rangle \\
\mathcal{W}_8 &= \langle W_{Min}^{invadd*}, W_{Maj}^{invadd*} \rangle
\end{aligned}$$

with

$$W_{Min}^{add*} = \left\langle \underbrace{0, \ldots, 0}_{|Min|}, \frac{2}{|Maj|(|Maj|+1)}, \ldots, \frac{2(|Maj|-1)}{|Maj|(|Maj|+1)}, \frac{2}{|Maj|+1} \right\rangle,$$

$$W_{Maj}^{add*} = \left\langle \underbrace{0, \ldots, 0}_{|Maj|}, \frac{2}{|Min|(|Min|+1)}, \ldots, \frac{2(|Min|-1)}{|Min|(|Min|+1)}, \frac{2}{|Min|+1} \right\rangle,$$

$$W_{Min}^{exp*} = \left\langle \underbrace{0, \ldots, 0}_{|Min|}, \frac{1}{2^{|Maj|}-1}, \ldots, \frac{2^{|Maj|-2}}{2^{|Maj|}-1}, \frac{2^{|Maj|-1}}{2^{|Maj|}-1} \right\rangle,$$

$$W_{Maj}^{exp*} = \left\langle \underbrace{0, \ldots, 0}_{|Maj|}, \frac{1}{2^{|Min|}-1}, \ldots, \frac{2^{|Min|-2}}{2^{|Min|}-1}, \frac{2^{|Min|-1}}{2^{|Min|}-1} \right\rangle,$$

$$W_{Min}^{s} = \left\langle \underbrace{0, \ldots, 0}_{|Min|+|Maj|-1}, 1 \right\rangle,$$

$$W_{Maj}^{s} = \left\langle \underbrace{0, \ldots, 0}_{|Min|+|Maj|-1}, 1 \right\rangle,$$

$$W_{Min}^{invadd*} = \left\langle \underbrace{0,\ldots,0}_{|Min|}, \frac{1}{|Maj|\sum_{i=1}^{|Maj|}\frac{1}{i}}, \frac{1}{(|Maj|-1)\sum_{i=1}^{|Maj|}\frac{1}{i}}, \ldots, \frac{1}{\sum_{i=1}^{|Maj|}\frac{1}{i}} \right\rangle,$$

$$W_{Maj}^{invadd*} = \left\langle \underbrace{0,\ldots,0}_{|Maj|}, \frac{1}{|Min|\sum_{i=1}^{|Min|}\frac{1}{i}}, \frac{1}{(|Min|-1)\sum_{i=1}^{|Min|}\frac{1}{i}}, \ldots, \frac{1}{\sum_{i=1}^{|Min|}\frac{1}{i}} \right\rangle,$$

$$W_{Min}^{add*,\gamma} = \left\langle \underbrace{0,\ldots,0}_{|Min|+|Maj|-r}, \frac{2}{r(r+1)}, \frac{4}{r(r+1)}, \ldots, \frac{2(r-1)}{r(r+1)}, \frac{2}{r+1} \right\rangle.$$

In these expressions, $|Min|$ and $|Maj|$ are the sizes of the minority and majority classes respectively and $r = \lceil |Min| + \gamma(|Maj| - |Min|) \rceil$. The weight vectors are used within the calculations of the lower approximations as listed in Sects. 6.5.1.2 and 6.5.1.3 below, where the W_{Min} and W_{Maj} vectors are used for the minority and majority class calculations respectively. We have added the seventh and eighth combination to evaluate the effect of using strict or inverse additive weights on both classes. These complement the inclusion of $\mathcal{W}_1$ and $\mathcal{W}_4$, which respectively use additive and exponential weights for the lower approximation calculations regardless of the class being evaluated. As a result, comparing weighting schemes $\mathcal{W}_1$, $\mathcal{W}_4$, $\mathcal{W}_7$ and $\mathcal{W}_8$ corresponds to the comparison between the actions of *Add*, *Exp*, *Strict* and *Invadd* (Chap. 3), provided that a number of positions are set to zero in the weight vector ($|Min|$ for the minority class and $|Maj|$ for the majority class). This renders them class dependent. Naturally, the definition of *Strict* ($\mathcal{W}_7$) is not influenced by these additional zero positions.

The combinations listed above are the ones proposed and evaluated in [361], with the addition of $\mathcal{W}_7$ and $\mathcal{W}_8$. We should note that the use of the leading zero weights made perfect sense in that context (that is, single-instance binary classification), since, as discussed at the beginning of this section, a crisp class relation is used to measure the membership degree of an instance (observation) to a class. In our IFRMIC and BFRMIC methods discussed below, the class membership degrees are set to the fuzzy values $C(x)$ and $C(X)$ respectively, which are calculated as listed in Tables 6.2 and 6.3. The choice of the (number of) zero positions in the weight vectors is consequently somewhat artificial here. We will therefore also test the performance of the weighting schemes when a crisp class membership relation is used. Naturally, such a crisp class relation only makes sense for the BFRMIC methods, because the observations (bags) are associated with a class label. In the IFRMIC methods, we compute the class membership degree of instances ourselves and it is not appropriate to simply use the crisp class label of the bag they belong to.

6.5.1.2 The IFRMIC Family

Our instance-based fuzzy rough classifiers determine the $\underline{C}(X)$ values in (6.5) by aggregating the corresponding $\underline{C}(x)$ values for all instances x in bag X. The instance-wise values are determined as

$$\underline{C}(x) = \text{OWA}_{W_L}(\{\mathcal{I}(R_I(x, y), C(y)) \mid y \in B, B \in T\}), \tag{6.6}$$

with T the full training set and y the training instances. Weight vector W_L is set to a W_{Min} vector when C is the minority class and to a W_{Maj} vector when C is the majority class. The W_{Min}-W_{Maj} combination corresponds to one of those listed in Sect. 6.5.1.1. The values $|Min|$ and $|Maj|$ in their definitions are set to the total number of instances in minority and majority class bags respectively. The instance similarity relation $R_I(\cdot, \cdot)$ coincides with the one listed in Sect. 6.3.1.1. The instance-to-class membership degrees $C(y)$ can be computed in the same way as listed in Table 6.2, namely by aggregating $B(y)$ values for $B \in T_C$ by means of Max, MaxExp, MaxInvadd, MaxAdd or Avg. The $B(y)$ values can themselves be obtained with the same alternatives. As these values are only required for instances in training bags, they can be precomputed prior to the classification phase of the IFRMIC methods.

The aggregation of the set $\{\underline{C}(x) \mid x \in X\}$ to $\underline{C}(X)$ is handled by the same alternatives as included for our IFMIC methods in Table 6.2, namely:

$$\begin{array}{ll}
\text{Max:} & \underline{C}(X) = \max\limits_{x \in X} \underline{C}(x) \\
\text{MaxExp:} & \underline{C}(X) = \text{OWA}_{W_U^{exp}}(\{\underline{C}(x) \mid x \in X\}) \\
\text{MaxInvadd:} & \underline{C}(X) = \text{OWA}_{W_U^{invadd}}(\{\underline{C}(x) \mid x \in X\}) \\
\text{MaxAdd:} & \underline{C}(X) = \text{OWA}_{W_U^{add}}(\{\underline{C}(x) \mid x \in X\}) \\
\text{Avg:} & \underline{C}(X) = \frac{1}{|X|} \sum\limits_{x \in X} \underline{C}(x)
\end{array}$$

6.5.1.3 The BFRMIC Family

The calculation of the $\underline{C}(X)$ values by our bag-based fuzzy rough classifiers is performed entirely at the bag level and is a direct extension of expression (3.5). The general $\underline{C}(X)$ formulation of our bag-based methods is therefore given by

$$\underline{C}(X) = \text{OWA}_{W_L}(\{\mathcal{I}(R(X, B), C(B)) \mid B \in T\}). \tag{6.7}$$

The bag similarity values $R(X, B)$ and the $C(B)$ membership degree of training bags B to class C can be computed by the alternatives listed in Table 6.3. In particular, $R(X, B)$ can be obtained with H, HExp, HInvadd, HAdd, AvgH, AvgHExp, AvgH-Invadd or AvgHAdd and $C(B)$ can be calculated using Max, MaxExp, MaxInvadd, MaxAdd or Avg. As for the IFRMIC methods, the $C(B)$ values need only be computed for the training bags. To avoid repeated calculations, they can be derived prior to the classification phase. The W_L vector in the calculation of $\underline{C}(X)$ is set to a W_{Min} vector when C is the minority class and to a W_{Maj} vector when C is the majority class. As for the IFRMIC methods, the W_{Min}-W_{Maj} combination corresponds to one of those listed in Sect. 6.5.1.1. The $|Min|$ and $|Maj|$ values are now respectively set to the number of minority class and majority class training bags.

As noted in Sect. 6.5.1.1, we can use crisp membership degrees of bags to classes rather than the $C(B)$ values computed with Max, MaxExp, MaxInvadd, MaxAdd or Avg. In particular, for a training bag B and classes C_0 and C_1, we set

$$C_0(B) = \begin{cases} 1 & \text{if } B \in C_0 \\ 0 & \text{if } B \in C_1 \end{cases} \quad \text{and} \quad C_1(B) = \begin{cases} 1 & \text{if } B \in C_1 \\ 0 & \text{if } B \in C_0 \end{cases}. \tag{6.8}$$

When we use these definitions as opposed to the ones listed in Table 6.3, the definition of the IFROWANN weighting schemes makes intuitively more sense, as they once again coincide with the reduction to a class complement as described in Sect. 3.1.3. In our experiments, we evaluate this variant within (6.7) in combination with the eight weight settings listed in Sect. 6.5.1.1 as well.

6.5.2 *Overview of the Framework*

A visual overview of the calculation flow of our IFRMIC and BFRMIC classifiers is presented in Fig. 6.3. The bag-based methods require the specification of how to calculate the $R(X, B)$ and $C(B)$ values. We can use the alternatives listed in Table 6.3, with the addition of the crisp class membership degrees defined in expression (6.8). The $\underline{C}(X)$ values are obtained by means of (6.7) and one of the eight weight combinations listed in Sect. 6.5.1.1. Our instance-based methods rely on the definitions of the $B(y)$ and $C(y)$ values as well as the way to aggregate the set of $\underline{C}(x)$ values to $\underline{C}(X)$. We use the different settings provided in Table 6.2, which lists five alternatives for each of them. The $\underline{C}(x)$ values are derived using (6.6) and, as for the BFRMIC methods, the IFROWANN weight combinations given in Sect. 6.5.1.1.

In naming our methods, we use the following conventions based on the abbreviations introduced in Tables 6.2 and 6.3 and the weight combinations of Sect. 6.5.1.1:

- **Instance-based fuzzy rough multi-instance classifiers**: these methods are listed as IFRMIC-$B(y)$-$C(y)$-$\underline{C}(X)$-$\mathcal{W}_*$. As an example, our instance-based IFRMIC-Max-Avg-MaxAdd-$\mathcal{W}_1$ classifier uses Max to calculate the $B(y)$ values, Avg to

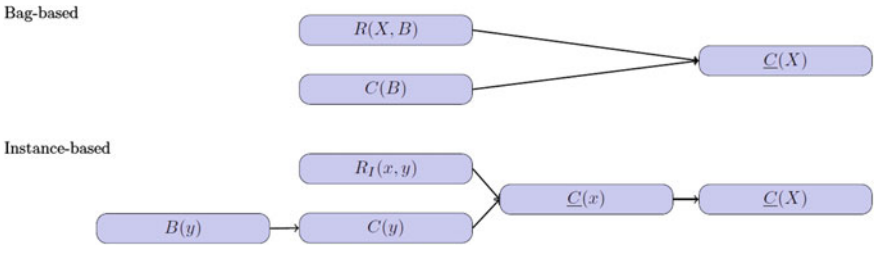


Fig. 6.3 Overview of the flow of our instance-based and bag-based fuzzy rough multi-instance classifiers

determine $C(y)$, weight combination $\mathcal{W}_1$ in its $\underline{C}(x)$ calculations and finally Max-Add to derive $\underline{C}(X)$.

- **Bag-based fuzzy rough multi-instance classifiers**: these methods are listed as BFRMIC-$R(X, B)$-$C(B)$-$\mathcal{W}_*$. As an example, our bag-based BFRMIC-H-Avg-$\mathcal{W}_1$ classifier uses the Hausdorff similarity to compute the similarity between bags, Avg to calculate $C(B)$ and weight combination $\mathcal{W}_1$ to compute $\underline{C}(X)$. In case of crisp class membership degrees for the training bags, we write 'Crisp' for the $C(B)$ component.

6.5.3 Theoretical Complexity Analysis

We now discuss the cost complexity of our IFRMIC and BFRMIC proposals. As noted in Sects. 6.5.1.2 and 6.5.1.3, some components can be precomputed during a training phase. We therefore split the computational cost into two parts: the cost of the learning stage and the classification cost of an unseen bag X. As before, n_B and n_X denote the sizes of bags B and X, M the number of training bags, n_{max} the size of the largest training bag, b_{max} the size of the largest class and c the number of classes. We use n as the total number of training instances. The cost of an instance similarity or distance calculation is $\mathcal{O}(d)$, where d is the number of input features.

IFRMIC methods We first consider the training phase of our instance-based methods, in which they precompute the $C(y)$ values for all training instances. In order to do so, the $B(y)$ affinity values need to be derived first:

- $\boldsymbol{B(y)}$: as discussed in Sect. 6.3.4, the Max and Avg settings have complexity $\mathcal{O}(n_{max} \cdot d)$, while the other alternatives come at $\mathcal{O}(n_{max} \cdot (\log(n_{max}) + d))$ cost. This corresponds to the cost of computing $B(y)$ for one instance y. As this needs to be repeated for each training instance and each training bag, the cost of computing all $B(y)$ values is $\mathcal{O}(M \cdot n \cdot n_{max} \cdot d)$ for Max and Avg and $\mathcal{O}(M \cdot n \cdot n_{max} \cdot (\log(n_{max}) + d))$ for MaxExp, MaxInvadd and MaxAdd.
- $\boldsymbol{C(y)}$: once the $B(\cdot)$ values have been computed, they can be used to determine the instance-to-class membership degrees. As derived in Sect. 6.3.4, the Avg and Max alternatives have complexity $\mathcal{O}(b_{max})$, while the MaxExp, MaxInvadd and Max-Add have complexity $\mathcal{O}(b_{max} \log(b_{max}))$. When the $B(\cdot)$ values have been derived in the first step, the cost of computing all $C(y)$ values (that is, for all instances, for all classes) is $\mathcal{O}(c \cdot n \cdot b_{max})$ for Avg and Max and $\mathcal{O}(c \cdot n \cdot b_{max} \log(b_{max}))$ for MaxExp, MaxInvadd and MaxAdd.

Next, we assess the cost of classifying a new bag X, knowing that all $C(y)$ values needed in (6.6) have been computed.

- $\boldsymbol{\underline{C}(x)}$: the similarity of x with all training instances y needs to be computed, which can be achieved at $\mathcal{O}(n \cdot d)$ cost. Next, the set of values to aggregate needs to be constructed. Since all components have been (pre)computed, this can be done at linear cost, that is, $\mathcal{O}(n)$. The OWA aggregation of this set has a computational

complexity of $\mathcal{O}(n \log(n))$. The total cost of computing $\underline{C}(x)$ is $\mathcal{O}(n \cdot (\log(n) + d))$.

- $\underline{\boldsymbol{C}}(\boldsymbol{X})$: the $\underline{C}(x)$ values are aggregated to $\underline{C}(X)$ by means of Avg, Max, MaxExp, MaxInvadd or MaxAdd. The former two settings have linear complexity $\mathcal{O}(n_X)$, while the latter three come at $\mathcal{O}(n_X \log(n_X))$ cost.

To illustrate the above, we consider the actions of our IFRMIC-MaxAdd-MaxInvadd-MaxExp-$\mathcal{W}_1$ algorithm. As we did in Sect. 6.3.4, each component of this method has the highest cost listed above, such that we derive an upper bound on the complexity of the included IFRMIC methods. In the training phase, the $C(y)$ values are calculated for all training instances y as follows:

$$\begin{aligned} C(y) &= \mathrm{OWA}_{W_U^{invadd}}(\{B(y) \mid B \in T_C\}) \\ &= \mathrm{OWA}_{W_U^{invadd}}(\{\mathrm{OWA}_{W_U^{add}}(\{R_I(y,z) \mid z \in B\}) \mid B \in T_C\}). \end{aligned}$$

For each instance y, for each class C, this calculation has complexity

$$\mathcal{O}(b_{max} \cdot (n_{max} \cdot (\log(n_{max}) + d) + \log(b_{max}))).$$

The total computational cost of the training phase is consequently

$$\mathcal{O}(n \cdot c \cdot b_{max} \cdot (n_{max} \cdot (\log(n_{max}) + d) + \log(b_{max}))). \tag{6.9}$$

To classify bag X, the IFRMIC method computes, for each class,

$$\begin{aligned} \underline{C}(X) &= \mathrm{OWA}_{W_U^{exp}}(\{\underline{C}(x) \mid x \in X\}) \\ &= \mathrm{OWA}_{W_U^{exp}}(\{\mathrm{OWA}_{W_L}(\{\mathcal{I}(R_I(x,y), C(y)) \mid y \in B, B \in T\}) \mid x \in X\}), \end{aligned}$$

in which the $C(y)$ values have been precomputed. The cost to compute $\underline{C}(X)$ is

$$\mathcal{O}(n_X \cdot (\log(n_X) + n \cdot (\log(n) + d))).$$

Taking into account that this needs to be repeated for each class, the total classification cost of bag X for this IFRMIC method is

$$\mathcal{O}(c \cdot n_X \cdot (\log(n_X) + n \cdot (\log(n) + d))).$$

BFRMIC methods We first consider the training phase of our bag-based methods, in which they precompute the $C(B)$ values for all training bags. These calculations are based on bag similarity values $R(\cdot, \cdot)$:

- $\boldsymbol{R}(\boldsymbol{A}, \boldsymbol{B})$: the calculation of the similarity between two bags comes at $\mathcal{O}(n_{max}^2 \cdot d)$ cost for H and AvgH and at $\mathcal{O}(n_{max}^2(\log(n_{max}) + d))$ cost for HExp, HInvadd, HAdd, AvgHExp, AvgHInvadd and AvgHAdd. These computations are repeated for each pair of bags, resulting in total complexities of $\mathcal{O}(M^2 \cdot n_{max}^2 \cdot d)$ (H, AvgH)

and $\mathcal{O}(M^2 \cdot n_{max}^2(\log(n_{max}) + d))$ (HExp, HInvadd, HAdd, AvgHExp, AvgHInvadd, AvgHAdd).

- $\boldsymbol{C(B)}$: the $C(B)$ calculation is based on all training bags belonging to class C. It can be obtained at $\mathcal{O}(b_{max})$ cost for Avg and Max and at $\mathcal{O}(b_{max}\log(b_{max}))$ cost for MaxExp, MaxInvadd and Invadd. We need to derive these values for all training bags B and all classes C, resulting in a total cost of $\mathcal{O}(c \cdot M \cdot b_{max})$ (Avg, Max) and $\mathcal{O}(c \cdot M \cdot b_{max}\log(b_{max}))$ (MaxExp, MaxInvadd, MaxAdd).

Note that in case of crisp class membership degrees for the training bags (Sect. 6.5.1.3), this entire step can be avoided.

Next, we derive the cost of classifying a new bag X, knowing that all $C(B)$ values needed in (6.7) have been computed. In order to derive $\underline{C}(X)$, we need to compute the bag similarity $R(X, B)$ of X with all training bags B. Based on the complexity results discussed above, this step has $\mathcal{O}(M \cdot n_{max}^2 \cdot d)$ cost for H and AvgH and $\mathcal{O}(M \cdot n_{max}^2(\log(n_{max}) + d))$ cost for HExp, HInvadd, HAdd, AvgHExp, AvgHInvadd and AvgHAdd. The construction of the set of values to aggregate has linear cost $\mathcal{O}(M)$. The final OWA aggregation has $\mathcal{O}(M\log(M))$ complexity, such that the total cost of computing $\underline{C}(X)$ is $\mathcal{O}(M \cdot (\log(M) + n_{max}^2(\log(n_{max}) + d)))$.

As an example, we compute the computational complexity associated with our BFRMIC-AvgHInvadd-MaxInvadd-$\mathcal{W}_1$ method, which constitutes an upper bound on the complexity of the included BFRMIC methods. In its training phase, this method computes the $C(B)$ values for all training bags as $C(B) = \mathrm{OWA}_{W_U^{invadd}}(\{R(A, B) \mid A \in T_C\})$ with $R(A, B)$ computed by means of AvgHInvadd. Based on the derivations above, computing all $C(B)$ values for all training bags and all classes results in a total training complexity of

$$\mathcal{O}(c \cdot M \cdot b_{max}\log(b_{max}) + M^2 \cdot n_{max}^2(\log(n_{max}) + d)). \tag{6.10}$$

To classify a bag X, the BFRMIC method computes, for each class,

$$\underline{C}(X) = \mathrm{OWA}_{W_L}(\{\mathcal{I}(R(X, B), C(B)) \mid B \in T\}),$$

for which all $C(B)$ values have been precomputed. Computing all $R(X, B)$ values by means of AvgHInvadd implies a cost of $\mathcal{O}(M \cdot n_{max}^2(\log(n_{max}) + d))$. Taking the cost of the OWA aggregations into account, we derive a total classification cost of

$$\mathcal{O}(M \cdot (c \cdot \log(M) + n_{max}^2(\log(n_{max}) + d))).$$

Summary Comparing the complexity bounds (6.9) and (6.10) for the training time of our instance-based and bag-based methods respectively, we note that they can be split up in three terms:

- Term in $\mathcal{O}(c \cdot b_{max}\log(b_{max}))$: this term is linear in the number of training instances n for the IFRMIC methods and linear in the number of training bags M for the BFRMIC methods. The latter value is smaller than the former by definition.

- Term in $\mathcal{O}(n_{max}\log(n_{max}))$: this term has coefficient $n \cdot c \cdot b_{max}$ for IFRMIC and coefficient $M^2 \cdot n_{max}$ for BFRMIC.
- Term in $\mathcal{O}(n_{max} \cdot d)$: this term has coefficient $n \cdot c \cdot b_{max}$ for IFRMIC and coefficient $M^2 \cdot n_{max}$ for BFRMIC.

The complexity bounds on the classification costs of the IFRMIC and BFRMIC methods contain no common or highly similar terms and it is not straightforward to compare the two.

6.6 Experimental Study of Our Fuzzy Rough Multi-instance Classifiers

In Sect. 6.5, we have presented a framework of fuzzy rough set based multi-instance methods, that can be divided into two groups: the IFRMIC and BFRMIC classifiers. As we did in Sect. 6.4, we now conduct an internal parameter comparison of these two families. We focus on imbalanced multi-instance classification and describe the datasets used in this study in Sect. 6.6.1. Sections 6.6.2 and 6.6.3 present the actual parameter comparison for the IFRMIC and BFRMIC methods respectively. For both groups, we first fix the weight combination to $\mathcal{W}_4$ and evaluate the other parameter settings. Afterwards, we compare the different weight combinations listed in Sect. 6.5.1.1 and, for the BFRMIC methods, the effect of using crisp class membership degrees for the training bags.

6.6.1 Datasets

The 33 imbalanced multi-instance datasets are presented in Table 6.13, which includes their number of features, number of bags and total number of instances. We list the sizes of class 0 and class 1 as well as the IR between them. The minimum, mean, median and maximum bag sizes for each dataset are included as well. As done in Sect. 6.4, all reported results are obtained via five-fold cross validation. As evaluation measures, we use the class-wise accuracies, the AUC and the balanced accuracy. In both (6.6) and (6.7), we use the Łukasiewicz implicator ($I_L(a,b) = \min(1-a+b,1)$).

6.6.2 The IFRMIC Family

Our instance-based fuzzy rough classifiers depend on four parameters: (i) the instance-to-bag affinity $B(y)$, (ii) the instance-to-class membership degree $C(y)$, (iii) the IFROWANN weight combinations used to obtain $\underline{C}(x)$ and (iv) the way to

Table 6.13 Imbalanced multi-instance datasets used in the experimental evaluation

Dataset	Class sizes						Bag sizes			
	nFeat	nBags	nInsts	nCl0	nCl1	IR	Min	Mean	Median	Max
WIRSel-1	304	113	3423	92	21	4.38	4	30.29	24	200
WIRSel-2	298	113	3423	92	21	4.38	4	30.29	24	200
WIRSel-3	303	113	3423	92	21	4.38	4	30.29	24	200
WIRSel-4	303	113	3423	24	89	3.71	4	30.29	24	200
WIRSel-5	302	113	3423	24	89	3.71	4	30.29	24	200
WIRSel-6	304	113	3423	24	89	3.71	4	30.29	24	200
Corel20-1	9	2000	7947	1900	100	19.00	2	3.97	3	13
Corel20-2	9	2000	7947	1900	100	19.00	2	3.97	3	13
Corel20-3	9	2000	7947	1900	100	19.00	2	3.97	3	13
Corel20-4	9	2000	7947	1900	100	19.00	2	3.97	3	13
Corel20-5	9	2000	7947	1900	100	19.00	2	3.97	3	13
Corel20-6	9	2000	7947	1900	100	19.00	2	3.97	3	13
Corel20-7	9	2000	7947	1900	100	19.00	2	3.97	3	13
Corel20-8	9	2000	7947	1900	100	19.00	2	3.97	3	13
Corel20-9	9	2000	7947	1900	100	19.00	2	3.97	3	13
Corel20-10	9	2000	7947	1900	100	19.00	2	3.97	3	13
Corel20-11	9	2000	7947	1900	100	19.00	2	3.97	3	13
Corel20-12	9	2000	7947	1900	100	19.00	2	3.97	3	13
Corel20-13	9	2000	7947	1900	100	19.00	2	3.97	3	13
Corel20-14	9	2000	7947	1900	100	19.00	2	3.97	3	13
Corel20-15	9	2000	7947	1900	100	19.00	2	3.97	3	13
Corel20-16	9	2000	7947	1900	100	19.00	2	3.97	3	13
Corel20-17	9	2000	7947	1900	100	19.00	2	3.97	3	13
Corel20-18	9	2000	7947	1900	100	19.00	2	3.97	3	13
Corel20-19	9	2000	7947	1900	100	19.00	2	3.97	3	13
Corel20-20	9	2000	7947	1900	100	19.00	2	3.97	3	13
Elephant	230	125	912	25	100	4.00	2	7.30	7	13
Fox	230	121	781	21	100	4.76	2	6.45	6	13
Mut_atoms	10	167	1438	42	125	2.98	5	8.61	8	15
Mut_bonds	16	160	3558	35	125	3.57	8	22.24	22	40
Mut_chains	24	152	4630	27	125	4.63	8	30.46	30	52
Tiger	230	126	708	26	100	3.85	1	5.62	5	12
Thioredoxin	8	193	26611	168	25	6.72	35	137.88	145	189

aggregate these values to $\underline{C}(X)$. The first, second and fourth components can be set to Max, MaxExp, MaxInvadd, MaxAdd and Avg, while the weight combination options are listed in Sect. 6.5.1.1. We evaluate our classifiers by means of their majority class accuracy (Acc maj), minority class accuracy (Acc min), AUC and balanced accuracy values on the datasets in Table 6.13.

Table 6.14 Five best performing IFRMIC methods for each evaluation measure. The results are taken as averages over the datasets in Table 6.13. Weight combination $\mathcal{W}_4$ is used. For each evaluation measure, the bottom line lists the method with the lowest mean result

Classifier	Acc maj	Classifier	Acc min
MaxInvadd-Max-MaxInvadd	0.9942	Max-Avg-Avg	0.7609
MaxExp-MaxExp-MaxExp	0.9942	Max-MaxAdd-MaxInvadd	0.7543
MaxExp-MaxExp-MaxInvadd	0.9937	Max-MaxAdd-MaxAdd	0.7528
MaxAdd-MaxExp-MaxInvadd	0.9935	Max-Avg-MaxAdd	0.7520
MaxInvadd-Max-MaxExp	0.9932	Max-Avg-MaxInvadd	0.7519
Max-Avg-Max	0.5417	MaxInvadd-MaxExp-MaxExp	0.1011
Classifier	AUC	Classifier	Balacc
Max-MaxAdd-MaxAdd	0.8192	Max-MaxAdd-Avg	0.7451
Max-MaxAdd-Avg	0.8146	Max-MaxAdd-MaxAdd	0.7365
Max-Avg-MaxAdd	0.8145	Max-MaxAdd-MaxInvadd	0.7306
Max-Avg-Avg	0.8122	MaxInvadd-MaxAdd-Avg	0.7247
Max-MaxAdd-MaxInvadd	0.8120	MaxExp-MaxAdd-Avg	0.7195
Avg-Avg-Max	0.6842	Avg-MaxExp-MaxExp	0.5459

In a first step, we fix the IFROWANN weight combination to $\mathcal{W}_4$, a well-performing setting in the original paper [361]. Table 6.14 lists the top five performing IFRMIC methods for each evaluation measure. As we observed for our instance-based fuzzy classifiers in Sect. 6.4.2, the Max setting for $B(y)$ is clearly preferable based on its strong performance for the AUC and balanced accuracy summary measures as well as its favourable results on the minority class. Several methods obtain a near perfect classification of the majority class at the cost of a very poor recognition of minority elements. We discuss the relative performance of the different parameter settings in Sect. 6.6.2.1. The effect of the various IFROWANN weight combinations on the results is assessed in Sect. 6.6.2.2.

6.6.2.1 Setting Rankings

As stated above, we fix the weight combination used in (6.6) to $\mathcal{W}_4$ in this section. Table 6.15 presents the mean results and standard deviations of the different settings for $B(y)$, $C(y)$ and $\underline{C}(X)$, each of which can be set to Max, MaxExp, MaxInvadd, MaxAdd and Avg. The results were once again obtained as averages over the datasets in Table 6.13 and based on all IFRMIC methods using a particular setting. For example, the results for the Avg setting of $B(y)$ are computed using the 25 IFRMIC-Avg-*-*-$\mathcal{W}_4$ methods.

We observe that the AUC and balanced accuracy rankings of the $B(y)$ and $\underline{C}(X)$ alternatives are more or less the same as the corresponding settings within the IFMIC methods (see Tables 6.7 and 6.9) and we can refer the reader to the above discussion

Table 6.15 Setting rankings of the $B(y)$, $C(y)$ and $\underline{C}(X)$ alternatives of the IFRMIC methods. Weight combination $\mathcal{W}_4$ is used

$B(y)$	Acc maj	$C(y)$	Acc maj	$\underline{C}(X)$	Acc maj
MaxInvadd	0.8520±0.1618	MaxExp	0.9893±0.0050	Avg	0.8668±0.1451
MaxExp	0.8483±0.1584	Max	0.9828±0.0179	MaxAdd	0.8582±0.1561
Avg	0.8483±0.1647	MaxInvadd	0.9577±0.0207	MaxInvadd	0.8556±0.1593
MaxAdd	0.8455±0.1644	MaxAdd	0.6944±0.0385	MaxExp	0.8454±0.1669
Max	0.8431±0.1630	Avg	0.6130±0.0282	Max	0.8112±0.1777
$B(y)$	Acc min	$C(y)$	Acc min	$\underline{C}(X)$	Acc min
Max	0.4440±0.2555	Avg	0.7149±0.0244	Avg	0.4086±0.2619
MaxExp	0.3989±0.2516	MaxAdd	0.6874±0.0364	MaxAdd	0.4036±0.2608
MaxAdd	0.3960±0.2624	MaxInvadd	0.3181±0.0348	Max	0.4024±0.2386
MaxInvadd	0.3921±0.2587	Max	0.1567±0.0295	MaxInvadd	0.3993±0.2601
Avg	0.3715±0.2453	MaxExp	0.1253±0.0165	MaxExp	0.3885±0.2569
$B(y)$	AUC	$C(y)$	AUC	$\underline{C}(X)$	AUC
Max	0.7895±0.0203	MaxAdd	0.7808±0.0314	MaxAdd	0.7870±0.0170
MaxExp	0.7679±0.0235	Avg	0.7738±0.0336	Avg	0.7852±0.0168
MaxInvadd	0.7654±0.0305	MaxInvadd	0.7680±0.0270	MaxInvadd	0.7805±0.0174
MaxAdd	0.7612±0.0324	Max	0.7572±0.0290	MaxExp	0.7600±0.0187
Avg	0.7492±0.0319	MaxExp	0.7534±0.0249	Max	0.7206±0.0221
$B(y)$	Balacc	$C(y)$	Balacc	$\underline{C}(X)$	Balacc
Max	0.6436±0.0591	MaxAdd	0.6909±0.0297	Avg	0.6377±0.0637
MaxExp	0.6236±0.0519	Avg	0.6639±0.0228	MaxAdd	0.6309±0.0591
MaxInvadd	0.6220±0.0565	MaxInvadd	0.6379±0.0192	MaxInvadd	0.6274±0.0571
MaxAdd	0.6207±0.0569	Max	0.5697±0.0110	MaxExp	0.6170±0.0524
Avg	0.6099±0.0487	MaxExp	0.5573±0.0074	Max	0.6068±0.0380

for an explanation. With respect to the $C(y)$ calculation, in which the membership degree of training instances to classes is determined, the rankings are different from the ones observed for the IFMIC methods in Table 6.8. In particular, the averaging approaches Avg and MaxAdd perform notably better within IFRMIC compared to their actions within IFMIC. The results in Table 6.15 show that this is due to their performance on the minority class. Considering these results in more detail, we can ascertain that this behaviour is independent of which class is the minority class (that is, minority class 0 or minority class 1, see Table 6.13).

A $C(y)$ value is computed as an aggregation of $B(y)$ values, where bag B belongs to class C. Let us assume that $B(y)$ is computed by means of Max, the preferred setting. This means that the affinity of instance y with bag B is set to its maximum similarity with an instance in that bag. We can compute the distribution of the mean and maximum $B(y)$ values (corresponding to $C(y)$ with Avg and Max respectively), in which we can make a distinction based on the class of B. We observe:

- **Avg for $C(y)$**: on average across the 33 datasets in Table 6.13, for majority bags B, the mean $B(y)$ values are between 0.5909 (average minimum) and 0.8831 (average maximum), with average mean and median values of 0.8385 and 0.8416 respectively. For minority bags B, the mean $B(y)$ values are between 0.5866 (average minimum) and 0.8982 (average maximum), with average mean and median values of 0.8351 and 0.8370 respectively. These values are clearly at the same level and the influence of the class imbalance is diminished.
- **Max for $C(y)$**: on the other hand, for the maximum $B(y)$ values, we find 0.7257(min)-0.9536(mean)-0.9515(median)-0.9990(max) on average for majority bags B compared to 0.6410(min)-0.9148(mean)-0.9124(median)-0.9955(max) for minority bags B. These values are noticeably different, such that we can expect relatively too high majority class memberships degrees to be assigned to all instances. These will propagate to the level of bags, leading to many misclassifications of the minority class.

When we compute the difference between the mean $B(y)$ values for the majority and minority classes, the average value on the 33 datasets is 0.0034 and we observe both positive and negative differences. For the maximum $B(y)$ values, this average difference is 0.0388 and is positive for all datasets. This indicates that the observed problem of relatively too high majority class membership degrees with Max for $C(y)$ occurs consistently across the datasets.

6.6.2.2 Weight Combinations

In this section, we fix $B(y)$ to Max, $C(y)$ to MaxAdd and $\underline{C}(X)$ to Avg and evaluate the effect of different IFROWANN weight combinations in the calculation of the $\underline{C}(x)$ values in (6.6). The results are presented in Table 6.16 as averages across the 33 datasets in Table 6.13. Based on the class-wise accuracy and balanced accuracy values, we can conclude that two settings yield a befitting balance between the two classes, namely combinations $\mathcal{W}_4$ and $\mathcal{W}_7$. Setting $\mathcal{W}_3$, that combines exponential weights for the minority class calculations with additive weights for the majority class calculations, has a very strong performance on the majority class, while leading to many misclassifications on the minority class. On the other hand, combinations $\mathcal{W}_1$, $\mathcal{W}_2$, $\mathcal{W}_5$, $\mathcal{W}_6$ and $\mathcal{W}_8$ favour the minority class and perform poorly on the majority class. With respect to the AUC, $\mathcal{W}_4$ and $\mathcal{W}_7$ come out on top, but are dominated by $\mathcal{W}_8$. The AUC value of $\mathcal{W}_5$ is acceptable, but the results of the four other alternatives are relatively too low. In all, $\mathcal{W}_4$ and $\mathcal{W}_7$ clearly are most appropriate. The good performance of $\mathcal{W}_4$ has been observed in earlier studies as well, which was why we selected it as the fixed setting in the previous section. It uses exponential weights for both class approximations. Combination $\mathcal{W}_7$ uses strict instead of exponential weights and is therefore related to $\mathcal{W}_4$. The other six schemes take more values into account in the aggregation (for one or both classes), which proves to be inappropriate here. The OWA aggregation in (6.6) is taken over all training instances and can consequently be quite lengthy. The minimum number of instances in the datasets

Table 6.16 Evaluation of the eight weight combinations within IFRMIC-Max-MaxAdd-Avg-*. The value for γ in $\mathcal{W}_5$ and $\mathcal{W}_6$ is 0.1

Weights	Acc maj	Weights	Acc min	Weights	AUC	Weights	Balacc
$\mathcal{W}_3$	0.9966	$\mathcal{W}_2$	0.9983	$\mathcal{W}_8$	0.8386	$\mathcal{W}_7$	0.7478
$\mathcal{W}_4$	0.7467	$\mathcal{W}_6$	0.9941	$\mathcal{W}_7$	0.8173	$\mathcal{W}_4$	0.7451
$\mathcal{W}_7$	0.7396	$\mathcal{W}_1$	0.9912	$\mathcal{W}_4$	0.8146	$\mathcal{W}_5$	0.6607
$\mathcal{W}_5$	0.4428	$\mathcal{W}_8$	0.9167	$\mathcal{W}_5$	0.8087	$\mathcal{W}_8$	0.6444
$\mathcal{W}_8$	0.3721	$\mathcal{W}_5$	0.8787	$\mathcal{W}_6$	0.7209	$\mathcal{W}_3$	0.5491
$\mathcal{W}_6$	0.0708	$\mathcal{W}_7$	0.7560	$\mathcal{W}_1$	0.6747	$\mathcal{W}_6$	0.5325
$\mathcal{W}_1$	0.0438	$\mathcal{W}_4$	0.7436	$\mathcal{W}_3$	0.6733	$\mathcal{W}_1$	0.5175
$\mathcal{W}_2$	0.0036	$\mathcal{W}_3$	0.1015	$\mathcal{W}_2$	0.6428	$\mathcal{W}_2$	0.5010

from Table 6.13 is 708, but the maximum is 26611 and the median value 7947. This indicates why *Strict* or *Exp* may be preferred (see Chap. 3).

Finally, we note that we have also varied the γ parameter in $\mathcal{W}_5$ and $\mathcal{W}_6$. The lowest values of γ provided the best results with AUC values of 0.8394 ($\mathcal{W}_5$ with $\gamma = 0$) and 0.7705 ($\mathcal{W}_6$ with $\gamma = 0$) and balanced accuracy values of 0.7419 ($\mathcal{W}_5$ with $\gamma = 0$) and 0.5631 ($\mathcal{W}_6$ with $\gamma = 0$).

6.6.3 The BFRMIC Family

Our bag-based fuzzy rough classifiers rely on the definition of three parameters: (i) the bag similarity relation $R(X, B)$, (ii) the bag-to-class membership degree $C(B)$ and (iii) the way to compute the $\underline{C}(X)$ in (6.7). We evaluate our classifiers by means of their majority class accuracy, minority class accuracy, AUC and balanced accuracy values on the datasets in Table 6.13.

As we did in Sect. 6.6.2, we first fix the weight combination to $\mathcal{W}_4$ and compare the eight alternatives for $R(X, B)$ and five alternatives for $C(B)$ listed in Table 6.3. Table 6.17 provides the five BFRMIC methods with the best results for each evaluation measure. For the AUC and balanced accuracy measures, we encounter methods with similar settings as the best performing ones in Sect. 6.4.3 on top. It is striking that four out of five methods in the top five for the balanced accuracy use MaxAdd to compute the $C(B)$ values. We compare the different parameter settings in Sect. 6.6.3.1 and study the effect of the IFROWANN weight combination in Sect. 6.6.3.2. The version of BFRMIC with crisp class membership degrees for the training bags is evaluated separately in Sect. 6.6.3.3.

Table 6.17 Five best performing BFRMIC methods for each evaluation measure. The results are taken as averages over the datasets in Table 6.13. Weight combination $\mathcal{W}_4$ is used. For each evaluation measure, the bottom line lists the method with the lowest mean result

Classifier	Acc maj	Classifier	Acc min
HInvadd-MaxExp	0.9762	AvgH-MaxAdd	0.7613
AvgH-MaxExp	0.9757	AvgH-Avg	0.7557
HAdd-MaxExp	0.9750	AvgHExp-Avg	0.7434
H-MaxExp	0.9742	HAdd-Avg	0.7409
HExp-MaxExp	0.9739	HExp-Avg	0.7336
HAdd-Avg	0.6516	HAdd-MaxExp	0.2776
Classifier	AUC	Classifier	Balacc
AvgH-Max	0.8807	AvgH-MaxAdd	0.7608
AvgH-MaxExp	0.8784	AvgHExp-MaxAdd	0.7451
AvgH-MaxInvadd	0.8776	AvgHInvadd-MaxAdd	0.7410
AvgHExp-MaxInvadd	0.8576	AvgH-MaxInvadd	0.7402
AvgHInvadd-MaxInvadd	0.8556	AvgHAdd-MaxAdd	0.7397
H-Avg	0.7494	AvgHAdd-Max	0.6255

6.6.3.1 Setting Rankings

Fixing the weight combination in (6.7) to $\mathcal{W}_4$, we evaluate the performance of the eight settings for $R(X, B)$ and five settings for $C(B)$. Table 6.18 lists the average results and standard deviations of all alternatives for these two parameters. We observe that the rankings of the $R(X, B)$ settings are similar to the ones listed for the BFMIC methods in Table 6.11 and that AvgH comes out on top for all included measures. However, notable differences are observed for the rankings of the $C(B)$ versions.

With respect to the $C(B)$ calculations, we observe that Max, MaxExp and MaxInvadd perform very well on the majority class at the cost of many classification errors on the minority class. The behaviour of MaxAdd and Avg is distinctly different. These settings obtain a better balance between the minority and majority class accuracies. This is particularly true for MaxAdd, which is reflected in its superior balanced accuracy value. We explain the poor performance of Max (and MaxExp, MaxInvadd) in the following paragraphs.

As described in Sect. 6.5.1.3, the BFRMIC methods precompute their $C(B)$ values for the training bags in a learning phase. They do not use a leave-one-out procedure, that is, the similarity of a bag with itself is included in the aggregation to $C_B(B)$, where C_B is the class label of bag B. As a result, a training bag should be pulled more to its own class. In particular, Max sets $C_B(B) = 1$. When using an average-related procedure, this effect diminishes as the class size increases. Based on this observation, we can derive the following. To classify a bag X to the majority class C_{maj} or minority class C_{min}, the aggregation lengths in $\underline{C}_{maj}(X)$ and $\underline{C}_{min}(X)$ are

the same. In particular,

$$\underline{C}_{maj}(X) = \mathrm{OWA}_{W^{exp*}_{Maj}}(\{\min(1 - R(X, B) + C_{maj}(B), 1) \mid B \in T\}),$$
$$\underline{C}_{min}(X) = \mathrm{OWA}_{W^{exp*}_{Min}}(\{\min(1 - R(X, B) + C_{min}(B), 1) \mid B \in T\}).$$

Due to the large impact of the similarity of a training bag with itself in the calculation of $C(B)$ by Max, the training bags taking part in the aggregation of $\underline{C}_{maj}(X)$ will mostly belong to the minority class, while those in the aggregation of $\underline{C}_{min}(X)$ will mostly belong to the majority class. Concretely, we will find

$$\underline{C}_{maj}(X) = \mathrm{OWA}_{W^{exp*}_{Maj}}(\underbrace{\{1, 1, \ldots, 1\}}_{B \in C_{maj}} \cup \{\min(1 - R(X, B) + C_{maj}(B), 1) \mid B \in C_{min}\}), \quad (6.11)$$
$$\underline{C}_{min}(X) = \mathrm{OWA}_{W^{exp*}_{Min}}(\underbrace{\{1, 1, \ldots, 1\}}_{B \in C_{min}} \cup \{\min(1 - R(X, B) + C_{min}(B), 1) \mid B \in C_{maj}\}). \quad (6.12)$$

As a result of the sorting step in the OWA procedure, the values stemming from bags $B \in C_{maj}$ in (6.11) and from bags $B \in C_{min}$ in (6.12) are placed at the beginning of the ordered sequence (Definition 3.1.1) and are likely assigned a zero weight by the exponential weights used in $\mathcal{W}_4$. This supports our above claim that $\underline{C}_{maj}(X)$ and $\underline{C}_{min}(X)$ depend primarily on contributions of minority class and majority class bags respectively.

On average across the minority class 1 datasets, the maximum similarity values of minority class bags with majority class bags have a minimum value of 0.9160, a mean value of 0.9449, a median value of 0.9456 and a maximum value of 0.9697. For the minority class 0 datasets, these average values are 0.7556 (minimum), 0.8363 (mean), 0.8295 (median) and 0.9609 (maximum). In the same way, we can determine that the maximum similarity values of majority class bags with minority class bags have an average minimum value of 0.8545, average mean value of 0.9164, average median value of 0.9143 and average maximum value of 0.9697 for minority class 1 datasets. On the minority class 0 datasets, these values are 0.7234, 0.8144, 0.8121 and 0.9609 respectively. It is evident that the $C(B)$ values computed with Max can be expected to be noticeably higher for minority bags and $C = C_{maj}$ than for majority bags and $C = C_{min}$. As a result, we can expect $\underline{C}_{maj}(X)$ computed with (6.11) to be usually higher than $\underline{C}_{min}(X)$ computed with (6.12), which results in an easy assignment of the majority class label and a high accuracy on this class. This is particularly true for Max, on which the above reported values are based, but similar conclusions can be drawn for the strongly related MaxExp and MaxInvadd aggregations.

For completeness (and to show why Avg and MaxAdd are not expected to have this problem), the distributions of the average similarity values of minority class bags with majority class bags are 0.8482(min)-0.8800(mean)-0.8815(median) -0.8946(max) (minority class 1 datasets) and 0.6739(min)-0.7130(mean)-0.7126

Table 6.18 Setting rankings of the $R(X, B)$ and $C(B)$ alternatives. Weight combination $\mathcal{W}_4$ is used

$R(X, B)$	Acc maj	$C(B)$	Acc maj
AvgH	0.8713±0.1222	MaxExp	0.9706±0.0060
AvgHExp	0.8637±0.1198	Max	0.9579±0.0081
H	0.8609±0.1294	MaxInvadd	0.9540±0.0059
AvgHInvadd	0.8601±0.1201	MaxAdd	0.7432±0.0124
HInvadd	0.8545±0.1190	Avg	0.6691±0.0139
AvgHAdd	0.8545±0.1357		
HExp	0.8543±0.1359		
HAdd	0.8524±0.1349		
$R(X, B)$	Acc min	$C(B)$	Acc min
AvgH	0.5646±0.1643	Avg	0.7310±0.0179
AvgHExp	0.5399±0.1684	MaxAdd	0.7204±0.0231
AvgHInvadd	0.5181±0.1803	MaxInvadd	0.4588±0.0367
AvgHAdd	0.5062±0.1923	Max	0.3397±0.0332
HInvadd	0.5015±0.1827	MaxExp	0.3120±0.0355
HAdd	0.5000±0.1992		
HExp	0.4971±0.1841		
H	0.4717±0.1790		
$R(X, B)$	AUC	$C(B)$	AUC
AvgH	0.8605±0.0230	MaxInvadd	0.8277±0.0380
AvgHExp	0.8397±0.0155	MaxAdd	0.8076±0.0249
AvgHInvadd	0.8255±0.0162	Max	0.8034±0.0404
AvgHAdd	0.8171±0.0172	MaxExp	0.8027±0.0408
HAdd	0.7974±0.0199	Avg	0.7929±0.0240
HInvadd	0.7825±0.0158		
HExp	0.7713±0.0083		
H	0.7608±0.0059		
$R(X, B)$	Balacc	$C(B)$	Balacc
AvgH	0.7180±0.0309	MaxAdd	0.7318±0.0166
AvgHExp	0.7018±0.0339	MaxInvadd	0.7064±0.0199
AvgHInvadd	0.6891±0.0397	Avg	0.7000±0.0121
AvgHAdd	0.6803±0.0465	Max	0.6488±0.0181
HInvadd	0.6780±0.0306	MaxExp	0.6413±0.0179
HAdd	0.6762±0.0399		
HExp	0.6757±0.0294		
H	0.6663±0.0300		

Table 6.19 Evaluation of the eight weight combinations within BFRMIC-AvgH-MaxAdd-*. The value for γ in $\mathcal{W}_5$ and $\mathcal{W}_6$ is 0.1

Weights	Acc maj	Weights	Acc min	Weights	AUC	Weights	Balacc
$\mathcal{W}_3$	0.9526	$\mathcal{W}_2$	0.9873	$\mathcal{W}_4$	0.8403	$\mathcal{W}_7$	0.7637
$\mathcal{W}_4$	0.7604	$\mathcal{W}_1$	0.9716	$\mathcal{W}_7$	0.8371	$\mathcal{W}_4$	0.7608
$\mathcal{W}_7$	0.7566	$\mathcal{W}_6$	0.9574	$\mathcal{W}_8$	0.8034	$\mathcal{W}_5$	0.6954
$\mathcal{W}_5$	0.5779	$\mathcal{W}_8$	0.8996	$\mathcal{W}_5$	0.7967	$\mathcal{W}_3$	0.6749
$\mathcal{W}_8$	0.4357	$\mathcal{W}_5$	0.8129	$\mathcal{W}_3$	0.7965	$\mathcal{W}_8$	0.6677
$\mathcal{W}_6$	0.2326	$\mathcal{W}_7$	0.7709	$\mathcal{W}_6$	0.7433	$\mathcal{W}_6$	0.5950
$\mathcal{W}_1$	0.1138	$\mathcal{W}_4$	0.7613	$\mathcal{W}_1$	0.6915	$\mathcal{W}_1$	0.5427
$\mathcal{W}_2$	0.0440	$\mathcal{W}_3$	0.3972	$\mathcal{W}_2$	0.6548	$\mathcal{W}_2$	0.5156

(median)-0.7501(max) (minority class 0 datasets). The distributions of the average similarity values of majority class bags with minority class bags are 0.8135 (min)-0.8800(mean)-0.8799(median)-0.9169(max) (minority class 1 datasets) and 0.6715(min)-0.7130(mean)-0.7142(median)-0.7434(max) (minority class 0 datasets). Clearly, these values are far closer together, which can lead to more confusion between classes, but a better recognition of the minority class.

6.6.3.2 Weight Combinations

We fix $R(X, B)$ to AvgH and $C(B)$ to MaxAdd and assess the effect of the IFROWANN weight combinations in the $\underline{C}(X)$ lower approximation calculations in (6.7). Table 6.19 lists the results of this evaluation as average values across the datasets in Table 6.13. As for our IFRMIC methods, weighting schemes $\mathcal{W}_4$ and $\mathcal{W}_7$ yield the best balance between the majority and minority class accuracies, which is reflected in their high balanced accuracy results. The other schemes favour one of the two classes. Combinations $\mathcal{W}_4$ and $\mathcal{W}_7$ have the highest AUC values as well, but the results of the other alternatives are not much lower for this measure. Only $\mathcal{W}_1$ and $\mathcal{W}_2$ have a poor AUC value, which indicates that using weight vector W_{Min}^{add*} is not a good idea.

We have studied the effect of varying the γ parameter on the performance of $\mathcal{W}_5$ and $\mathcal{W}_6$ as well, which is set to 0.1 in Table 6.19. The same conclusion holds as for our IFRMIC methods (see Sect. 6.6.2.2), namely that lower γ values provide better results. In particular, a lower γ leads to (i) a higher majority class accuracy, (ii) a somewhat lower minority class accuracy, (iii) a higher AUC and (iv) a higher balanced accuracy. Setting $\gamma = 0$ yields AUC results of 0.8294 ($\mathcal{W}_5$) and 0.7826 ($\mathcal{W}_6$) and balanced accuracy values of 0.7363 ($\mathcal{W}_5$) and 0.6419 ($\mathcal{W}_6$).

Table 6.20 Performance of the BFRMIC-AvgH-Crisp-* methods with crisp class membership relation (6.8) for the training bags

Weights	Acc maj	Weights	Acc min	Weights	AUC	Weights	Balacc
$\mathcal{W}_3$	1.0000	$\mathcal{W}_2$	0.9872	$\mathcal{W}_6 - 0.0$	0.9030	$\mathcal{W}_6 - 0.0$	0.8370
$\mathcal{W}_5 - 0.0$	0.9910	$\mathcal{W}_6 - 0.1$	0.9129	$\mathcal{W}_8$	0.9006	$\mathcal{W}_8$	0.8120
$\mathcal{W}_5 - 0.1$	0.9815	$\mathcal{W}_6 - 0.0$	0.8649	$\mathcal{W}_6 - 0.1$	0.8989	$\mathcal{W}_6 - 0.1$	0.7821
$\mathcal{W}_4$	0.9696	$\mathcal{W}_1$	0.8302	$\mathcal{W}_4$	0.8909	$\mathcal{W}_7$	0.7771
$\mathcal{W}_7$	0.9505	$\mathcal{W}_8$	0.7016	$\mathcal{W}_7$	0.8837	$\mathcal{W}_4$	0.7609
$\mathcal{W}_8$	0.9225	$\mathcal{W}_7$	0.6037	$\mathcal{W}_1$	0.8813	$\mathcal{W}_1$	0.7563
$\mathcal{W}_6 - 0.0$	0.8091	$\mathcal{W}_4$	0.5523	$\mathcal{W}_5 - 0.1$	0.8790	$\mathcal{W}_5 - 0.1$	0.6646
$\mathcal{W}_1$	0.6824	$\mathcal{W}_5 - 0.1$	0.3477	$\mathcal{W}_5 - 0.0$	0.8708	$\mathcal{W}_5 - 0.0$	0.5801
$\mathcal{W}_6 - 0.1$	0.6512	$\mathcal{W}_5 - 0.0$	0.1691	$\mathcal{W}_2$	0.8692	$\mathcal{W}_2$	0.5499
$\mathcal{W}_2$	0.1127	$\mathcal{W}_3$	0.0256	$\mathcal{W}_3$	0.8342	$\mathcal{W}_3$	0.5128

6.6.3.3 Crisp Class Memberships

As argued in Sect. 6.5.1.3, it can make more intuitive sense to use crisp class membership degrees for the training bags in (6.7) instead of computing $C(B)$ with Max, MaxExp, MaxInvadd, MaxAdd or Avg. As we did in Sect. 6.6.3.2, the bag similarity values $R(X, B)$ are still computed with AvgH, the preferred alternative according to Table 6.18.

In Table 6.20, we repeat the analysis conducted in Sect. 6.6.3.2 and evaluate the performance of the different IFROWANN weight combinations within this BFRMIC method. We use two values of γ in combinations $\mathcal{W}_5$ and $\mathcal{W}_6$, namely $\gamma = 0$ and $\gamma = 0.1$. These results should be compared to the ones listed in Table 6.19, where the $C(B)$ values are computed with MaxAdd instead of the crisp class membership degrees (6.8). We observe the following:

- On average, the majority class accuracy benefits from using a crisp class relation, while the minority class accuracy decreases.
- On average, the AUC of the BFRMIC method increases when a crisp class relation is used instead of the MaxAdd setting for $C(B)$. This holds for all weight combinations. The largest increase is observed for $\mathcal{W}_2$ with a rise in mean AUC of 0.2144. The smallest increase is observed for $\mathcal{W}_3$, but a positive mean difference of 0.0377 is still reported.
- A decrease in balanced accuracy is observed for combinations $\mathcal{W}_3(-0.1621)$, $\mathcal{W}_5 - 0.0(-0.1562)$ and $\mathcal{W}_5 - 0.1(-0.0308)$. For the remaining evaluated weight combinations, an increase in balanced accuracy is obtained, ranging from a modest mean improvement of 0.0001 for $\mathcal{W}_4$ to a notable increase of 0.2136 for $\mathcal{W}_1$.
- When using weight combination $\mathcal{W}_6$ with $\gamma = 0$ within BFRMIC with the crisp class membership degree (6.8), both the highest mean AUC and highest mean balanced accuracy are obtained. In fact, this is the best mean performance of any method on the imbalanced datasets in Table 6.13 encountered so far. Note that by

setting γ to zero, the sixth weight combination reduces to $\langle W^{Maj}_{add*}, W^{Maj}_{exp*} \rangle$. As a consequence, the lower approximation values computed in (6.7) are a priori based on the same number of observations, because W^{Maj}_{add*} and W^{Maj}_{exp*} have the same number of leading zero positions (namely, $|Maj|$). The lower approximation of the minority class is obtained with additive weights and that of the majority class with exponential weights.

As a final step, we study this last observation in more detail. The weight vectors used in (6.7) when using $\mathcal{W}_6$ with $\gamma = 0$ consist of $|Maj|$ leading zeros, followed by W^{add}_L for the minority class and W^{exp}_L for the majority class, both with $p = |Min|$ (see Sect. 3.2.1 for the weight vector definitions). In Table 6.21, we evaluate the performance of this BFRMIC method when options other than W^{add}_L and W^{exp}_L are selected to fill up the weight vectors in $\mathcal{W}_6$. We compare combinations of *Strict*, *Add*, *Exp* and *Invadd*. The original definition of $\mathcal{W}_6$ with $\gamma = 0$ corresponds to *Add-Exp* in the table. This clearly is a good choice, as it attains the second highest AUC and balanced accuracy values and the third highest minority class accuracy. However, its majority class accuracy is relatively low, namely the lowest but two. The *Invadd-Exp* alternative has a considerably higher majority class accuracy (0.9059 compared to 0.8091) but lower minority class accuracy (0.7799 compared to 0.8649). Nevertheless, its mean AUC and balanced accuracy are both highest among the evaluated methods. These are also the highest mean values observed on the imbalanced datasets so far.

When we compare *Invadd-Exp* and *Add-Exp* in more detail, we observe that their difference in performance on the minority and majority classes presents itself everywhere, that is, across all datasets of Table 6.13. *Invadd-Exp* has a higher majority class accuracy than *Add-Exp* on all datasets. *Add-Exp* has a higher minority class accuracy than *Invadd-Exp* on all datasets, except for ties on WIRSel-6, Corel20-8 and Mut_bonds. We can refer the reader to Sect. 3.3.4.3 for a discussion on why the additive weights can lead to a better minority class recognition than inverse additive weights on imbalanced datasets. The AUC values of *Invadd-Exp* and *Add-Exp* are mostly close together, although the former attains 22 wins and the latter only 11, reflected in the slightly higher mean AUC for *Invadd-Exp*. With respect to the balanced accuracy, *Invadd-Exp* has the highest mean value and attains 22 wins compared to 11 wins for *Add-Exp*.

Some other aspects of note:

- The minority class accuracy increases when *Strict* or *Exp* are used for the majority class weights, regardless of the weights for the minority class. When *Invadd* or *Add* are used for the majority class weights, the majority class accuracy increases at the cost of the minority class accuracy.
- The minority class accuracy is mostly better when *Invadd* or *Add* are used for its weights.
- The settings with symmetric weights (*Exp-Exp*, *Strict-Strict*, *Add-Add*, *Invadd-Invadd*) are not the top performers, but not the worst either.

Table 6.21 Variants of $\mathcal{W}_6$-0.0 within BFRMIC-AvgH-Crisp. The first weight component refers to the weights for the minority class, the second to weights for the majority class

Combination	Acc maj	Combination	Acc min	Combination	AUC	Combination	Balacc
Exp-Add	1.0000	*Add-Strict*	0.9298	*Invadd-Exp*	0.9038	*Invadd-Exp*	0.8429
Strict-Add	1.0000	*Invadd-Strict*	0.8660	*Add-Exp*	0.9030	*Add-Exp*	0.8370
Strict-Invadd	0.9993	*Add-Exp*	0.8649	*Invadd-Strict*	0.8965	*Invadd-Strict*	0.8326
Exp-Invadd	0.9990	*Invadd-Exp*	0.7799	*Add-Strict*	0.8945	*Exp-Strict*	0.8091
Invadd-Add	0.9979	*Exp-Strict*	0.7026	*Add-Invadd*	0.8936	*Add-Strict*	0.8059
Add-Add	0.9910	*Strict-Strict*	0.6037	*Exp-Strict*	0.8918	*Strict-Strict*	0.7771
Invadd-Invadd	0.9870	*Exp-Exp*	0.5523	*Exp-Exp*	0.8909	*Exp-Exp*	0.7609
Strict-Exp	0.9847	*Add-Invadd*	0.4884	*Invadd-Invadd*	0.8856	*Add-Invadd*	0.7311
Add-Invadd	0.9738	*Strict-Exp*	0.4537	*Strict-Strict*	0.8837	*Strict-Exp*	0.7192
Exp-Exp	0.9696	*Invadd-Invadd*	0.3016	*Strict-Exp*	0.8757	*Invadd-Invadd*	0.6443
Strict-Strict	0.9505	*Add-Add*	0.1691	*Add-Add*	0.8708	*Add-Add*	0.5801
Exp-Strict	0.9156	*Exp-Invadd*	0.1327	*Exp-Invadd*	0.8627	*Exp-Invadd*	0.5658
Invadd-Exp	0.9059	*Strict-Invadd*	0.0891	*Invadd-Add*	0.8599	*Strict-Invadd*	0.5442
Add-Exp	0.8091	*Invadd-Add*	0.0822	*Strict-Invadd*	0.8444	*Invadd-Add*	0.5401
Invadd-Strict	0.7992	*Exp-Add*	0.0256	*Exp-Add*	0.8342	*Exp-Add*	0.5128
Add-Strict	0.6820	*Strict-Add*	0.0164	*Strict-Add*	0.8149	*Strict-Add*	0.5082

6.7 Global Experimental Comparison

In this section, we compare our best performing IFMIC, BFMIC, IFRMIC and BFRMIC methods among themselves and to other multi-instance classifiers. We describe the existing multi-instance classifiers used in this evaluation in Sect. 6.7.1. We evaluate the performance of the multi-instance classification algorithms on both balanced and imbalanced datasets. Section 6.7.2 focuses on the former, while the latter is studied in Sect. 6.7.3.

6.7.1 Included Methods

We select representative methods among our IFMIC, BFMIC, IFRMIC and BFRMIC families that have shown a good performance in the previous section. These are the IFMIC-Max-MaxInvadd-Avg (IFMIC) and BFMIC-AvgH-MaxExp (BFMIC) methods for our fuzzy classifiers and the IFRMIC-Max-MaxAdd-Avg-$\mathcal{W}_4$ (IFRMIC) and BFRMIC-AvgH-MaxAdd-$\mathcal{W}_4$ (BFRMIC) methods for our fuzzy rough classifiers. We also include the methods evaluated in Sect. 6.6.3.3, as they have shown an excellent performance on the imbalanced datasets. We denote them as BFRMIC-AvgH-Crisp-Add-Exp (BFRMIC-CrAE) and BFRMIC-AvgH-Crisp-Invadd-Exp (BFRMIC-CrIE).

In the comparison of our methods to existing work on the approximately balanced multi-instance datasets from Table 6.5, we include the following general multi-instance classifiers:

- **CitationKNN** [433]: this method uses the nearest neighbour approach on the level of bags. The distance between bags is measured by the Hausdorff measure. CitationKNN extends single-instance kNN by using both references and citers of a bag. The former are the nearest neighbours of a bag, while the latter are samples for which the bag under consideration is a nearest neighbour. When classifying a bag, the labels of both its references and citers are considered and the bag is assigned to the most prominent class among them. Following [130], we use two references and four citers.
- **SimpleMI** [130]: this method converts every bag to a single instance. Afterwards, it applies a traditional single-instance learner (here, C4.5) to classify these transformed elements. The representative instance of a bag is constructed by taking the geometric mean of each feature over all the instances in the bag.
- **MIWrapper** [160]: this is a wrapper approach to apply traditional, single-instance learners (here, C4.5) to multi-instance data. To classify an unseen bag, all its instances are classified by the base learner. The bag prediction is obtained by taking the arithmetic mean of the instance predictions.
- **MILES** [93]: the dataset is converted to a single-instance format, such that a single-instance classifier can be used to make bag class predictions. In the original proposal, a support vector machine was used as base classifier. A more general

version of this method was developed in [158], where AdaBoost was put forward as a strong choice for base learner. We use this method in our experimental study as well. The MILES method is a powerful representative of the embedded space paradigm. In earlier studies (e.g. [93, 157]), it has been shown to outperform other embedded multi-instance classifiers like DDSVM [94] and YARDS [157].

- **MILR** [455]: this algorithm is an extension of the traditional logistic regression classifier to the multi-instance setting. The model requires the specification of the procedure to aggregate instance-level class conditional probabilities to the level of bags. We use the arithmetic mean for this purpose, as recommended by the authors.
- **miSVM** [17]: this method is a support vector machine for multi-instance problems, using the maximum pattern margin formulation. We have set the complexity constant C to one and use a linear kernel.
- **BARTMIP** [487]: this embedded algorithm consists of two stages. First, it applies a multi-instance clustering algorithm to cluster the training bags in k clusters. Based on this procedure, every bag is mapped to a new feature space. In this space, a bag is represented by a single vector containing k features, each feature being the distance of the bag to the centroid of a constructed cluster. Single-instance support vector machines are used as the classification algorithm in the induced space. We have used the parameter settings of the original proposal.

We also consider an evaluation on the class imbalanced multi-instance datasets from Table 6.13 and include several multi-instance proposals resistant to such imbalance in this experimental study. All methods were proposed in [439, 440]:

- **BagSMOTE**: the authors of [439, 440] developed oversampling methods as extensions of the single-instance SMOTE method ([81], Sect. 4.1.2). In their BagSMOTE method, they increase the size of the minority class by artificially generating new minority bags. In InstanceSMOTE, they create new instances and add them to existing minority bags. The imbalance ratio of the dataset remains the same in the latter algorithm, but the aim is to obtain a better representation of the minority class. In their experimental study, the authors showed that their BagSMOTE algorithm yields better results than InstanceSMOTE. Both methods are preprocessing algorithms and need to be combined with a multi-instance classifier to complete the classification process. We select BagSMOTE and combine it with the MITI classifier [49] in our experiments, as recommended in [440].
- **Ab1, Ab2, Ab3, Ab4**: these methods are cost-sensitive classifiers based on the AdaBoost.M1 boosting scheme [161]. The traditional weight update formula of the latter is

$$D_{t+1} = \frac{D_t(i) K_t(X_i, y_i)}{Z_t},$$

with

$$K_t(X_i, y_i) = \exp(-\alpha_t y_i h_t(X_i)).$$

In these expressions, t is the iteration number, Z_t is a normalization factor chosen such that D_{t+1} is a probability distribution and $\alpha_t \in \mathbb{R}$ is the coefficient associated with classifier h_t, representing its weight in the final classification aggregation of the ensemble. The authors of [439, 440] introduce class-dependent costs in the weight update formula. The cost ratios are set in favour of the minority class, implying that relatively more effort is taken to correctly classify minority bags. As the real cost ratios are generally unavailable, the authors advise the heuristic use of the imbalance ratio as cost ratio. They propose four cost-sensitive boosting methods, similar to their single-instance alternatives from [390]:

- Ab1: $K_t(X_i, y_i) = \exp(-C_i \alpha_t y_i h_t(X_i))$
- Ab2: $K_t(X_i, y_i) = C_i \cdot \exp(-\alpha_t y_i h_t(X_i))$
- Ab3: $K_t(X_i, y_i) = C_i \cdot \exp(-C_i \alpha_t y_i h_t(X_i))$
- Ab4: $K_t(X_i, y_i) = C_i^2 \cdot \exp(-C_i^2 \alpha_t y_i h_t(X_i))$

The values C_i are the cost items associated with the bags. Bags of the same class are associated with the same cost.

For the sake of readability, we only list mean performance values, averaged across the datasets in Tables 6.5 and 6.13. We complement these values with the number of wins and poor results obtained by each method on the relevant group of datasets. As we did in Chap. 3, the performance of a method on a particular dataset is considered poor when it is at least 0.05 lower than the highest observed value for that dataset. We also compare the performance of the classifiers by means of a statistical analysis.

6.7.2 Balanced Data

In this section, we focus on the balanced datasets listed in Table 6.5. We use the class-wise accuracies as well as the overall accuracy as evaluation measures. In Sect. 6.7.2.1, we first compare the selected IFMIC, BFMIC, IFRMIC and BFRMIC methods among each other. We can conclude that the results of our fuzzy methods are superior to those of our fuzzy rough methods. Next, Sect. 6.7.2.2 compares the IFMIC and BFMIC methods to existing multi-instance classifiers. Note that we do not include the BFRMIC-CrAE and BFRMIC-CrIE methods in this evaluation on the balanced datasets, as they do not treat the two classes symmetrically (that is, they use different weights in the OWA aggregations of the class approximations).

6.7.2.1 Comparison of Our Methods

Table 6.22 presents the mean accuracy values for the selected IFMIC, BFMIC, IFRMIC and BFRMIC methods on the balanced datasets from Table 6.5. With respect to the overall accuracy, the BFMIC method attains the highest mean value as well as the most wins and the fewest poor results. Second place goes to the IFMIC method,

Table 6.22 Comparison of selected methods on the balanced datasets from Table 6.5

Method	Acc cl0			Acc cl1			Accuracy		
	Mean	nWins	nPoor	Mean	nWins	nPoor	Mean	nWins	nPoor
IFMIC	0.7671	9	12	**0.8513**	13	11	0.8128	12	4
BFMIC	**0.8045**	24	5	0.8311	13	13	**0.8202**	17	3
IFRMIC	0.7073	8	19	0.8448	15	10	0.7777	3	15
BFRMIC	0.7164	7	18	0.8367	8	13	0.7820	5	12

that performs at almost the same level. A notable difference between these methods is observed in their class-wise performance: the instance-based method yields better results on class 1, while the bag-based method has the better performance on class 0. The same difference in behaviour can be observed when comparing IFRMIC and BFRMIC. The fuzzy methods clearly outperform the fuzzy rough methods with respect to the class 0 accuracy and the overall accuracy, while the results for the class 1 accuracy are relatively close together.

6.7.2.2 Comparison to Other Methods

We now compare the selected IFMIC and BFMIC methods to the CitationKNN, SimpleMI, MIWrapper, MILES, MILR, miSVM and BARTMIP methods. We do not include our IFRMIC and BFRMIC methods, as the results in Table 6.22 indicate that they are not competitive with IFMIC and BFMIC on the balanced datasets.

The accuracy results are presented in Table 6.23. Our methods yield the highest overall accuracy as well as the highest class 1 accuracy (the positive class). With respect to class 0, the MILES method attains the highest mean accuracy, with BFMIC

Table 6.23 Comparison of our fuzzy multi-instance methods to existing multi-instance classifiers on the balanced datasets from Table 6.5

Method	Acc cl0			Acc cl1			Accuracy		
	Mean	nWins	nPoor	Mean	nWins	nPoor	Mean	nWins	nPoor
IFMIC	0.7671	7	21	**0.8513**	17	7	0.8128	3	10
BFMIC	0.8045	7	19	0.8311	7	18	**0.8202**	9	8
CitKNN	0.7925	5	17	0.6033	2	28	0.7046	0	22
SimpleMI	0.7964	7	19	0.7907	3	22	0.7949	6	14
MIWrapper	0.8124	7	17	0.7808	1	26	0.8014	5	14
MILES	**0.8220**	4	15	0.7888	2	21	0.8074	6	9
MILR	0.7674	1	23	0.7937	1	29	0.7895	3	18
miSVM	0.6465	3	26	0.7958	11	17	0.7369	0	27
BARTMIP	0.7346	6	22	0.8087	7	19	0.7762	6	13

Table 6.24 Statistical analysis of the accuracy results reported in Table 6.23. The p-value of the Friedman test is smaller than 0.000001, which means that significant differences are detected. P-values of the Holm post-hoc procedure that imply a significant difference with the control method at the 5% significance level are printed in boldface

Method	Friedman rank	p_{Holm}
IFMIC	4.0455	≥ 0.999999
BFMIC	3.6515	–
CitKNN	6.7727	**0.000026**
SimpleMI	4.9848	0.239839
MIWrapper	4.3636	0.872567
MILES	4.1061	≥ 0.999999
MILR	5.3333	0.075672
miSVM	7.1515	**0.000002**
BARTMIP	4.5909	0.654060

in third place and IFMIC in seventh. Aside from the highest mean overall accuracy, our BFMIC method also has the most wins and fewest poor results for this measure. We complement these results with a statistical analysis by means of the Friedman test reported in Table 6.24. Our BFMIC method is assigned the lowest Friedman rank and shown to significantly outperform the CitationKNN and miSVM methods. The results show that our BFMIC and IFMIC methods are dominant overall. They may yield slightly sub-optimal results on some datasets however, since they do not significantly outperform several competitors despite the noticeable difference in mean accuracy.

6.7.3 Imbalanced Data

We now turn our attention to the imbalanced datasets from Table 6.13. We developed our IFRMIC and BFRMIC methods as multi-instance classifiers particularly suited for binary imbalanced data. We have considerably improved these proposals by using a crisp class membership degree within BFRMIC (see Sect. 6.6.3.3). Nevertheless, we should also verify whether the use of fuzzy rough set theory (IFRMIC, BFRMIC) as opposed to fuzzy set theory (IFMIC, BFMIC) is required at all to adequately handle the class imbalance in multi-instance datasets. To this end, we compare the performance of our different proposals to each other in Sect. 6.7.3.1. We show the dominance of our BFRMIC methods with the crisp class membership relation (6.8). In Sect. 6.7.3.2, we further compare these algorithms to existing multi-instance classifiers developed for class imbalanced data.

6.7.3.1 Comparison of Our Methods

We compare the performance of four of our fuzzy rough methods and two of our fuzzy methods. The former have been developed as extensions of the IFROWANN method and should be more resistant to the imbalance between classes. However, the latter have shown a good behaviour on the balanced datasets in Sect. 6.7.2 and it should be verified where they rank in comparison to our imbalance-resistant fuzzy rough set based methods.

Table 6.25 lists the mean class-wise accuracy, the AUC and balanced accuracy of these six methods. As could be suspected from the analysis presented in Sect. 6.6, the best global results (AUC and balanced accuracy) are obtained by our BFRMIC-CrAE and BFRMIC-CrIE methods, the latter in particular. Only for the majority class accuracy are they dominated by other methods, namely IFMIC and BFMIC. However, the latter yield a poor minority class accuracy on almost all datasets. Our fuzzy multi-instance classifiers do not handle the class imbalance well and lead to too many minority class misclassifications. Our fuzzy rough methods find a better balance between the two classes. The mean class-wise accuracies of the IFRMIC and BFRMIC methods are almost the same for the minority and majority class. However,

Table 6.25 Comparison of selected methods on the imbalanced datasets from Table 6.13

Method	Acc maj			Acc min		
	Mean	nWins	nPoor	Mean	nWins	nPoor
IFMIC	0.9517	11	6	0.5742	0	28
BFMIC	**0.9696**	25	0	0.5523	0	30
IFRMIC	0.7467	0	29	0.7436	12	20
BFRMIC	0.7604	0	31	0.7613	1	25
BFRMIC-CrAE	0.8091	0	30	**0.8649**	21	4
BFRMIC-CrIE	0.9059	0	19	0.7799	2	22
Method	AUC			Balacc		
	Mean	nWins	nPoor	Mean	nWins	nPoor
IFMIC	0.8847	6	3	0.7629	2	19
BFMIC	0.8917	8	4	0.7609	2	24
IFRMIC	0.8146	1	19	0.7451	2	23
BFRMIC	0.8403	0	21	0.7608	0	25
BFRMIC-CrAE	0.9030	6	1	0.8370	10	8
BFRMIC-CrIE	**0.9038**	13	1	**0.8429**	18	4

they are outperformed by BFRMIC-CrAE and BFRMIC-CrIE on both classes, which explains the considerable difference in balanced accuracy of the latter two methods compared to the remaining four.

We must note that the relatively poor behaviour of IFMIC and BFMIC on the imbalanced datasets can not be entirely unexpected, as their internal parameters have been optimized on the balanced datasets. However, we have also compared the best performing IFMIC and BFMIC classifiers on the imbalanced datasets (having evaluated all 165 included fuzzy multi-instance methods on these datasets as well) to BFRMIC-CrAE and BFRMIC-CrIE and still observe the clear superiority of the latter two methods.

6.7.3.2 Comparison to Other Methods

Taking the conclusions of the previous section into account, we now compare our BFRMIC-CrAE and BFRMIC-CrIE methods to existing multi-instance classifiers. The Ab1, Ab2, Ab3, Ab4 and BagSMOTE methods have been developed specifically for imbalanced multi-instance data. We also include the general MIWrapper and MILES classifiers, because they perform relatively decently on our imbalanced datasets.

A summary of the results is provided in Table 6.26. It is clear that our methods (BFRMIC-CrIE in particular) outperform all others in terms of the AUC and balanced accuracy measures. This conclusion holds for both the mean performance as well as the number of wins and poor results across the 33 datasets. Moreover, based on the results of the statistical analysis with the Friedman test presented in Table 6.27, we can conclude that BFRMIC-CrIE outperforms all included existing methods with statistical significance at the 5% significance level.

From the class-wise accuracies, we can observe that the other methods are hindered more by the imbalance between the classes and favour one of the two. The two general classifiers, MIWrapper and MILES, have a high majority class accuracy and a low minority class accuracy, as can be expected for classifiers not developed with class skewness in mind. Surprisingly, the same can be observed for the Ab1 and Ab2 methods. The other two cost-sensitive boosting algorithms, Ab3 and Ab4, have an excellent performance on the minority class, but at the cost of a poor recognition of majority class observations. This is taken to the extreme in Ab4. BagSMOTE in combination with the MITI classifier performs better than the cost-sensitive boosting methods, as evidenced by its higher balanced accuracy value. Nevertheless, it is still clearly (and significantly) outperformed by our proposals. As a final note, when comparing Tables 6.25 and 6.26 we observe that our IFMIC and BFMIC methods provide higher mean AUC and balanced accuracy values than the existing classifiers as well.

Table 6.26 Comparison of our fuzzy rough multi-instance methods with a crisp class membership relation to existing multi-instance classifiers on the imbalanced datasets from Table 6.13

Method	Acc maj			Acc min		
	Mean	nWins	nPoor	Mean	nWins	nPoor
BFRMIC-CrAE	0.8091	0	32	0.8649	4	22
BFRMIC-CrIE	0.9059	0	20	0.7799	1	28
Ab1	0.9220	7	7	0.3502	2	30
Ab2	0.8077	9	6	0.3741	6	27
Ab3	0.3746	0	32	0.9216	13	8
Ab4	0.0191	0	33	**0.9840**	31	2
BagSMOTE	0.8529	0	26	0.6484	2	29
MIWrapper	**0.9766**	18	4	0.2392	0	33
MILES	0.9622	3	5	0.3963	0	33
Method	AUC			Balacc		
	Mean	nWins	nPoor	Mean	nWins	nPoor
BFRMIC-CrAE	0.9030	8	3	0.8370	11	5
BFRMIC-CrIE	**0.9038**	20	3	**0.8429**	20	3
Ab1	0.7713	0	27	0.6361	1	30
Ab2	0.7179	0	31	0.5909	0	33
Ab3	0.8273	0	25	0.6481	0	29
Ab4	0.7755	0	30	0.5015	0	32
BagSMOTE	0.7506	0	32	0.7506	1	27
MIWrapper	0.8140	4	18	0.6079	0	31
MILES	0.8626	2	15	0.6792	1	31

6.7.4 Summary

To sum up, we can confidently conclude that we have proposed competitive methods for balanced multi-instance classification. Furthermore, our BFRMIC-CrAE and BFRMIC-CrIE extensions prove to be very strong proposals for class imbalanced multi-instance data. These methods outperform all included methods with statistical significance for both the AUC and balanced accuracy.

Table 6.27 Statistical analysis of the AUC and balanced accuracy results reported in Table 6.26. The p-values of both Friedman tests are smaller than 0.000001, which means that significant differences are detected. P-values of the Holm post-hoc procedure that imply a significant difference with the control method at the 5% significance level are printed in boldface

Method	AUC		Balacc	
	Friedman rank	p_{Holm}	Friedman rank	p_{Holm}
BFRMIC-CrAE	2.1364	0.589639	1.9848	0.57423
BFRMIC-CrIE	1.7727	–	1.6061	–
Ab1	6.7879	$\leq$ **0.000001**	6.1212	$\leq$ **0.000001**
Ab2	7.6364	$\leq$ **0.000001**	6.5455	$\leq$ **0.000001**
Ab3	4.4545	**0.000209**	5.2576	$\leq$ **0.000001**
Ab4	6.4545	$\leq$ **0.000001**	8.6515	$\leq$ **0.000001**
BagSMOTE	7.6061	$\leq$ **0.000001**	3.5303	**0.008631**
MIWrapper	4.5455	**0.000156**	6.4545	$\leq$ **0.000001**
MILES	3.6061	**0.013085**	4.8485	**0.000005**

6.8 Conclusion

This chapter has discussed multi-instance classification, a prediction setting in which observations correspond to bags of instances. Class labels are only assigned to bags as a whole, while instance labels are implicit or hidden. The prediction task is to derive a class label for newly presented bags. Multi-instance classification algorithms can be divided into three groups based on which general prediction approach they follow. We discern between instance-based, bag-based and embedded methods.

We have first developed a framework of multi-instance classifiers based on fuzzy set theory. We proposed both instance-based and bag-based methods for general multi-instance classification, that is, for the classification of multi-instance datasets with a relatively balanced class distribution. We have described these algorithms and proposed various ways to set their internal parameters. In a similar fashion, we have presented two families of instance-based and bag-based classifiers for class imbalanced multi-instance data. These methods are based on fuzzy rough set theory and are extensions of the single-instance IFROWANN method. As for our fuzzy set based methods, we propose a large variety of options for the internal settings of these fuzzy rough algorithms. We have performed an extensive experimental evaluation of our fuzzy and fuzzy rough multi-instance classifiers.

We developed four groups of methods, namely (i) fuzzy instance-based, (ii) fuzzy bag-based, (iii) fuzzy rough instance-based and (iv) fuzzy rough bag-based multi-instance classifiers. The first part of the experiments conducted in this chapter consisted of a thorough internal comparison of these four groups. We evaluated the fuzzy methods on balanced multi-instance data and the fuzzy rough methods on imbalanced multi-instance data in this stage. From among the evaluated alternatives, we were

able to put forward a specific preferred parameter setting for each computation step of our algorithms. All conclusions were meticulously explained based on the nature of the multi-instance datasets.

The second part of our study consisted of a further comparison of our best-performing methods amongst each other and against existing multi-instance classifiers. This included an evaluation on both balanced and imbalanced multi-instance datasets. On balanced data, we were able to show that our fuzzy methods are highly competitive with existing proposals. With respect to class imbalanced multi-instance classification, we showed that our best fuzzy rough algorithms significantly outperform all competitors in this area. These particularly strong new methods form an important contribution of this chapter and rely on crisp rather than fuzzy bag-to-class membership degrees.

We note that we have limited the current study to two-class multi-instance data and this for both the balanced and imbalanced datasets. This has been a mainly practical consideration, as few publicly available multi-instance datasets contain more than two classes. Nevertheless, our fuzzy multi-instance classifiers are by definition not restricted to binary datasets as they make no internal assumptions on the number of classes. Naturally, our final conclusions with regard to the preferred parameter settings may change when more than two classes are present, although the observations made here should still be useful to explain the behaviour of our fuzzy multi-instance methods in a multi-class setting as well. Our fuzzy rough algorithms for imbalanced multi-instance classification were developed as extensions of the single-instance binary IFROWANN method. As a consequence, they are also limited to binary data, since they assume the presence of a single majority class and a single minority class. In the context of multi-class imbalanced multi-instance data, the same approach as presented in Chap. 4 can be followed. Using the fuzzy rough methods developed here within the OVO decomposition scheme and with the addition of our FROVOCO summary terms in the aggregation step is expected to result in a strong performance in that setting as well.

Chapter 7
Multi-label Learning

The defining characteristic of multi-label as opposed to single-label data is that each instance can belong to several classes at once. The multi-label classification task is to predict all relevant labels of a target instance. This chapter presents and experimentally evaluates our FRONEC method, the Fuzzy Rough NEighbourhood Consensus. Our proposal is a nearest neighbour based multi-label classifier that relies on fuzzy rough set theory to derive a consensus prediction between the labelsets of near neighbours of a target instance. The nearest neighbour idea has been incorporated in a variety of existing multi-label classifiers and we evaluate our method within this family. We are able to conclude its advantages on both synthetic and real-world datasets. This chapter is structured as follows. Section 7.1 provides a brief introduction to the multi-label learning domain, focusing on the components necessary for a full understanding of the remainder of the chapter. An overview of the family of nearest neighbour based multi-label classifiers is given in Sect. 7.2. Section 7.3 presents our FRONEC proposal and describes how a confident labelset prediction can be derived from neighbourhood information using fuzzy rough set operators. Our experimental study is conducted in Sect. 7.4, where we compare different variants of our proposal amongst themselves and to existing nearest neighbour based multi-label classifiers. Finally, Sect. 7.5 formulates our conclusions.

7.1 Introduction to Multi-label Learning

In a multi-label dataset, an observation can belong to several classes at the same time, that is, more than one class label can be associated with the same instance [195, 216, 488]. The total number of possible classes is known, but the number of labels per instance can differ across the dataset. Multi-label learning can be more challenging

S. Vluymans, *Dealing with Imbalanced and Weakly Labelled Data in Machine Learning using Fuzzy and Rough Set Methods*, Studies in Computational Intelligence 807, https://doi.org/10.1007/978-3-030-04663-7_7

than single-label learning or learning all possible classes independently, as there may exist correlations between certain class labels. An example of such a situation is the existence of a label hierarchy, which needs to be taken into account in the prediction process [408]. Application domains of multi-label classification include image processing (e.g. [68, 250, 256, 443]), bioinformatics (e.g. [98, 291, 428, 441]) and text categorization (e.g. [138, 249, 333]). This section serves as an introduction to the multi-label learning domain. Section 7.1.1 describes the general structure of multi-label datasets and Sect. 7.1.2 presents the field of multi-label classification.

7.1.1 Multi-label Data

In a multi-label dataset, every instance x is described by a number of input features and is associated with a set of labels (its *labelset*) instead of a single class label. When there are a total number of m possible class labels, the labelset L_x of instance x can be represented as a binary vector $L_x = \langle l_1(x), l_2(x), \ldots, l_m(x) \rangle$. A value $l_i(x) = 1$ is interpreted as the presence of label l_i for observation x, while $l_i(x) = 0$ indicates its absence. The general format of a multi-label dataset is depicted in Table 7.1. The dataset contains n elements described by d features $\{a_1, a_2, \ldots, a_d\}$. Every instance can belong to up to m classes and the possible class labels are $l_1, l_2, \ldots, l_{m-1}$ and l_m.

Several measures have been introduced to characterize multi-label datasets. The *label cardinality* of a multi-label dataset is defined as the average number of labels per instance [410]. In a traditional (single-label) dataset, the label cardinality is one. For a multi-label dataset, it exceeds this threshold. The *label density* metric is a normalized version of the label cardinality. For each instance, the percentage of active labels is computed as the ratio of its number of actual class labels to m. The density is obtained by averaging these percentages across the instances in the dataset. A third alternative is the P_{min} metric [411], defined as the percentage of instances in the dataset with only one class label. Other measures, evaluating specific characteristics of multi-label datasets, have been proposed as well and are summarized in [216].

Multi-label datasets are sensitive to class imbalance, since many of them have a high number of possible classes and a relatively low label cardinality [76]. As an alternative to the single-label imbalance ratio (see Sect. 4.1.1), three metrics to measure the degree of class imbalance in a multi-label dataset were proposed in [76]. They are based on the *IRlbl* measure. For a class label l_i, this value is defined as the ratio of the number of instances belonging to the largest class to the size of class l_i. The *IRlbl* value is one for the most appearing label and (far) exceeds one for less frequent classes. The *MaxIR* and *MeanIR* measures respectively compute the maximum and average *IRlbl* value obtained by the m possible class labels. The third measure proposed in [76] is the dispersion measure CVIR, defined as the ratio of the standard deviation of the *IRlbl* values to the *MeanIR*.

Table 7.1 General format of a multi-label dataset with n instances, d features and m possible classes

a_1	a_2	...	a_d	l_1	l_2	...	l_m
$a_1(x_1)$	$a_2(x_1)$	...	$a_d(x_1)$	$l_1(x_1)$	$l_2(x_1)$	...	$l_m(x_1)$
$a_1(x_2)$	$a_2(x_2)$	...	$a_d(x_2)$	$l_1(x_2)$	$l_2(x_2)$	...	$l_m(x_2)$
...	...	...	...	...	...	...	...
$a_1(x_n)$	$a_2(x_n)$	...	$a_d(x_n)$	$l_1(x_n)$	$l_2(x_n)$	...	$l_m(x_n)$

7.1.2 Multi-label Classification

The task of a multi-label classifier is to predict the complete labelset of a target instance. This is inherently different from single-label classification, where only one outcome label needs to be predicted. Several approaches to multi-label classification have been proposed in the literature. The recent overview book [216] discerns between three main families: the data transformation methods, method adaptation algorithms and ensemble classifiers. The former group of methods applies a transformation to the multi-label dataset, such that it degenerates to one or more easier-to-handle single-label problems on which a single-label classifier can be applied. Two well-known representatives of this family are the binary relevance (BR, [196, 485]) and label powerset (LP, [55]) transformations. BR creates m binary single-label datasets, one for each class. Each dataset contains the same instances as the original multi-label dataset, but their labelsets are transformed to a single label. For the dataset associated with class l_i, an instance x receives the label 'positive' when $l_i(x) = 1$ and 'negative' otherwise. The LP transformation on the other hand creates only one single-label dataset. Each possible labelset receives an identifier, such that labelsets that entirely coincide are associated with the same identifier. This identifier is used as the single new class label. The second family of multi-label classifiers, the method adaptation algorithms, handle the multi-label dataset directly and are often based on modifications or generalizations of existing single-label classification schemes. An example is the MLKNN method proposed in [486], which is a nearest neighbour classifier for multi-instance data. The third group consists of ensemble solutions to multi-label classification (see Sect. 2.2.7). Further subgroups consist of ensembles of binary classifiers (e.g. the ensemble of classifier chains method [363]) and ensembles based on multi-class methods (e.g. the ensemble of pruned sets method [362]).

Aside from general multi-label classification, classifiers specifically suited for imbalanced multi-label data have been developed in recent years as well (e.g. [79, 80, 428]). The approaches for dealing with class imbalance listed in Sect. 4.1.2 have been extended and implemented for multi-label data. Among the data level techniques, we can list LP based oversampling and undersampling [74], a multi-label edited nearest neighbour algorithm [75], random resampling by label [76], a multi-label SMOTE algorithm [77], resampling by decoupling highly imbalanced

labels [78, 80] and an undersampling method for text categorization data [121]. Aside from resampling methods, algorithm level [91, 212, 279, 396] and ensemble solutions [392, 393] have been proposed as well.

7.2 Nearest Neighbour Based Multi-label Classifiers

In this chapter, we focus on nearest neighbour methods for multi-instance classification. The general nearest neighbour paradigm has been recalled in Sect. 2.2.1 and the link between nearest neighbour and fuzzy rough classification approaches has been explained in Sect. 1.3.2. Several multi-label classifiers based on or extending single-label kNN have been proposed in the literature. We provide an overview in this section. In our experimental study conducted in Sect. 7.4, we select a number of these classifiers to compare against our proposed FRONEC method. This selection is made based on the popularity and performance of these methods in other experimental studies. We also take into account how many parameters need to be set by the user. The more parameters that should be set manually, the less attractive a method is for practical use. We structure our discussion as follows. Sections 7.2.1 and 7.2.2 describe basic unweighted and weighted nearest neighbour approaches to multi-label classification. Section 7.2.3 recalls the MLKNN algorithm from [486] and later modifications made to it. Finally, Sect. 7.2.4 groups miscellaneous multi-label methods incorporating the nearest neighbour idea.

7.2.1 Basic Unweighted Approaches

Two basic techniques were proposed in [385] by combining the LP and BR transformations (see Sect. 7.1.2) with the single-label kNN classifier. Their LPKNN method classifies an instance by first locating its k nearest neighbours and then predicting the most prevalent labelset among these elements. The second algorithm, the BRKNN method, is equivalent to using the single-label kNN method within the binary relevance scheme, although it computes the nearest neighbours of an instance only once instead of m times. This optimization is valid as the nearest neighbours of the target will be the same in each generated binary dataset. The classifier assigns x to a class l when this label is present in at least half of its nearest neighbours. Using this heuristic, it is possible that x is not assigned to any class at all. To address this issue, the authors of [385] developed two extensions called BRKNN-a and BRKNN-b. For each class l, these methods define the label confidence score as the percentage of the nearest neighbours of x that belong to class l. When none of the classes is present in at least half of its neighbours, BRKNN-a assigns x to the class with the highest label confidence score, while BRKNN-b assigns x to the s highest-scoring classes, where s is the average size of the labelsets of the neighbours of x. In the later work of [192],

the authors proposed the BRKNN-new algorithm as an improvement of BRKNN by considering label correlations.

The class-balanced k nearest neighbour (BKNN, [431]) and multi-label categorical kNN methods (ML-CKNN, [231]) compute class scores based on the nearest neighbours of an instance within a class and predict the classes for which the score exceeds a given threshold. The BRkSC and BRSC methods of [298] follow a similar idea, but are based on the shell-nearest neighbour algorithm [489]. The so-called class certainty factors are determined based on the approximate shell-nearest neighbours of the instance to classify. The method predicts each class for which the certainty factor is positive.

We note that the review work [216] also listed the kNNc method [69] as a multi-label nearest neighbour classifier. However, we believe that the authors of [69] used incorrect terminology in their proposal, as their method is a multi-class single-label classifier.

7.2.2 Basic Weighted Approaches

It is common practice to incorporate weights in a nearest neighbour method in order to increase the importance of certain neighbours (see Sect. 2.2.1). In [100], the authors propose a ranking-based multi-label kNN approach that assigns weights to the neighbours of an element to reflect how trustworthy their labels are. To classify an instance, its k nearest neighbours are ranked according to this trust measure and a weighted voting strategy is used to derive the labelset prediction. Similarly, the study of [453] proposed a distance-weighted multi-label k nearest neighbour algorithm called MLC-DWkNN, which assigns an instance to class l when the sum of the weights of the neighbours that belong to l exceeds the sum of the weights of the neighbours that do not. Four weighting functions were evaluated. Another weighted nearest neighbour approach to multi-label classification is the Margin-Ratio kNN method (Mr.kNN, [292]). Relevance values of training instances (computed based on a modified fuzzy c-means clustering method) are used as weights in the voting procedure of a kNN classifier.

7.2.3 MLKNN and Related Methods

The MLKNN method from [486] was introduced as one of the first lazy multi-label classifiers and remains popular to date. It makes label predictions based on the maximum a posteriori principle and the k nearest neighbours of a target instance. Simply put, it counts the occurrences of all classes among the neighbours and evaluates how likely the presence of a class label is based on these counts and the same information computed for the training elements. Class predictions are made based on the presence and number of occurrences of certain labels in the neighbourhood of the target instance.

It has been pointed out in the literature that a limitation of the MLKNN method is that it does not take label correlations into account. The DMLKNN method of [470, 474] deals with this shortcoming by considering a global maximum a posteriori rule, thereby integrating label dependencies. The FSKNN method [248] uses a fuzzy similarity measure to first group the training elements into clusters. It reduces the computational cost of the neighbour search of MLKNN, since it only uses a subset of the clusters to locate them.

7.2.4 Other Nearest Neighbour Based Multi-label Classifiers

The authors of [96] proposed the IBLR method that combines nearest neighbour based learning and logistic regression (see Sect. 2.2.6). The class labels of the neighbours of an element are used as supplementary features. One classifier (a logistic regression model) is trained for each class, but dependencies between labels are taken into account. Aside from its neighbourhood information, the IBLR+ generalization takes additional features of the target instance into account as well. In IBLR, the bias term of the logistic regression model is constant, while it depends on the target instance in the IBLR+ version.

The MLRS method was proposed in [476] as a nearest neighbour multi-label classifier based on neighbourhood rough sets [228, 464]. At training time, this method computes, for each pair of labels, the conditional probability that a label l_i occurs when the presence of label l_j is already known. Next, to classify an instance x, its k nearest neighbours are located, based on which the marginal probabilities for the presence of the class labels are determined. Finally, for each class label, the probability that x belongs to this class is calculated. As additional input, this method requires a threshold on the class probability values, above which a class label is accepted.

Two fuzzy nearest neighbour approaches [123] to multi-label classification are the FkNN [45] and Fuzzy Veristic kNN (FV-kNN, [473]) methods. The former is an adaptation of the fuzzy nearest neighbour classifier [257]. To classify an instance x, its k nearest neighbours are located and a membership degree to each class is computed based on the membership degrees of the neighbours to the classes as well as their distance to x. The FV-kNN method uses a fuzzy kNN rule based on the theory of veristic variables, which can take on several values at once with different degrees.

Other proposals include a kernel-based kNN approach for multi-label label propagation [255], the Mallows method [97] based on calibrated label ranking [171] and the Mallows model [315], the multi-label ranking method kNN-MLR* [62], the RW.kNN method [450] based on random walks in a neighbourhood graph and the evidential multi-label kNN method (EML-kNN, [471, 472]) based on a multi-label extension of the Dempster-Shafer framework.

Finally, we can refer the reader to the recent work of [364], where the authors reviewed several nearest neighbour based multi-label classifiers and proposed a new method called MLDGC based on the data gravitation based algorithm [343]. Their

proposal is compared to various representatives of the family of nearest neighbour based multi-label classifiers. The authors of this study also establish guidelines for new proposals within this classifier family, which we follow in this chapter.

7.3 Multi-label Classification Using a Fuzzy Rough Neighbourhood Consensus

We now present our proposed method that uses fuzzy rough set theory to derive a consensus prediction from the labelsets of the k nearest neighbours of a target instance. The fuzzy rough step replaces the voting procedures commonly implemented in the methods discussed in Sect. 7.2. Our proposal is called FRONEC, which refers to its use of a Fuzzy Rough NEighbourhood Consensus. Section 7.3.1 describes the general labelset prediction procedure performed by our method. It relies on instance quality measures, of which we propose three variants in Sect. 7.3.2. These quality calculations in turn require the definition of a labelset similarity relation. We discuss two alternatives in Sect. 7.3.3. To conclude, we derive the computational complexity of FRONEC in Sect. 7.3.4.

7.3.1 General FRONEC Procedure

Our method is related to the BRKNN and LPKNN methods (Sect. 7.2.1). When classifying an instance x, these algorithms locate its k nearest neighbours and derive a labelset prediction based on the labels of such near training elements. The former does so by retaining the labels that appear in at least half of the neighbours, while the latter predicts the labelset that most frequently occurs among them. We posit that fuzzy rough set theory can provide a more suitable way to construct a consensus labelset based on the k nearest neighbours of x.

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

$$POS(x) = \max_{y \in T}[\min_{z \in T}[\mathcal{I}(R(z, x), R_d(y, z)]], \tag{7.1}$$

where T is the training set and relation $R_d(\cdot, \cdot)$ expresses the labelset similarity of two elements. Expression (7.1) can be summarized as $POS(x) = \max_{y \in T}[q_x(y)]$, where $q_x(y)$ measures the quality of y relevant to x. Linking this back to the definitions provided in Sect. 3.1.1, the positive region operator locates the element y for which the membership degree of x to the lower approximation of the fuzzy concept $R_d(y, \cdot)$ is largest. This membership value is measured by $q_x(y)$. In single-instance learning, the label similarity relation is crisp, that is, two elements x and y either have the same

class label ($R_d(x, y) = 1$) or they do not ($R_d(x, y) = 0$). In this case, the definition of the positive region reduces to the membership degree of x to the lower approximation of its own decision class [107]. This value has been used as an instance quality measure in e.g. [415, 416] (see Sect. 3.5.4).

Within FRONEC, when classifying x, we use a quality metric $Q_x(\cdot)$ that is based on the definition of the positive region (7.1). Our method proceeds as follows:

1. Define the set $N(x)$ of the k nearest neighbours of x in the training set.
2. Construct the set Y that consists of elements y for which $Q_x(y)$ is largest (Sect. 7.3.2). Multiple elements y may tie for the highest value.
3. Any class that appears in at least half of the elements of Y is predicted for x.

Our experiments show that the third step is mostly superfluous, as the labelsets of all instances in Y almost always coincide. This implies that the predicted labelset is exactly encountered as such in the training set, an aspect not taken into account by nearest neighbour alternatives such as BRKNN and LPKNN. Note that the elements in Y are not required to be located in the vicinity of x (that is, $Y \nsubseteq N(x)$). Only their labelsets, which represent an appropriate consensus of those of the nearest neighbours of x, are of importance. Our experiments confirm that consensus labelsets are indeed selected that did not appear among the neighbours of target instances. The elements in Y have the largest encountered values of $Q_x(y)$, that is, the quality of their labelset with respect to x is highest. The selection of the elements with the largest Q_x values relates back to the presence of the maximum operator in (7.1).

7.3.2 *Instance Quality Measure*

The instance quality measure $Q_x(\cdot)$ used by FRONEC is based on the definition of the fuzzy rough positive region (7.1), to which we make two modifications:

1. In the interest of computational cost, we reduce the set of elements over which the minimum is taken to the k nearest neighbours of x. This is exactly the set for which we wish to find a consensus labelset.
2. We replace the minimum operator by an OWA aggregation. In this way, our quality measure can benefit from the superior robustness properties of the OWA based fuzzy rough set model (Chap. 3).

We consider three different definitions for $Q_x(\cdot)$ given the instance x to classify. The quality measure in FRONEC-1 stays closest to (7.1) and is defined as

$$Q_x^{(1)}(y) = \mathrm{OWA}_{W_L}(\{\mathcal{I}(R(z, x), R_d(y, z)) \mid z \in N(x)\}). \tag{7.2}$$

This measure is related to the OWA based fuzzy rough lower approximation (3.5). In version FRONEC-2, we consider the use of the fuzzy rough upper approximation and set

$$Q_x^{(2)}(y) = \mathrm{OWA}_{W_U}(\{\mathcal{T}(R(z, x), R_d(y, z)) \mid z \in N(x)\}). \tag{7.3}$$

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

$$Q_x^{(3)}(y) = \frac{Q_x^{(1)}(y) + Q_x^{(2)}(y)}{2}. \tag{7.4}$$

7.3.3 Labelset Similarity Relation

A crucial component of the quality calculations in (7.2–7.4) is the labelset similarity relation $R_d(\cdot, \cdot)$. In a single-label dataset, as described in Sect. 7.3.1, this relation can take on only values zero and one, reflecting whether the instances belong to the same class or not. In a multi-label dataset on the other hand, $R_d(x, y)$ should express how similar the labelsets of instances x and y are. The comparison between two labelsets L_x and L_y can be summarized in a 2x2 table:

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

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

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

$$R_d^{(1)}(x, y) = 1 - \frac{|L_x \,\Delta\, L_y|}{m}, \tag{7.5}$$

  where the Δ operator constructs the symmetric difference between its two arguments and the $|\cdot|$ operation measures the cardinality of the resulting set. This relation can take on only a limited set of values, namely those in $\left\{\frac{i}{m} \mid i = 0, 1, \ldots, m\right\}$. When using relation $R_d^{(1)}$, the values a, b, c and d in the above table are set to the number of times their corresponding combination occurs (e.g. a is the number of classes present in both L_x and L_y).
- **Label distribution based similarity relation**: in our second relation $R_d^{(2)}(\cdot, \cdot)$, values a, b, c and d are based on label distribution information. Let $P = \langle p_1, p_2, \ldots, p_m \rangle$ be the prior class probability vector, where p_i represents the ratio of training elements that belong to the ith class. Relation $R_d^{(2)}$ takes into account whether a label is common or rare. It is based on the following idea. If a rare class label is present in both labelsets, this should be rewarded more than when they both contain the same common class label. Similarly, when both sets do not contain a common class label l, this should be rewarded more than when a rare class is not present. This relation rewards unexpected behaviour (presence of rare

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

Require: Training set X, labelsets L_x and L_y, prior class probabilities $P = \langle p_1, p_2, \ldots, p_m \rangle$
Ensure: $R_d^{(2)}(L_x, L_y)$
1: $a \leftarrow 0, b \leftarrow 0, c \leftarrow 0, d \leftarrow 0$
2: **for** $l = 1, 2, \ldots, m$ **do**
3: **if** $(L_x)_i = (L_y)_i$ **then**
4: **if** $(L_x)_i = (L_y)_i = 1$ **then**
5: $a \leftarrow a + (1 - p_i)$
6: **else**
7: $d \leftarrow d + p_i$
8: **else**
9: **if** $(L_x)_i = 1$ **then**
10: $b \leftarrow b + \frac{1}{2}$
11: **else**
12: $c \leftarrow c + \frac{1}{2}$
13: $R_d(L_x, L_y) \leftarrow \frac{a+d}{a+b+c+d}$

labels, absence of common labels) more than expected behaviour. The calculation is presented in Algorithm 1. In line 5, an update of a is performed when both labelsets contain label l_i. We add $(1 - p_i)$ to the current value of a. This implies that a higher reward is given to less common labels (smaller p_i). In line 7, label l_i is absent in both labelsets. Since this is more unexpected for a common label, a higher reward is given depending on how common l_i is. In lines 10 and 12, the values of b and c are updated when a label is present in only one of the two labelsets. We add $\frac{1}{2}$ to the current value, which is the average of p_i and $(1 - p_i)$.

It should be clear from their description that the characteristics of these two labelset similarity relations are different. As an example, we evaluate the similarity between selected labelsets of a dataset with prior class probability vector

$$P = \langle 0.439, 0.234, 0.151, 0.650, 0.250, 0.202, 0.565, 0.209, 0.066, 0.049 \rangle.$$

We assess the labelset similarity of an instance x with $L_x = \langle 1, 1, 0, 1, 1, 1, 1, 1, 1, 0 \rangle$ with ten other elements of which the labelsets are listed in Table 7.2. As stated above, the Hamming based relation $R_d^{(1)}$ can only take on a limited set of values, while relation $R_d^{(2)}$ has no such prior restriction. It is important to note that labelsets that have the same value for the first relation do not necessarily coincide in the second relation. For example, consider L_{y_8} and L_{y_9}, which both only contain one label and have a Hamming based similarity of 0.3 with L_x. However, the value of the label distribution based relation of the former is almost twice as large as that of the latter. The reason is that both L_x and L_{y_8} contain the rare class label l_9, while the shared label l_4 of L_x and L_{y_9} is far more common in the dataset. We also observe large differences between some values of the two relations. We can consequently expect different classification results for our FRONEC method when either $R_d^{(1)}$ or $R_d^{(2)}$ is used.

Table 7.2 Similarity of labelset $L_x = \langle 1, 1, 0, 1, 1, 1, 1, 1, 1, 0\rangle$ with ten other labelsets according to relations $R_d^{(1)}$ and $R_d^{(2)}$

	$R_d^{(1)}$	$R_d^{(2)}$		$R_d^{(1)}$	$R_d^{(2)}$
$L_{y_1} = \langle 1, 1, 0, 1, 1, 1, 1, 1, 0, 0\rangle$	0.9000	0.9029	$L_{y_6} = \langle 1, 0, 0, 0, 0, 1, 0, 0, 0, 0\rangle$	0.4000	0.3419
$L_{y_2} = \langle 1, 1, 0, 1, 1, 0, 1, 0, 0, 0\rangle$	0.7000	0.6712	$L_{y_7} = \langle 0, 1, 0, 1, 0, 0, 0, 0, 0, 0\rangle$	0.4000	0.3049
$L_{y_3} = \langle 0, 0, 0, 1, 0, 1, 1, 1, 0, 0\rangle$	0.6000	0.5627	$L_{y_8} = \langle 0, 0, 0, 0, 0, 0, 0, 0, 1, 0\rangle$	0.3000	0.2447
$L_{y_4} = \langle 1, 1, 1, 1, 0, 0, 1, 0, 0, 0\rangle$	0.5000	0.4637	$L_{y_9} = \langle 0, 0, 0, 1, 0, 0, 0, 0, 0, 0\rangle$	0.3000	0.1357
$L_{y_5} = \langle 1, 0, 0, 1, 0, 0, 1, 0, 0, 0\rangle$	0.5000	0.3821	$L_{y_{10}} = \langle 0, 0, 0, 0, 0, 0, 0, 0, 0, 1\rangle$	0.1000	0.0324

7.3.4 Computational Complexity

Having described all components of our proposal, we are now in a position to evaluate its computational complexity. Despite its nearest neighbour aspect, which generally implies a lazy learning nature and therefore a negligible training phase, FRONEC can precompute and store all pairwise labelset similarity values between the training instances. As should be clear from the quality measure definitions (7.2–7.4), labelset similarity comparisons are only computed between training elements. This allows these values to be precomputed, such that repeated calculations are avoided at classification time. The cost of this calculation is quadratic in the number of instances, but needs to be performed only once during training. The cost of one pairwise labelset similarity calculation is linear in the number of possible labels regardless of whether $R_d^{(1)}$ or $R_d^{(2)}$ is used.

To classify an instance x, the procedure described in Sect. 7.3.1 is applied:

1. The first step is to construct the set $N(x)$ of the k nearest neighbours of x in the training set. This process has a computational cost that is linear in the number of instances and linear in the number of features. The latter is due to the cost of the feature similarity (or distance) calculation (3.13). These feature similarity values are also used in the second step. If they can be stored, repeated calculations are avoided.
2. Next, the quality value $Q_x(y)$ is calculated for every training element y. Since the quality of y depends on the target instance x, it can not be precomputed during training. As all $R(\cdot, x)$ and $R_d(\cdot, \cdot)$ similarity values have already been determined, the construction of the sets to aggregate in (7.2) and (7.3) is linear in the number of neighbours k. The OWA aggregation has a cost of $\mathcal{O}(k \log(k))$ due to the sorting step in Definition 3.1.1. Quality calculation (7.4) is slightly more costly, since it includes both (7.2) and (7.3). In particular, two sorting operations are required. In total, the second step of the classification procedure is linear in the number of instances and linearithmic in k.
3. Finally, the predicted labelset for x needs to be constructed. This requires a pass through set Y, for which the cost is linear in $|Y|$. This value is upper bounded by the total number of training instances. The presence of each class label needs to

be evaluated. The third step in the classification of x is consequently linear in the number of training instances and the number of possible class labels.

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

7.4 Experimental Study

We now proceed with an empirical evaluation of our FRONEC proposal. The experimental set-up is described in Sect. 7.4.1, listing the datasets, evaluation measures, classifiers and their parameters used in our experiments. Section 7.4.2 performs the internal comparison of the FRONEC variants. It compares the three different quality measures (see Sect. 7.3.2) and two labelset similarity relations (see Sect. 7.3.3). The comparison of FRONEC to existing nearest neighbour based multi-label classifiers is carried out in Sects. 7.4.3 and 7.4.4. The former bases its comparison on synthetic datasets, while the latter uses real-world datasets.

7.4.1 Experimental Set-Up

For the sake of clarity, we specify the different components of our experimental study in separate paragraphs below.

Datasets The majority of the publicly available multi-label datasets have a high dimensionality, which may form an issue for nearest neighbour based classifiers [41]. To avoid this problem, we use the Mldatagen generator [401] to create synthetic multi-label datasets. We have used the HyperSpheres strategy and have fixed the number of features to 20, the number of labels to 10 and the number of instances to 5000. We have created a total number of 30 datasets by varying the number of relevant, irrelevant and redundant features as well as the percentage of class noise. The names of the datasets reflect the characteristics of the attributes and the percentage of class noise. For example, dataset 'd-5-10-5-p30' has five relevant attributes, ten irrelevant attributes, five redundant attributes and thirty percent class noise. We use five-fold cross validation in all experiments. Aside from the synthetic datasets, we also include six real-world multi-label dataset with a relatively moderate dimensionality. Their characteristics are listed in Table 7.3. These datasets are used in a final comparison between our proposal and state-of-the-art nearest neighbour multi-label classifiers in Sect. 7.4.4. All datasets and used partitions are available for download at http://www.cwi.ugent.be/sarah.php.

Evaluation Measures There exists a wide spectrum of metrics to evaluate the performance of multi-label classifiers [216]. We have selected the Hamming loss, F-

Table 7.3 Real-world multi-label datasets. We list the number of instances (nInst), features (nFeat) and number of possible labels (nLab)

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

measure, recall, precision and subset accuracy measures. They are example-based and expect a strict assignment of instances to classes (that is, instead of a label ranking). This is appropriate in the context of nearest neighbour classification. Let x be a test instance in the test set Ts, L_x its true labelset and $\hat{L}_x$ the predicted labelset. The above listed evaluation measures are defined as follows:

- Hamming loss:

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

  with the Δ operator as defined in Sect. 7.3.3. The total number of prediction errors is divided by both the number of test instances $|Ts|$ and the number of possible labels m.
- F-measure:

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

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

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

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

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

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

Methods and Parameter Settings From the nearest neighbour based classifiers described in Sect. 7.2, we select the BRKNN-b, LPKNN, MLKNN, IBLR+ and MLDGC algorithms as representative methods. From a preliminary comparison, we observed that these methods have a good classification performance on the datasets included in our study. Furthermore, their operation is easy to understand and they require only the k parameter to be set by the user. The MLKNN and IBLR methods are often included in experimental comparisons of newly proposed multi-label classifiers. MLDGC is the recently proposed method from [364]. Similar to our set-up, the authors compared this method within the family of nearest neighbour multi-label classifiers. MLDGC was shown to outperform the other methods in this study, but, as noted in the conclusion of [364], it remains important to compare newly proposed nearest neighbour multi-label methods to both MLDGC and the other state-of-the-art nearest neighbour classifiers. We follow their guideline here.

Since these methods are related to the nearest neighbour classification paradigm, they all depend on the k parameter, the number of nearest neighbours used in the prediction process. In other experimental studies, its value is often set to ten. However, in our experiments, we do not fix this value beforehand, but allow the classifier to set its own k value during the training phase. A method does so by evaluating its classification performance, measured by the leave-one-out subset accuracy on the training set, for values of k between 1 and 20. It sets the parameter to the value yielding the best performance. We have opted to use the subset accuracy as evaluation measure during this process, because it is the most exact one, that is, it counts predictions that are entirely correct.

FRONEC Settings We compare six alternatives of our FRONEC method. As described in Sect. 7.3.2, versions FRONEC-1, FRONEC-2 and FRONEC-3 differ from each other by their use of the instance quality measure. Each of these methods has a choice between the two labelset similarity relations described in Sect. 7.3.3. As part of our experimental study, we compare these six alternatives amongst each other. In expressions (7.2–7.4), we use the Łukasiewicz connectives, that is, $\mathcal{I}(a, b) = \min(1, 1 - a + b)$ and $\mathcal{T}(a, b) = \max(a + b - 1, 0)$ (see Table 1.6).

The fuzzy rough operators within FRONEC depend on the definition of the feature similarity relation $R(\cdot, \cdot)$, for which we use relation (3.13). For the sake of a sensible and fair comparison, we set the distance relation $d(\cdot, \cdot)$ used by all other methods to the complement of the similarity relation, that is, $d(x, y) = 1 - R(x, y)$. This implies that the k nearest neighbours of an element x are those that are most similar to this instance according to our standard instance similarity relation.

7.4.2 FRONEC Variants

In this section, we use the 30 synthetic datasets described in Sect. 7.4.1 to compare six versions of our proposal, namely the FRONEC-1, FRONEC-2 and FRONEC-3 methods with the two labelset similarity relations $R_d^{(1)}$ and $R_d^{(2)}$. In this stage, we fix

Table 7.4 Comparison of the different FRONEC methods using the additive weighting scheme for the OWA aggregations. The value of k was set to 20

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

the value of k to 20 and select the additive OWA weighting scheme (see Sect. 3.2.1) within the instance quality calculations (7.2–7.4). We observed that these settings provide good results on average. In later sections, we use the optimization procedure described in Sect. 7.4.1 to allow FRONEC to choose the OWA weighting scheme and k value itself during the training phase.

We present the average results of the six methods taken over the 30 synthetic datasets in Table 7.4. We use the Hamming loss, F-measure and subset accuracy measures to evaluate their classification performance. Aside from these average results, we also take statistical comparisons by means of the Wilcoxon test into account.

Choice of Labelset Similarity Relation We observe that the choice of labelset similarity relation has the largest influence on the Hamming loss and F-measure metrics. This is reflected in the mean values reported in the table, but also in the results per dataset. For the Hamming loss, almost all datasets prefer the $R_d^{(1)}$ relation, while $R_d^{(2)}$ provides better F-measure values on all datasets. When comparing the results of the two labelset similarity relations with the Wilcoxon test, we observe that $R_d^{(1)}$ is always significantly better than $R_d^{(2)}$ for the Hamming loss ($R^+ = 453$, $R^- = 12$, $p = 0.000005$ for FRONEC-1; $R^+ = 465$, $R^- = 0$, $p = 0.000001$ for FRONEC-2; $R^+ = 435$, $R^- = 0$, $p = 0.000002$ for FRONEC-3), while the reverse holds for the F-measure ($R^+ = 465$, $R^- = 0$, $p = 0.000001$ for FRONEC-1; $R^+ = 464$, $R^- = 1$, $p = 0.000002$ for FRONEC-2; $R^+ = 465$, $R^- = 0$, $p = 0.000002$ for FRONEC-3). With respect to the subset accuracy, the results are less clear-cut. The majority of the datasets do prefer $R_d^{(1)}$, which is also expressed by the mean values reported in Table 7.4 and the results of the Wilcoxon test comparing $R_d^{(1)}$ to $R_d^{(2)}$ ($R^+ = 379.5$, $R^- = 85.5$, $p = 0.002353$ for FRONEC-1; $R^+ = 433$, $R^- = 32$, $p = 0.000031$ for FRONEC-2; $R^+ = 306$, $R^- = 129$, $p = 0.051715$ for FRONEC-3). Since there does not seem to be a general inclination towards one of the two labelset similarity relations, we continue to use both. It would not be prudent to discard $R_d^{(2)}$, as it significantly outperforms $R_d^{(1)}$ with respect to the F-measure.

The difference in the relation preferences can be explained by the fact that more labels are generally predicted when $R_d^{(2)}$ is used, that is, the size of the predicted labelsets is larger on average when using $R_d^{(2)}$ compared to using $R_d^{(1)}$. This will be shown and discussed in further detail in later sections. An explanation of this behaviour lies with the comparison conducted in Sect. 7.3.3. Due to its definition

and the fact that $R_d^{(1)}$ can take on only a limited set of values, less extreme differences in the values of this relation between large and small labelsets are observed. Small labelsets are not so easily dominated by larger ones, while, based on the example comparison in Sect. 7.3.3, $R_d^{(2)}$ seems to be more sensitive to the size of a labelset. Furthermore, when comparing the labelsets of two instances x and y, the situation $(L_x)_i \neq (L_y)_i$ is penalized more severely by $R_d^{(1)}$ than by $R_d^{(2)}$. As a result, the latter is naturally allowed to make more guesses in its predictions and will lead to more predicted labels. Its superior values for the F-measure show that this characteristic is not taken to the extreme (e.g. predicting every label) and a favourable balance between recall and precision is obtained. We come back to this point further on.

Choice of Quality Measure The results of the Wilcoxon tests comparing FRONEC-1, FRONEC-2 and FRONEC-3 can be found in Table 7.5. When comparing the three versions, we observe that the latter two provide better results than the former, both on average as well as based on the results per dataset and the statistical analysis. The difference between these methods lies with their use of the instance quality measure. The inclusion of the operator related to the fuzzy rough upper approximation is shown to be more beneficial than that of the one related to the lower approximation. The lower approximation relies on a fuzzy implicator $\mathcal{I}$. The quality $Q_x^{(1)}(y)$ in the prediction of x evaluates how strongly the similarity between x and its neighbours z implies the similarity of the labelsets L_y and L_z. The upper approximation on the other hand uses a t-norm $\mathcal{T}$, which is related to a conjunction. Since a t-norm is commutative, the similarity between x and its neighbours z and the similarity between L_y and L_z have equal importance. The value of $Q_x^{(2)}(y)$ aggregates the fuzzy conjunctions of these two types of similarity values. In the search of a consensus labelset among those of the nearest neighbours of x, the second approach seems more intuitive. We conclude that the FRONEC-2 alternative can be preferred overall, since it obtains good classification results and is computationally more efficient than FRONEC-3 (Sect. 7.3.4), while not being significantly inferior to the latter for all evaluation measures. Nevertheless, Table 7.5 does show that FRONEC-3 significantly outperforms FRONEC-2 with respect to the Hamming loss. If a user is particularly interested in this metric, we advise the use of FRONEC-3 instead of FRONEC-2.

7.4.3 Comparison on Synthetic Datasets

In this section, we compare the performance of FRONEC to the selected nearest neighbour based multi-label classifiers on the 30 synthetic datasets described in Sect. 7.4.1. The comparison on the real-world datasets from Table 7.3 is postponed to Sect. 7.4.4. We use version FRONEC-2, which we denote as FRONEC in the remainder of this discussion. Our method depends on two internal parameters: the number of neighbours k and the weighting scheme for the OWA aggregation in (7.3). We use the optimization procedure in Sect. 7.4.1 to let FRONEC select its

Table 7.5 Results of the Wilcoxon test comparison of the quality measures (7.2–7.4) within FRONEC for both labelset similarity relations. P-values that imply significant differences at the 5% significance level are printed in boldface

	$R_d^{(1)}$				$R_d^{(2)}$			
	Comparison	R^+	R^-	p	Comparison	R^+	R^-	p
Hloss	$Q^{(2)}$ vs $Q^{(1)}$	316.0	149.0	0.080864	$Q^{(1)}$ vs $Q^{(2)}$	243.0	192.0	0.565543
	$Q^{(3)}$ vs $Q^{(1)}$	465.0	0.0	**0.000001**	$Q^{(3)}$ vs $Q^{(1)}$	425.5	9.5	**0.000005**
	$Q^{(3)}$ vs $Q^{(2)}$	432.5	2.5	**0.000002**	$Q^{(3)}$ vs $Q^{(2)}$	465.0	0.0	**0.000001**
F	$Q^{(2)}$ vs $Q^{(1)}$	422.0	13.0	**0.000009**	$Q^{(2)}$ vs $Q^{(1)}$	477.0	18.0	**0.000009**
	$Q^{(3)}$ vs $Q^{(1)}$	407.0	58.0	**0.000297**	$Q^{(3)}$ vs $Q^{(1)}$	422.0	43.0	**0.000093**
	$Q^{(2)}$ vs $Q^{(3)}$	328.0	137.0	**0.047743**	$Q^{(2)}$ vs $Q^{(3)}$	266.5	198.5	0.475675
SubAcc	$Q^{(2)}$ vs $Q^{(1)}$	373.0	62.0	**0.000672**	$Q^{(2)}$ vs $Q^{(1)}$	370.5	94.5	**0.004289**
	$Q^{(3)}$ vs $Q^{(1)}$	453.5	11.5	**0.000005**	$Q^{(3)}$ vs $Q^{(1)}$	455.0	10.0	**0.000005**
	$Q^{(2)}$ vs $Q^{(3)}$	291.5	143.5	0.103411	$Q^{(3)}$ vs $Q^{(2)}$	288.0	177.0	0.246954

own setting during the training phase depending on the data under consideration. For k, it evaluates all natural numbers between 1 and 20. The candidate weighting schemes are the strict, additive, exponential and inverse additive weights listed in Sect. 3.2.1. We do not use our weight guidelines proposed in Chap. 3, because they are not fully relevant here. The upper approximation in (7.3) is computed over a set of k nearest neighbours instead of over a full class of training instances. Secondly, our guidelines were proposed for single-label instead of multi-label data and the conclusions formulated in Chap. 3 are based on a crisp class similarity relation.

The median k values selected by the internal optimization procedures are 18.5 (BRKNN-b), 19 (LPKNN), 17 (MLKNN), 15.5 (IBLR+), 15 (MLDGC), 19 (FRONEC-$R_d^{(1)}$) and 19 (FRONEC-$R_d^{(2)}$). The FRONEC methods most often select the additive OWA weight setting. Note that although FRONEC sets two parameters instead of one, this does not lead to an unfair comparison with the other methods. The reason is that BRKNN-b, LPKNN, MLKNN, IBLR+ and MLDGC have only one user-defined value, which is k. Furthermore, since the FRONEC methods usually select the additive OWA weighting scheme, fixing this choice would not greatly change the reported performance either. This can also be taken into account when the parameter optimization step of FRONEC is deemed too time-consuming and the user would prefer to only optimize the k parameter.

A summary of the results for the different performance metrics is presented in Table 7.6. We provide the mean value and standard deviation of the results across the 30 synthetic datasets. For each evaluation measure, the best mean result is printed in bold. Note that for the F-measure, recall, precision and subset accuracy the best result is the highest value, while for the Hamming loss this is the lowest one. For each method, we also count the number of times it yielded the best (nBest) or worst (nWorst) result on the 30 datasets.

We report the results of the statistical analysis of these values in Tables 7.7–7.8. The former presents the analysis of the Friedman test and Holm post-hoc procedure, while the latter conducts a pairwise comparison of our FRONEC methods to the other

Table 7.6 Summary of the classification results on the 30 synthetic datasets. We present the average (av) and standard deviation (stdev) across these datasets as well as the number of times a method attained the best or worst result for a measure. The best mean values are printed in bold

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

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

Table 7.7 Statistical analysis of the results summarized in Table 7.6. We use the Friedman test and the Holm post-hoc procedure. P-values of the latter implying statistical significance at the 5% significance level are printed in bold

Hamming loss ($p_{Friedman} \leq 0.000001$)			Subset accuracy ($p_{Friedman} = 0.000001$)		
Method	Rank	p_{Holm}	Method	Rank	p_{Holm}
BRKNN-b	4.5667	$\leq$**0.000001**	BRKNN-b	3.6333	**0.046302**
LPKNN	6.7000	$\leq$**0.000001**	LPKNN	5.6000	$\leq$**0.000001**
MLKNN	1.9333	0.120233	MLKNN	4.7167	**0.000126**
IBLR+	1.0667	–	IBLR+	3.5833	**0.046302**
MLDGC	4.3333	$\leq$**0.000001**	MLDGC	4.2167	**0.003643**
FRONEC-$R_d^{(1)}$	3.9333	**0.000001**	FRONEC-$R_d^{(1)}$	2.3667	–
FRONEC-$R_d^{(2)}$	5.4667	$\leq$**0.000001**	FRONEC-$R_d^{(2)}$	3.8833	**0.019635**
F-measure ($p_{Friedman} \leq 0.000001$)			Precision ($p_{Friedman} \leq 0.000001$)		
BRKNN-b	2.5833	0.135166	BRKNN-b	4.5667	$\leq$**0.000001**
LPKNN	4.7000	**0.000001**	LPKNN	6.7000	$\leq$**0.000001**
MLKNN	6.4333	$\leq$**0.000001**	MLKNN	1.8667	0.188593
IBLR+	4.6167	**0.000001**	IBLR+	1.1333	–
MLDGC	4.3167	**0.000013**	MLDGC	4.5000	$\leq$**0.000001**
FRONEC-$R_d^{(1)}$	3.6000	**0.001821**	FRONEC-$R_d^{(1)}$	3.8667	**0.000002**
FRONEC-$R_d^{(2)}$	1.7500	–	FRONEC-$R_d^{(2)}$	5.3667	$\leq$**0.000001**
Recall ($p_{Friedman} \leq 0.000001$)					
BRKNN-b	2.8667	**0.019769**			
LPKNN	3.0667	**0.014322**			
MLKNN	6.9000	$\leq$**0.000001**			
IBLR+	5.9667	$\leq$**0.000001**			
MLDGC	3.8000	**0.000193**			
FRONEC-$R_d^{(1)}$	3.8333	**0.000193**			
FRONEC-$R_d^{(2)}$	1.5667	–			

7.4.3.1 Summary of Results

Based on the results in Table 7.6 and the statistical analysis in Tables 7.7–7.8, we can observe the following:

- **Hamming loss**: the dominance of the IBLR+ method is clear. It has the lowest average result and yields the best value on 28 out of the 30 datasets. The statistical analysis in Table 7.7 shows that IBLR+ significantly outperforms all other methods except MLKNN for this evaluation measure. The results of the Wilcoxon tests confirm that IBLR+ significantly outperforms our proposal with respect to the Hamming loss. The same holds for MLKNN. MLDGC significantly outperforms FRONEC-$R_d^{(2)}$ as well. However, FRONEC-$R_d^{(1)}$ is significantly better than

Table 7.8 Pairwise comparison between FRONEC (FR) and other methods by means of the Wilcoxon test. P-values implying significant differences at the 5% significance level are printed in boldface (in favour of a FRONEC method) or underlined (in favour of some other method)

Hamming loss				Subset accuracy			
Comparison	R^+	R^-	p	Comparison	R^+	R^-	p
FR-$R_d^{(1)}$ vs BRKNN-b	303.0	162.0	0.143162	FR-$R_d^{(1)}$ vs BRKNN-b	416.5	48.5	**0.000136**
FR-$R_d^{(1)}$ vs LPKNN	465.0	0.0	**0.000002**	FR-$R_d^{(1)}$ vs LPKNN	465.0	0.0	**0.000002**
MLKNN vs FR-$R_d^{(1)}$	465.0	0.0	0.000002	FR-$R_d^{(1)}$ vs MLKNN	400.0	65.0	**0.000531**
IBLR+ vs FR-$R_d^{(1)}$	465.0	0.0	0.000002	FR-$R_d^{(1)}$ vs IBLR+	314.0	151.0	0.090014
FR-$R_d^{(1)}$ vs MLDGC	270.0	195.0	0.432080	FR-$R_d^{(1)}$ vs MLDGC	376.5	58.5	**0.000544**
BRKNN-b vs FR-$R_d^{(2)}$	287.0	178.0	0.256721	FR-$R_d^{(2)}$ vs BRKNN-b	288.0	177.0	0.249392
FR-$R_d^{(2)}$ vs LPKNN	465.0	0.0	**0.000002**	FR-$R_d^{(2)}$ vs LPKNN	454.5	10.5	**0.000004**
MLKNN vs FR-$R_d^{(2)}$	465.0	0.0	0.000002	FR-$R_d^{(2)}$ vs MLKNN	331.5	133.5	**0.040187**
IBLR+ vs FR-$R_d^{(2)}$	465.0	0.0	0.000001	FR-$R_d^{(2)}$ vs IBLR+	269.0	196.0	0.445469
MLDGC vs FR-$R_d^{(2)}$	380.0	85.0	0.002334	FR-$R_d^{(2)}$ vs MLDGC	318.0	147.0	0.075401
FR-$R_d^{(1)}$ vs FR-$R_d^{(2)}$	465.0	0.0	**0.000001**	FR-$R_d^{(1)}$ vs FR-$R_d^{(2)}$	402.0	33.0	**0.000063**
F-measure				Precision			
Comparison	R^+	R^-	p	Comparison	R^+	R^-	p
BRKNN-b vs FR-$R_d^{(1)}$	407.5	57.5	0.000296	FR-$R_d^{(1)}$ vs BRKNN-b	299.5	165.5	0.162853
FR-$R_d^{(1)}$ vs LPKNN	314.0	151.0	0.091680	FR-$R_d^{(1)}$ vs LPKNN	465.0	0.0	**0.000002**
FR-$R_d^{(1)}$ vs MLKNN	464.0	1.0	**0.000002**	MLKNN vs FR-$R_d^{(1)}$	465.0	0.0	0.000002
FR-$R_d^{(1)}$ vs IBLR+	404.0	61.0	**0.000390**	IBLR+ vs FR-$R_d^{(1)}$	465.0	0.0	0.000002
FR-$R_d^{(1)}$ vs MLDGC	368.0	97.0	**0.005039**	FR-$R_d^{(1)}$ vs MLDGC	309.0	156.0	0.113248
FR-$R_d^{(2)}$ vs BRKNN-b	294.0	171.0	0.202225	BRKNN-b vs FR-$R_d^{(2)}$	328.0	137.0	0.048318
FR-$R_d^{(2)}$ vs LPKNN	412.5	52.5	**0.000198**	FR-$R_d^{(2)}$ vs LPKNN	465.0	0.0	**0.000002**
FR-$R_d^{(2)}$ vs MLKNN	465.0	0.0	**0.000002**	MLKNN vs FR-$R_d^{(2)}$	465.0	0.0	0.000002
FR-$R_d^{(2)}$ vs IBLR+	404.0	31.0	**0.000053**	IBLR+ vs FR-$R_d^{(2)}$	465.0	0.0	0.000002
FR-$R_d^{(2)}$ vs MLDGC	465.0	0.0	**0.000002**	MLDGC vs FR-$R_d^{(2)}$	349.0	116.0	0.016106
FR-$R_d^{(2)}$ vs FR-$R_d^{(1)}$	462.0	3.0	**0.000002**	FR-$R_d^{(1)}$ vs FR-$R_d^{(2)}$	463.0	2.0	**0.000002**

(continued)

Table 7.8 (continued)

Hamming loss				Subset accuracy			
Comparison	R^+	R^-	p	Comparison	R^+	R^-	p
Recall							
Comparison	R^+	R^-	p				
BRKNN-b vs FR-$R_d^{(1)}$	394.0	71.0	0.000810				
LPKNN vs FR-$R_d^{(1)}$	371.0	94.0	0.004250				
FR-$R_d^{(1)}$ vs MLKNN	465.0	0.0	**0.000002**				
FR-$R_d^{(1)}$ vs IBLR+	464.0	1.0	**0.000002**				
FR-$R_d^{(1)}$ vs MLDGC	290.5	174.5	0.227683				
FR-$R_d^{(2)}$ vs BRKNN-b	304.0	161.0	0.137613				
FR-$R_d^{(2)}$ vs LPKNN	325.0	140.0	0.055767				
FR-$R_d^{(2)}$ vs MLKNN	465.0	0.0	**0.000002**				
FR-$R_d^{(2)}$ vs IBLR+	465.0	0.0	**0.000002**				
FR-$R_d^{(2)}$ vs MLDGC	465.0	0.0	**0.000002**				
FR-$R_d^{(2)}$ vs FR-$R_d^{(1)}$	465.0	0.0	**0.000002**				

FRONEC-$R_d^{(2)}$ as well as LPKNN for this measure. This version also outperforms BRKNN-b and MLDGC, albeit not significantly.

- **Subset accuracy**: the highest mean result is obtained by our FRONEC method using relation $R_d^{(1)}$. It also attains the most wins, namely on 15 out of the 30 datasets. This method is assigned the lowest Friedman rank in Table 7.7 and significantly outperforms all others. This is remarkable, because, as described in Sect. 7.4.1, all methods optimize this measure internally, so we could have expected their results to be competitive with each other. This is not the case and our FRONEC-$R_d^{(1)}$ method is the best general option when one is most interested in the subset accuracy measure. The dominance of FRONEC-$R_d^{(1)}$ is confirmed by the Wilcoxon tests in Table 7.8.
- **F-measure, recall and precision**: the highest average value for the F-measure is obtained by our FRONEC method using labelset similarity relation $R_d^{(2)}$. This method also obtains the most wins, namely on 13 out of the 30 datasets. It is closely followed by BRKNN-b. In the statistical analysis in Table 7.7, FRONEC-$R_d^{(2)}$ is assigned the lowest Friedman rank and is shown to significantly outperform all other methods except BRKNN-b. The same holds for the Wilcoxon test in Table 7.8.

Although the same phenomenon can be observed in the experimental results of [364], it is remarkable that the IBLR+ and MLKNN methods perform so poorly in terms of the F-measure, while they provide the best Hamming loss results. The explanation lies with their recall and precision values, which are not at all balanced. These methods yield very poor results for the former measure and very good for the latter. This means that most of their predicted labels are correct (high precision), but they fail to predict many relevant labels (low recall). The other methods (in particular FRONEC-$R_d^{(2)}$) obtain a better recall-precision balance, which is reflected in a superior F-measure.

In summary, we can conclude that FRONEC performs best for the subset accuracy, F-measure and recall, but that IBLR+ is preferred in the evaluation by the Hamming loss and precision. As should be clear from the descriptions provided in Sect. 7.4.1, the subset accuracy and recall are related measures, as are the Hamming loss and precision. Consequently, it is not surprising that a method that performs well for one measure in a pair also attains good results for the other. The F-measure evaluates the trade-off between recall and precision. Since no method dominates all others for both measures, it is important to include the F-measure as summary metric to evaluate this trade-off. The results in Tables 7.6–7.8 show that FRONEC provides the best F-measure results, which can be interpreted as the most appropriate way to balance the different behaviour measured by the recall and precision. From this observation, we can cautiously conclude that our method is the overall best one, taking all included evaluation measures into account.

The results in Table 7.8 also stress the large effect that the choice of labelset similarity relation has on the performance of FRONEC. For each evaluation measure, the difference between FRONEC-$R_d^{(1)}$ and FRONEC-$R_d^{(2)}$ is found to be significant, sometimes in favour of the former (Hamming loss, subset accuracy, precision) and sometimes in favour of the latter (F-measure, recall). The characteristics of the two labelset similarity relations have been explained in Sect. 7.4.2.

Interestingly, the MLDGC method, although recently proposed and shown in the original paper to outperform the other included methods, does not perform notably well on these datasets. The explanation lies with the characteristics of the datasets used in our study. First, we have opted to limit ourselves to relatively low dimensional datasets, as the suitability of a nearest neighbour approach decreases when the number of features increases and the locality property is lost. Since we focus on nearest neighbour related methods, we feel it is appropriate to limit the number of features. When a high dimensional dataset needs to be processed with our proposal or one of the other included methods, we advise the application of a feature selection technique prior to the classification step. Considering the results in [364], it can be observed that MLDGC obtains the most wins and highest performance differences on relatively high dimensional datasets. This indicates that this method may be more robust against the high dimensionality than other nearest neighbour based classifiers, but it does not take away from the fact that the intuition behind it is somehow lost in the process. A second dataset property that causes the inferior performance of MLDGC is the label density of our datasets (see Sect. 7.1.1). The authors of [364] acknowledged that their

method performed best on datasets with a low label density. The label density of the datasets in [364] ranges from 0.009 to 0.485, while it ranges between 0.119 and 0.429 for our synthetic datasets, with an average of 0.282. These values are relatively high compared to those in the MLDGC study, which explains the lesser performance of this method.

We also wish to stress the comparison between FRONEC on the one hand and BRKNN-b and LPKNN on the other. As noted in Sect. 7.3.1, these methods are highly related. To classify an instance, they first determine its k nearest neighbours. Next, the labelsets of these neighbours are aggregated into a prediction. BRKNN-b and LPKNN do so by considering the labelsets of the neighbours themselves, while FRONEC searches the dataset for a labelset that forms an appropriate consensus. The results in Table 7.6 show that more accurate predictions are obtained by using the fuzzy rough approach incorporated in FRONEC.

7.4.3.2 Deeper Discussion on FRONEC, IBLR+ and MLKNN

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

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

In the definition of the precision (see Sect. 7.4.1) the size of the predicted labelset appears in the denominator. When the denominator of a fraction decreases, its overall value increases. As the predicted labelset sizes are smaller for IBLR+ and MLKNN than they are for FRONEC and these values are used in the denominator of the precision definition, a higher result for the former two methods is a logical consequence.

We also note the difference between the values for FRONEC-$R_d^{(1)}$ and FRONEC-$R_d^{(2)}$, which relates back to a point made in Sect. 7.4.2. The only difference between these two methods is their labelset similarity relation. Using relation $R_d^{(2)}$ tends to result in more predictions, which is reflected in its smaller difference between the true and predicted labelset cardinalities, larger labelset sizes and, finally, its lower precision value.

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

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

7.4.3.3 Complexity Comparison

When we compare the execution time of these methods, we observe average training times of 10.3279 s (LPKNN), 15.9900 s (MLDGC), 16.9344 s (BRKNN-b), 18.4833 s (MLKNN) and 69.3622 s (IBLR+) compared to 184.7964 s and 185.1450 s for FRONEC-$R_d^{(1)}$ and FRONEC-$R_d^{(2)}$ respectively. These values are taken as averages over ten runs of the algorithms. The higher training time of FRONEC is mainly due to its optimization of an additional parameter during the procedure described in Section 7.4.1. While its competing methods only need to decide on a value for k, FRONEC also makes a choice between four candidate OWA weighting schemes. In terms of the testing time, all methods can be considered fast with average times of 2.1851 s (BRKNN-b), 1.8784 s (LPKNN), 1.8891 s (MLKNN), 2.0514 s (IBLR+), 1.8982 s (MLDGC), 6.6611 s (FRONEC-$R_d^{(1)}$) and 6.9777 s (FRONEC-$R_d^{(2)}$) to classify a full test set. The test sets of the synthetic datasets all consist of 1000 instances. A more detailed discussion on execution times can be found in Sect. 7.4.4.2, but we discuss the theoretical complexity of the methods here. Recall that the theoretical complexity of our proposal has been derived in detail in Sect. 7.3.4. We disregard the parameter optimization step in this analysis.

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

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

In [364], the reported training cost of MLDGC is $\mathcal{O}(n^2 \cdot d)$. This corresponds to locating the nearest neighbours for each training instance. However, the additional cost of the remaining internal calculations is ignored. The labelset similarity values between all pairs of training instances are derived in $\mathcal{O}(n^2 \cdot m)$, while the neighbourhood weight values can be determined at a total cost of $\mathcal{O}(n \cdot k)$. The total training cost of MLDGC is therefore $\mathcal{O}(n^2 \cdot (d + m) + n \cdot k)$.

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

Recall that FRONEC has a training cost of $\mathcal{O}(n^2 \cdot m)$. This cost is quadratic in n, as is the cost of MLKNN and MLDGC. The training cost of IBLR+ is at least quadratic in n. BRKNN-b and LPKNN have negligible training phases. These derivations are reflected in the runtime values listed above and in Sect. 7.4.4.2.

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

FRONEC has a classification cost of $\mathcal{O}(n \cdot (d + m + k \log(k)))$. The runtime comparisons show that FRONEC has the highest classification time, although it is close to that of the other methods.

7.4.4 Comparison on Real-World Datasets

Up until now, we have used 30 synthetic datasets in our experimental comparison of the BRKNN-b, LPKNN, MLKNN, IBLR+ and MLDGC methods to our FRONEC-$R_d^{(1)}$ and FRONEC-$R_d^{(2)}$ proposals. In this section, we complement this study with a comparison of these algorithms on the six real-world multi-label datasets described in Table 7.3. As before, we use them in a five-fold cross validation set-up. In Sect. 7.4.4.1, we list and discuss the complete prediction results. Section 7.4.4.2 compares the classifiers in terms of their execution times, referring back to Sect. 7.4.3.3 for a comparison of their theoretical complexity.

7.4.4.1 Prediction Performance

Tables 7.9 and 7.10 present the full classification results for the five evaluation measures. For each dataset, the best result is printed in boldface and the worst value is underlined. The most remarkable observation is the poor performance of the IBLR+ method on these datasets, which is in contrast with its good results on the synthetic datasets in Sect. 7.4.3. Moreover, as we noted and explained in Sect. 7.4.3.1, the MLDGC method does not perform as well in our study as it did in its original proposal. It is outperformed by FRONEC for all evaluation measures.

As we observed in the analysis of the synthetic datasets, our FRONEC method makes the most accurate predictions based on the most stringent evaluation measure, the subset accuracy. FRONEC also retains the best balance between the recall and precision measures, as reflected in its value for the F-measure. On the real-world datasets, LPKNN achieves a good trade-off between precision and recall as well, such that a competitive F-measure value is obtained. Looking back at Table 7.6, the precision and recall values of LPKNN on the synthetic datasets were close together as well, although they were both lower than the corresponding results of FRONEC-$R_d^{(2)}$.

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

Table 7.9 Experimental results of the seven multi-label classifiers on the datasets in Table 7.3 for the Hamming loss and subset accuracy measures

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

remaining two evaluation measures, the Hamming loss and precision, but has the lowest average result for the recall.

Linking this discussion back to the details provided in Sect. 7.4.3.2, the mean differences between the cardinality of the true and predicted labelsets are −1.0465 (BRKNN-b), 0.0706 (LPKNN), 0.3045 (MLKNN), −0.0621 (IBLR+), 0.1364 (MLDGC), 0.1267 (FRONEC-$R_d^{(1)}$) and 0.0941 (FRONEC-$R_d^{(2)}$). As before, the high value of MLKNN explains its superior precision value. Among the labels correctly predicted by MLKNN, FRONEC-$R_d^{(1)}$ and FRONEC-$R_d^{(2)}$ also derive 93.35% and 93.17% respectively, which show that the correct predictions of MLKNN are almost always made by our method as well.

7.4.4.2 Timing Comparison

To complement the prediction performance analysis, we provide an indication of the execution times of these methods both in the training and testing phases. The former includes two components: (i) the time spent to select an appropriate parameter setting and (ii) the remaining work required during training. Our FRONEC method needs to set both a value for k and an OWA weighting scheme, while the other methods

Table 7.10 Experimental results of the seven multi-label classifiers on the datasets in Table 7.3 for the F-measure, precision and recall measures

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

only need to decide their k value. This is reflected in the relatively high parameter selection time of FRONEC-$R_d^{(1)}$ (13.1974 s) and FRONEC-$R_d^{(2)}$ (13.1185 s) compared to the values of BRKNN-b (0.8369 s), LPKNN (1.0060 s), MLKNN (0.8025 s) and MLDGC (0.8503 s). However, the time spent by IBLR+ to select an appropriate k value is notably higher at 68.4973 s on average. The high computational expense of IBLR+ has been commented on in [364] as well.

Once the parameters have been set, a method may perform some additional operations during its training phase. As discussed in Sect. 7.3.4, FRONEC precomputes the labelset similarity between each pair of training instances. This is achieved in 0.0210 s

and 0.0338 s by FRONEC-$R_d^{(1)}$ and FRONEC-$R_d^{(2)}$ respectively. The MLKNN and MLDGC methods also perform some additional calculations at an average cost of 0.7598 s and 0.7933 s. IBLR+ needs to construct a binary classifier for each class, which comes at the relatively high cost of 9.3378 s. The BRKNN-b and LPKNN methods do little else during training apart from selecting their k value, such that their additional training time is negligible at 0.0023 s and 0.0010 s respectively.

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

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

7.5 Conclusion

The topic of this chapter has been multi-label classification with a specific focus on nearest neighbour based algorithms. When classifying multi-label data, the task is to predict full labelsets rather than single labels. Nearest neighbour based approaches do so by aggregating the labelset information found in the vicinity of a target instance. We have reviewed existing nearest neighbour based multi-label classifiers and have shown that this aggregation step can take on various forms.

We have presented our FRONEC method. This algorithm belongs to the family of nearest neighbour based multi-label classifiers and bases its labelset prediction on the labelsets of the k nearest neighbours of the instance to classify. Rather than using a voting procedure in this process, a consensus prediction is derived using a fuzzy rough quality measure based on the fuzzy rough positive region. Internally, the quality measure requires the definition of a labelset similarity relation, that is, a fuzzy relation that measures how similar instances are in the outcome space. In traditional single-instance classification, instances either belong to the same class or they do not. In the multi-label setting, the outcome similarity is naturally more graded, as instances belong to several classes at once. We have proposed two ways to measure the labelset similarity of instances, one based on the Hamming distance between the binary labelset vector and one based on the distribution of the possible labels in the training set.

We have proposed six versions of our FRONEC method by using combinations of three instance quality measures and the two labelset similarity relations. As a first part of our experimental study, we have compared these variants among each other. The preference for one particular quality measure (namely, the one based on the fuzzy rough upper approximation) was clear. With respect to the labelset similarity relation, the results were less clear-cut. Depending on the evaluation measure used, either one of the two dominated the other. We decided to retain both labelset similarity relations in the global evaluation, comparing two versions of FRONEC to state-of-the-art nearest neighbour based multi-label classifiers.

Following the recommendations of [364], we have restricted the comparison of our FRONEC method to previous work within the family of nearest neighbour based multi-label classification methods. We have compared FRONEC to five existing algorithms on thirty synthetic and six real-world multi-label datasets by means of their results for five evaluation measures. We were able to conclude the strong performance of our method, attaining superior results for several metrics. Its closest competitors are the IBLR+ and MLKNN methods. These algorithms provide more favourable precision and Hamming loss results, while FRONEC performs better in terms of recall and subset accuracy. Based on the results for the F-measure, which evaluates the compromise of the two performance aspects represented by these two groups of metrics, we were able to conclude that our proposal attains the best balance between them. We also showed that FRONEC detects almost all correct label predictions made by IBLR+ and MLKNN. Its sub-optimal precision and Hamming loss values are due to it generally predicting more labels for a target instance, that is, it is more liberal in its predictions compared to IBLR+ and MLKNN.

With respect to future work in this area, we would recommend a deeper study of the effects of the labelset similarity relation on the prediction performance of FRONEC. We have currently evaluated two alternative definitions and have shown their strong influence on the classification results. For each evaluation measure, the results of the two versions of FRONEC were found to be significantly different and not consistently so in favour of one of the two. Therefore, a more in-depth study of these and alternative labelset similarity relations is warranted. This may lead to additional insights into the characteristics of our FRONEC method and can result in further performance enhancements.

Chapter 8
Conclusions and Future Work

In this book, we have presented fuzzy rough set based classification methods for various challenging types of data. We have studied class imbalanced data, semi-supervised data, multi-instance data and multi-label data. Fuzzy rough set theory allows to model the uncertainty present in data both in terms of vagueness (fuzziness) and indiscernibility or imprecision (roughness). We have focussed on the OWA based fuzzy rough set model [108], a noise-tolerant generalization of traditional fuzzy rough sets [132, 358]. In this concluding chapter, we first review the work and conclusions presented in Chaps. 1–7 in Sect. 8.1. Section 8.2 maps out several directions of future research.

8.1 Overview and Conclusions of the Presented Work

Chapter 1 introduced the main topics of this work. We discussed the notion of weakly labelled data. In traditional datasets, an observation is represented by a feature vector and an associated outcome. In this book, we have focussed on classification data, where the outcome (here, a class label) is drawn from a finite set of possible categories. When dealing with weakly labelled data, the relation between a single feature vector and a class label is not as explicit. Learning from particular types of such data has formed the focus of the later chapters. The second part of Chap. 1 was devoted to the intuitive introduction of fuzzy set theory [477] and rough set theory [342]. Both model uncertainty in data, the former from the perspective of vagueness or subjectivity and the latter from the viewpoint of indiscernibility. One possible hybridization of the frameworks is fuzzy rough set theory, first introduced in [132]. We use this mathematical tool to tackle the challenges posed by different types of classification datasets.

S. Vluymans, *Dealing with Imbalanced and Weakly Labelled Data in Machine Learning using Fuzzy and Rough Set Methods*, Studies in Computational Intelligence 807, https://doi.org/10.1007/978-3-030-04663-7_8

The second part of the introduction was presented in Chap. 2, in which we reviewed the classification domain and a variety of possible approaches to the prediction of class labels. Based on a training set of labelled observations (of which both the feature values and outcome are known) a classification model is derived to use in the subsequent prediction of the outcome of unlabelled elements based on their feature values. Aside from an overview of the general ways in which such predictions can be obtained, we also recalled the proper methodology to conduct classification experiments. We discussed measures evaluating the prediction capacity of classifiers, validation techniques to test their performance on independent data and statistical tests to compare the results of several algorithms.

In Chap. 3, we studied the OWA based fuzzy rough set model from [108] in great detail. OWA based fuzzy rough sets were proposed as a generalization of traditional fuzzy rough sets with a higher robustness against noise and outliers in the data. Fuzzy rough set theory approximates a concept in two ways by means of a fuzzy rough lower approximation (conservative) and a fuzzy rough upper approximation (liberal). In the traditional model, their definition depends on the minimum and maximum operators respectively, which are sensitive to noise in the data. The superior noise robustness of OWA based fuzzy rough sets is obtained by replacing the minimum and maximum by appropriate OWA aggregations. These constructs rely on a weighting scheme to define the weights used in their aggregation procedure. In Chap. 3, focusing on the classification context, we have shown that the effectiveness of an OWA weighting scheme depends on the characteristics of the dataset at hand. Based on a thorough experimental study, we have outlined a clear weighting scheme selection strategy for both the OWA based fuzzy rough lower and upper approximation. We have validated our proposals on independent datasets and in various algorithms and have clearly demonstrated their efficacy. Aside from the improved ease of use of OWA based fuzzy rough sets resulting from our study, we have also provided further insights in the internal workings of this model.

Chapter 4 marked the beginning of our development of fuzzy rough set based classifiers for particular classification problems. This chapter studied the challenge of multi-class imbalanced data, datasets with more than two classes with a (markedly) uneven distribution of observations across them. Better represented classes are often easier to recognize than classes of which only a few observations are known at training time. The class imbalance problem indicates that the former often dominate the latter, that is, minority class instances are frequently misclassified to a majority class. Traditional classifiers yield poor minority class accuracies contrasting very strong accuracy values on the majority classes. Such discrepancy between the performance on different classes needs to be dealt with and the research community has developed specialized algorithms to address this issue. Methods dealing with class imbalance can be divided into two general categories: data level approaches and algorithm level approaches. The former modify the data to reduce the class imbalance, while the latter modify the learner to take the imbalance in its training set into account. We proposed the FROVOCO method, an algorithm for multi-class imbalanced data classification belonging to the second category. We applied the IFROWANN method, an OWA based fuzzy rough set based classifier developed for two-class imbalanced

data, within the OVO decomposition scheme. The latter is a popular methodology to reduce a multi-class classification problem to a set of two-class problems on which binary methods can be applied. The OVO method considers each pair of classes separately, applying the IFROWANN classifier to discern between them. We proposed an adaptive version of IFROWANN that selects its OWA weights based on the imbalance of each binary problem at hand. To classify an instance, the information from all induced binary classifiers is aggregated into a prediction. Our novel WV-FROST aggregation combines the traditional weighted voting aggregation step with two fuzzy rough global affinity terms of the target instances with the decision classes. In a comprehensive experimental study, we provided empirical evidence of the benefits of the two components of FROVOCO (that is, the adaptive version of IFROWANN and the WV-FROST aggregation) and have furthermore shown its superior classification performance in comparison to the state-of-the-art in multi-class imbalanced classification.

Chapter 5 focused on semi-supervised classification, the situation where a (sizeable) part of the training set is unlabelled. A semi-supervised classification algorithm consequently has both labelled and unlabelled instances at its disposal during training and can use both to construct its classification model and derive class predictions. We studied the application of our OWA based fuzzy rough classifiers proposed in Chap. 3 on such semi-supervised datasets. Our first observation was that our proposals uphold their strong prediction performance even when using only the small labelled part of the training set. Nevertheless, a classic approach to semi-supervised classification is to extend the set of labelled instances by means of self-labelling, a procedure that derives class labels for some of the originally unlabelled elements, and we wished to study the interaction of our fuzzy rough set based methods with such techniques. Our experimental study showed that our methods are in fact not benefited by prior self-labelling. Instead, the information in the originally labelled part of the training set suffices for them to derive strong class predictions. In addition, our straightforward approach clearly outperforms previously proposed semi-supervised classification algorithms that do rely on self-labelling.

Up to this point, the samples in the datasets under study have consisted of single feature vectors associated with a class label. With respect to the latter, we have considered the situation of unequal representation of different classes (Chap. 4) and missing class labels for a substantial part of the training data (Chap. 5). In the final two chapters, we examine more considerable modifications to the traditional data format. Chapter 6 focuses on multi-instance data. In this setting, each observation corresponds to a bag of instances (feature vectors) and is labelled as a whole, while no class labels are known for its individual instances. The classification task is to predict the outcome of newly presented bags based on the instances they contain. We have proposed two frameworks of multi-instance classifiers. The first group of methods are based on fuzzy set theory and interpret both bags and classes as fuzzy sets. The methods in the second group use fuzzy rough set theory and were specifically developed for class imbalanced multi-instance data, extending the single-instance IFROWANN method. The frameworks fix the general flow of the algorithms, but the internal parameters defining the precise calculations can be varied. We proposed a

range of possible settings and evaluated 165 fuzzy set based multi-instance classifiers and more than 200 fuzzy rough set based multi-instance classifiers in our experiments. In doing so, we have been able to suggest particular parameter settings leading to a strong classification performance of our methods and have explained this behaviour. We can stress that we put forward fuzzy rough set based multi-instance classifiers for imbalanced multi-instance data with an excellent performance in comparison to existing work on this topic.

In Chap. 7, we turn our attention to multi-label classification. The observations are represented as single feature vectors, but can be associated with multiple class labels. As correlations between different labels can be present, the prediction task is inherently more difficult than predicting all classes separately. We focused on nearest neighbour based approaches to multi-label classification. The labelset of a target instance is predicted based on the labels of training instances located in its vicinity. We proposed a fuzzy rough set based approach to derive a consensus prediction from the classes present in the neighbourhood of the target. Based on the labelsets of the nearest neighbours, our FRONEC method searches the training set for a labelset that constitutes an appropriate agreement between them, for which it uses a quality measure based on the fuzzy rough positive region. We experimentally evaluated the prediction strength of our proposal among the family of nearest neighbour based multi-label classifiers and were able to conclude its strong performance.

In summary, this book has focused on the development of fuzzy rough set based classification algorithms for challenging class prediction settings. We have used the noise-robust OWA based fuzzy rough sets to model data uncertainty. Apart from providing us with strong prediction algorithms, our focus on this single fuzzy rough set model has allowed us to study, interpret, understand and explain its internal workings in great detail. In this way, we have bridged the gap between the theoretical definition and practical application of OWA based fuzzy rough sets. We have considered imbalanced data, semi-supervised data, multi-instance data and multi-label data and have developed strong performing classifiers for each of these settings. Aside from their demonstrated classification strength, an evident additional advantage of all proposed algorithms is their intuitive and easy-to-understand nature.

8.2 Future Research Directions

Aside from the future endeavours directly related to the presented work and discussed in the conclusions of the relevant chapters, we can propose a number of more general topics open for exploration in prospective research on fuzzy rough set based methods in machine learning. In Sect. 8.2.1, we discuss the challenges massive training sets can pose to the presented methods and formulate our advise on how these could be handled. Section 8.2.2 introduces combinations of the different types of classification data considered in this work and explains how the proposed methods can be combined or extended to deal with multiple settings at once. Section 8.2.3 considers the challenge of high dimensional data and the robustness of our methods against

this and other data quality issues. Finally, Sects. 8.2.4 and 8.2.5 respectively discuss the dataset shift problem and, more generally, the transfer learning approach.

8.2.1 Dealing with Large to Massive Training Sets

One aspect that we have hinted at in several places (but still kept mostly under wraps) is the computational challenges associated with fuzzy rough set based methods. These algorithms revolve around lower and/or upper approximation calculations, which themselves rely on pairwise similarity values of a target instance with training elements. When the training set is large, a sequential approach to these calculations becomes too time consuming. A distributed procedure to compute the membership degree of instances to the traditional fuzzy rough set approximations was proposed in [19]. However, in the traditional model, these membership degrees are defined by means of the minimum and maximum operators, which can easily be determined in a divide-and-conquer manner. A distributed approach to OWA based fuzzy rough set calculations is more challenging due to the sorting step in the OWA aggregations. Different options could be considered, depending on whether the exact or only an approximate OWA aggregation of all relevant values is aspired.

In light of future research efforts in this topic, when the training set is (very) large, we would advise against the aim to use all training elements in the OWA based approximation calculations like we have done throughout this work for moderately-sized datasets. The sizes of the sets to aggregate would increase drastically, which affects the definitions of the weight vectors in the OWA aggregations. As discussed in Chap. 3, only the strict or exponential weighting schemes tolerate large vector lengths acceptably well, but do reduce to (weighted) nearest neighbour calculations in that situation. The interpretation of true OWA based fuzzy rough set calculations are somewhat lost as a consequence.

As we suspect that the sizes of the sets to aggregate in the fuzzy rough lower and upper approximations should be kept moderate, we would advise the exploration of two possible research lines:

1. Combination with scalable instance selection: as briefly discussed in Chap. 1, instance selection methods reduce the training set by removing redundant and/or noisy elements in a preprocessing step. The application of instance selection on large datasets requires scalable or distributed techniques. In recent years, advances in instance selection for the big data scenario [191] have for instance been made using locality-sensitive hashing based techniques [18, 277] and the MapReduce scheme [117, 407]. When the training set has been reduced to a smaller size, the methods proposed in this work can be applied. The research question would be to find an adequate instance selection method that yields small datasets with sufficient information for our fuzzy rough set based methods to be applied on.

2. Modified OWA weight vectors: dynamic (that is, target-dependent) instance selection can be carried out by actively setting certain positions in the OWA weight vectors to zero. For example, a lower approximation weight vector W_L of length $p = 10^6$ can be constructed by putting the $9,99 \cdot 10^5$ leading positions to zero and filling the remaining 1000 positions with the W_L^{invadd} definition (with $p = 1000$). The effect of the large vector length on the weight distribution would be diminished and the amount of non-zero positions and its impact on the calculations is an important property to study. However, a shortcoming of this naive approach is that only the largest or smallest values would be considered in the aggregation steps, which are possibly not varied enough to easily discern between classes. In this sense, it is related to the scalable fuzzy rough set based feature selection algorithm from [245], wherein only neighbouring elements are used in the class approximation calculations. Secondly, the computational burden to sort the large sets of values to aggregate in the OWA aggregations would not be directly removed.

In conclusion, our advise to future researchers wishing to tackle the classification with large training sets (or even big data classification) by means of OWA based fuzzy rough set theory is twofold. First, we discourage the aim to exactly replicate the algorithms presented in this book on large datasets, as the intuition and interpretation behind them will be lost in the process. Simply because our algorithms can be implemented in a distributed or iterative approach using MapReduce, does not mean that they should be. Secondly, in our opinion, the most promising area of research would be the development of a scalable instance selection technique to (i) considerably reduce a large training set to a size similar to the datasets used in the experimental evaluations in this book and (ii) yield a training set from which our algorithms can still extract sufficient information. The interaction between the instance selection and prediction steps referenced in the second point forms an interesting topic of study.

8.2.2 Data Type Combinations

As recalled in Sect. 8.1, we have studied class imbalanced data, semi-supervised data, multi-instance data and multi-label data. Chapter 6 also considered imbalanced multi-instance data and, as noted in Chap. 7, multi-label data is often class imbalanced as well. Several other combinations of these challenging data types can be studied as well:

- Multi-instance multi-label data: in the area of multi-instance multi-label classification [510], bags of instances are associated with several labels. As we have developed algorithms for both multi-instance and multi-label classification, their assimilation to deal with datasets presenting both properties is a logical next step. If more than one class label can be assigned to the same bag, the consensus approach used in FRONEC can for example be combined with our conclusions from Chap. 6

to accommodate for this possibility. We have studied a variety of ways to measure the similarity between bags as well as the membership degree of a bag to a class, which can be used in the extension of FRONEC to multi-instance data.

- Semi-supervised multi-instance and semi-supervised multi-label data: in Chap. 5, we concluded the strong performance of the fuzzy rough classifiers proposed in Chap. 3 on datasets where only a small part of the training set is labelled. Our basic classifiers were able to extract sufficient information from this labelled data to make confident predictions. As multi-instance and multi-label training sets can be partially unlabelled as well, we could verify whether the same conclusion holds for our classifiers proposed in Chaps. 6 and 7.

8.2.3 High Dimensionality Problem

The challenge of high-dimensional data has been discussed in Sect. 2.1 and is highly relevant to the fuzzy rough set based methods presented in this book, due to their strong dependence on similarity calculations between observations. The sparsity of high-dimensional spaces implies that all observations are far away from each other and we can no longer strictly speak of *(very) similar* elements, a concept on which the fuzzy rough calculations rely. The datasets included in the experimental studies conducted in this book all contain a relatively low number of features. In light of the loss of locality in high-dimensional datasets, this selection has been intentional. As any fuzzy rough set based method intrinsically relies on similarity calculations, their suitability is reduced when the dimensionality is high. The direct application of the methods proposed in this book on high-dimensional datasets requires, for instance, the prior use of a dimensionality reduction technique to bring the number of features down to an appropriate level. The synergy between our methods and such techniques forms another topic of future research.

A related point is that we have mostly fixed the fuzzy relation measuring similarity between feature vectors to expression (3.13). In Chap. 6, we have considered several ways to measure the similarity between bags, although this was again restricted to general definitions instead of a data-dependent approach. The *similarity learning* and *metric learning* domains are concerned with extracting an appropriate similarity or distance function from a dataset in order to adequately measure this relation between the observations (e.g. [34, 331, 445]). The interaction between our fuzzy rough set based algorithms and such data-dependent similarity relations remains to be examined. Many similarity learning techniques are optimization algorithms and an important question is whether an existing or custom optimization objective is required to guarantee a strong prediction performance of our classifiers.

The above two challenges can be categorized under the more general data quality problem. For example, even when the number of features in a dataset is low, there may exist some redundancy or irrelevance among them, which may again pose restrictions on the suitability of a similarity based approach. Such issues can however be resolved in a preprocessing phase [189]. In general, we would argue that our proposed methods

are more robust against quality issues in the instance space (e.g. class noise) than the feature space, as a built-in resilience against the former is obtained by our use of the OWA based fuzzy rough sets.

8.2.4 Dataset Shift Problem

We note that an assumption that we have made throughout this work is that the data distribution is the same in the source domain (∼training data) and target domain (∼test data). The dataset shift problem [329, 357] refers to the situation in which the opposite holds, namely when the training and test distributions differ. We do not build explicit classification models based on the training data, but our observation-based approaches do implicitly assume that the similarity of a target observation with training elements is highly relevant to predict its outcome. A fundamental difference in distribution may impact the validity of this assumption. The robustness of our proposals to dataset shift as well as possible solutions to counteract this issue internally form a topic for future research.

8.2.5 Transfer Learning

Apart from different training and test distributions, as discussed in the previous section, the feature space and learning task can also differ in the source and target domains. The domain of *transfer learning* (e.g. [341, 446]) studies the settings in which such disparities occur, that is, where the available target training data is limited and cross-domain information transfer from different (but related) source data is called for. It would be interesting to evaluate, both theoretically and experimentally, how strongly the fuzzy rough set based classifiers proposed in this work are hindered by differences in source and target domains and assess how existing (or novel) transfer learning techniques could be used to remedy any disturbances. In particular, again referring to the strong inherent dependence of our fuzzy rough set based classifiers on feature similarity calculations between observations, considerable changes may be required when the training and test data are described by different feature spaces, for example by relying on an intelligent feature mapping.

Bibliography

1. Abdi, L., & Hashemi, S. (2016). To combat multi-class imbalanced problems by means of over-sampling techniques. *IEEE Transactions on Knowledge and Data Engineering*, *28*(1), 238–251.
2. Adamo, J. (2012). *Data mining for association rules and sequential patterns: Sequential and parallel algorithms*. Springer Science & Business Media.
3. Adams, R., & Ghahramani, Z. (2009). Archipelago: Nonparametric Bayesian semi-supervised learning. In *Proceedings of the 26th Annual International Conference on Machine Learning* (pp. 1–8). ACM.
4. Adankon, M., & Cheriet, M. (2010). Genetic algorithm-based training for semi-supervised SVM. *Neural Computing and Applications*, *19*(8), 1197–1206.
5. Afsar Minhas, F., Ross, E., & Ben-Hur, A. (2017). Amino acid composition predicts prion activity. *PLoS Computational Biology*, *13*(4), e1005465.
6. Aggarwal, C., Hinneburg, A., & Keim, D. (2001). On the surprising behavior of distance metrics in high dimensional spaces. In *Proceedings of the 8th International Conference on Database Theory* (pp. 420–434). Springer.
7. Aggarwal, C., & Reddy, C. (2013). *Data clustering: Algorithms and applications*. CRC Press.
8. Aha, D., Kibler, D., & Albert, M. (1991). Instance-based learning algorithms. *Machine Learning*, *6*(1), 37–66.
9. Aizerman, M., Braverman, E., & Rozoner, L. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, *25*, 821–837.
10. Albalate, A., Suchindranath, A., & Minker, W. (2010). A semi-supervised cluster-and-label algorithm for utterance classification. In *Proceedings of the Intelligent Environments Workshops* (pp. 61–70).
11. Alcalá-Fdez, J., Fernández, A., Luengo, J., Derrac, J., García, S., Sánchez, L., et al. (2011). Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic & Soft Computing*, *17*(2–3), 255–287.
12. Alpaydın, E., Cheplygina, V., Loog, M., & Tax, D. (2015). Single-vs. multiple-instance classification. *Pattern Recognition*, *48*(9), 2831–2838.
13. Amores, J. (2013). Multiple instance classification: Review, taxonomy and comparative study. *Artificial Intelligence*, *201*, 81–105.
14. An, S., & Hu, Q. (2012). Fuzzy rough decision trees. In *Proceedings of the International Conference on Rough Sets and Current Trends in Computing* (pp. 397–404). Springer.
15. An, S., Hu, Q., Pedrycz, W., Zhu, P., & Tsang, E. (2016). Data-distribution-aware fuzzy rough set model and its application to robust classification. *IEEE Transactions on Cybernetics*, *46*(12), 3073–3085.

S. Vluymans, *Dealing with Imbalanced and Weakly Labelled Data in Machine Learning using Fuzzy and Rough Set Methods*, Studies in Computational Intelligence 807, https://doi.org/10.1007/978-3-030-04663-7

16. An, S., Shi, H., Hu, Q., Li, X., & Dang, J. (2014). Fuzzy rough regression with application to wind speed prediction. *Information Sciences*, *282*, 388–400.
17. Andrews, S., Tsochantaridis, I., & Hofmann, T. (2003). Support vector machines for multiple-instance learning. In *Advances in neural information processing systems* (pp. 577–584).
18. Arnaiz-González, A., Díez-Pastor, J., Rodríguez, J., & García-Osorio, C. (2016). Instance selection of linear complexity for big data. *Knowledge-Based Systems*, *107*, 83–95.
19. Asfoor, H., Srinivasan, R., Vasudevan, G., Verbiest, N., Cornells, C., Tolentino, M., et al. (2014). Computing fuzzy rough approximations in large scale information systems. In *Proceedings of the 2014 IEEE International Conference on Big Data* (pp. 9–16). IEEE.
20. Ashfaq, R., Wang, X., Huang, J., Abbas, H., & He, Y. (2017). Fuzziness based semi-supervised learning approach for intrusion detection system. *Information Sciences*, *378*, 484–497.
21. Babenko, B., Yang, M., & Belongie, S. (2011). Robust object tracking with online multiple instance learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(8), 1619–1632.
22. Bach, M., Werner, A., Żywiec, J., & Pluskiewicz, W. (2017). The study of under-and over-sampling methods' utility in analysis of highly imbalanced data on osteoporosis. *Information Sciences*, *384*, 174–190.
23. Bao, F., Deng, Y., & Dai, Q. (2016). ACID: Association correction for imbalanced data in GWAS. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *15*(1), 316–322.
24. Barandela, R., Valdovinos, R., & Sánchez, J. (2003). New applications of ensembles of classifiers. *Pattern Analysis & Applications*, *6*(3), 245–256.
25. Barber, D. (2012). *Bayesian reasoning and machine learning*. Cambridge University Press.
26. Barua, S., Islam, M., Yao, X., & Murase, K. (2014). MWMOTE-majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Transactions on Knowledge and Data Engineering*, *26*(2), 405–425.
27. Basu, S., Davidson, I., & Wagstaff, K. (2008). *Constrained clustering: Advances in algorithms, theory, and applications*. CRC Press.
28. Batista, G., Prati, R., & Monard, M. (2004). A study of the behavior of several methods for balancing machine learning training data. *ACM Sigkdd Explorations Newsletter*, *6*(1), 20–29.
29. Batuwita, R., & Palade, V. (2010). FSVM-CIL: Fuzzy support vector machines for class imbalance learning. *IEEE Transactions on Fuzzy Systems*, *18*(3), 558–571.
30. Bechar, M., Settouti, N., Barra, V., & Chikh, M. (2017). Semi-supervised superpixel classification for medical images segmentation: Application to detection of glaucoma disease. *Multidimensional Systems and Signal Processing*, 1–20.
31. Beliakov, G. (2003). How to build aggregation operators from data. *International Journal of Intelligent Systems*, *18*(8), 903–923.
32. Beliakov, G., Mesiar, R., & Valaskova, L. (2004). Fitting generated aggregation operators to empirical data. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *12*(2), 219–236.
33. Belkin, M., Niyogi, P., & Sindhwani, V. (2006). Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of Machine Learning Research*, *7*, 2399–2434.
34. Bellet, A., Habrard, A., & Sebban, M. (2015). Metric learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, *9*(1), 1–151.
35. Bellinger, C., Drummond, C., & Japkowicz, N. (2017). Manifold-based synthetic oversampling with manifold conformance estimation. *Machine Learning*, *107*(3), 605–637.
36. Bellman, R. (1957). *Dynamic programming*. Princeton University Press.
37. Ben-David, S., Lu, T., & Pál, D. (2008). Does unlabeled data provably help? Worst-case analysis of the sample complexity of semi-supervised learning. In *Proceedings of the 21st Annual Conference on Learning Theory* (pp. 33–44).
38. Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends® in Machine Learning*, *2*(1), 1–127.

39. Bennett, K., & Demiriz, A. (1999). Semi-supervised support vector machines. In *Advances in neural information processing systems* (pp. 368–374).
40. Bertsekas, D. (1999). *Nonlinear programming*. Athena Scientific Belmont.
41. Beyer, K., Goldstein, J., Ramakrishnan, R., & Shaft, U. (1999). When is nearest neighbor meaningful? In *Proceedings of the International Conference on Database Theory* (pp. 217–235). Springer.
42. Bezdek, J., Ehrlich, R., & Full, W. (1984). FCM: The fuzzy c-means clustering algorithm. *Computers & Geosciences*, *10*(2–3), 191–203.
43. Bhatt, R., & Gopal, M. (2005). On fuzzy-rough sets approach to feature selection. *Pattern Recognition Letters*, *26*(7), 965–975.
44. Bhatt, R., & Gopal, M. (2008). FRCT: Fuzzy-rough classification trees. *Pattern Analysis and Applications*, *11*(1), 73–88.
45. Bhowmick, P., Basu, A., Mitra, P., & Prasad, A. (2010). Sentence level news emotion analysis in fuzzy multi-label classification framework. *Special issue: Natural Language Processing and its Applications* (p. 143).
46. Bian, H., & Mazlack, L. (2003). Fuzzy-rough nearest-neighbor classification approach. In *Proceedings of the 22nd International Conference of the North American Fuzzy Information Processing Society* (pp. 500–505). IEEE.
47. Bishop, C. (1995). *Neural networks for pattern recognition*. Oxford University Press.
48. Bishop, C., & Lasserre, J. (2007). Generative or discriminative? Getting the best of both worlds. *Bayesian Statistics*, *8*(3), 3–24.
49. Blockeel, H., Page, D., & Srinivasan, A. (2005). Multi-instance tree learning. In *Proceedings of the 22nd International Conference on Machine learning* (pp. 57–64). ACM.
50. Blum, A., & Chawla, S. (2001). Learning from labeled and unlabeled data using graph mincuts. In *Proceedings of the 18th International Conference on Machine Learning* (pp. 19–26).
51. Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. In *Proceedings of the 11th Annual Conference on Computational Learning Theory* (pp. 92–100). ACM.
52. Boongoen, T., & Shen, Q. (2008). Clus-DOWA: A new dependent OWA operator. In *Proceedings of the 2008 IEEE International Conference on Fuzzy Systems* (pp. 1057–1063). IEEE.
53. Boongoen, T., & Shen, Q. (2010). Nearest-neighbor guided evaluation of data reliability and its applications. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *40*(6), 1622–1633.
54. Boser, B., Guyon, I., & Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual Workshop on Computational Learning Theory* (pp. 144–152). ACM.
55. Boutell, M., Luo, J., Shen, X., & Brown, C. (2004). Learning multi-label scene classification. *Pattern Recognition*, *37*(9), 1757–1771.
56. Braddock, P., Hu, D., Fan, T., Stratford, I., Harris, A., & Bicknell, R. (1994). A structure-activity analysis of antagonism of the growth factor and angiogenic activity of basic fibroblast growth factor by suramin and related polyanions. *British Journal of Cancer*, *69*(5), 890–898.
57. Branco, P., Torgo, L., & Ribeiro, R. (2016). A survey of predictive modeling on imbalanced domains. *ACM Computing Surveys*, *49*(2), 31:1–31:50.
58. Breiman, L. (1996). Bagging predictors. *Machine Learning*, *24*(2), 123–140.
59. Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.
60. Breiman, L., Friedman, J., Stone, C., & Olshen, R. (1984). *Classification and regression trees*. CRC Press.
61. Briggs, F., Lakshminarayanan, B., Neal, L., Fern, X., Raich, R., Hadley, S., et al. (2013). The 9th annual MLSP competition: New methods for acoustic classification of multiple simultaneous bird species in a noisy environment. In *Proceedings of the 2013 IEEE International Workshop on Machine Learning for Signal Processing* (pp. 1–8). IEEE.
62. Brinker, K., & Hüllermeier, E. (2007). Case-based multilabel ranking. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence* (pp. 702–707).

63. Brown, G. (2011). Ensemble learning. In *Encyclopedia of machine learning* (pp. 312–320). Springer.
64. Bunkhumpornpat, C., Sinapiromsaran, K., & Lursinsap, C. (2009). Safe-level-SMOTE: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In *Advances in knowledge discovery and data mining* (pp. 475–482).
65. Bunkhumpornpat, C., Sinapiromsaran, K., & Lursinsap, C. (2012). DBSMOTE: Density-based synthetic minority over-sampling technique. *Applied Intelligence*, *36*(3), 664–684.
66. Bunkhumpornpat, C., & Subpaiboonkit, S. (2013). Safe level graph for synthetic minority over-sampling techniques. In *Proceedings of the 13th International Symposium on Communications and Information Technologies* (pp. 570–575). IEEE.
67. Burges, C. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, *2*(2), 121–167.
68. Cabral, R., De la Torre, F., Costeira, J., & Bernardino, A. (2015). Matrix completion for weakly-supervised multi-label image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(1), 121–135.
69. Calvo-Zaragoza, J., Valero-Mas, J., & Rico-Juan, J. (2015). Improving kNN multi-label classification in prototype selection scenarios using class proposals. *Pattern Recognition*, *48*(5), 1608–1622.
70. Camps-Valls, G., Marsheva, T., & Zhou, D. (2007). Semi-supervised graph-based hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, *45*(10), 3044–3054.
71. Carneiro, G., Chan, A., Moreno, P., & Vasconcelos, N. (2007). Supervised learning of semantic classes for image annotation and retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(3), 394–410.
72. Chapelle, O., Schölkopf, B., & Zien, A. (2006). *Semi-supervised learning*. MIT Press.
73. Chapelle, O., Sindhwani, V., & Keerthi, S. S. (2008). Optimization techniques for semi-supervised support vector machines. *Journal of Machine Learning Research*, *9*, 203–233.
74. Charte, F., Rivera, A., del Jesus, M., & Herrera F. (2013). A first approach to deal with imbalance in multi-label datasets. In *Proceedings of the International Conference on Hybrid Artificial Intelligence Systems* (pp. 150–160). Springer.
75. Charte, F., Rivera, A., del Jesus, M., & Herrera, F. (2014). MLeNN: A first approach to heuristic multilabel undersampling. In *Proceedings of the International Conference on Intelligent Data Engineering and Automated Learning* (pp. 1–9). Springer.
76. Charte, F., Rivera, A., del Jesus, M., & Herrera, F. (2015a). Addressing imbalance in multilabel classification: Measures and random resampling algorithms. *Neurocomputing*, *163*, 3–16.
77. Charte, F., Rivera, A., del Jesus, M., & Herrera, F. (2015b). MLSMOTE: Approaching imbalanced multilabel learning through synthetic instance generation. *Knowledge-Based Systems*, *89*, 385–397.
78. Charte, F., Rivera, A., del Jesus, M., & Herrera, F. (2015c). Resampling multilabel datasets by decoupling highly imbalanced labels. In *Proceedings of the International Conference on Hybrid Artificial Intelligence Systems* (pp. 489–501). Springer.
79. Charte, F., Rivera, A., del Jesus, M., & Herrera, F. (2017a, in press). Dealing with difficult minority labels in imbalanced mutilabel data sets. *Neurocomputing*.
80. Charte, F., Rivera, A., del Jesus, M., & Herrera, F. (2017b, in press). REMEDIAL-HwR: Tackling multilabel imbalance through label decoupling and data resampling hybridization. *Neurocomputing*.
81. Chawla, N., Bowyer, K., Hall, L., & Kegelmeyer, W. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357.
82. Chawla, N., & Karakoulas, G. (2005). Learning from labeled and unlabeled data: An empirical study across techniques and domains. *Journal of Artificial Intelligence Research*, *23*, 331–366.
83. Chawla, N., Lazarevic, A., Hall, L., & Bowyer, K. (2003). SMOTEBoost: Improving prediction of the minority class in boosting. In *Knowledge discovery in databases* (pp. 107–119).
84. Chen, D., He, Q., & Wang, X. (2010). FRSVMs: Fuzzy rough set based support vector machines. *Fuzzy Sets and Systems*, *161*(4), 596–607.

85. Chen, D., Hu, Q., & Yang, Y. (2011). Parameterized attribute reduction with Gaussian kernel based fuzzy rough sets. *Information Sciences*, *181*(23), 5169–5179.
86. Chen, D., Kwong, S., He, Q., & Wang, H. (2012a). Geometrical interpretation and applications of membership functions with fuzzy rough sets. *Fuzzy Sets and Systems*, *193*, 122–135.
87. Chen, D., Tsang, E., & Zhao, S. (2007a). An approach of attributes reduction based on fuzzy t_l rough sets. In *Proceedings of the 2007 IEEE International Conference on Systems, Man and Cybernetics* (pp. 486–491). IEEE.
88. Chen, D., Wang, X., & Zhao, S. (2007b). Attribute reduction based on fuzzy rough sets. In *Proceedings of the 2007 International Conference on Rough Sets and Intelligent Systems Paradigms* (pp. 381–390). Springer.
89. Chen, D., Zhang, L., Zhao, S., Hu, Q., & Zhu, P. (2012b). A novel algorithm for finding reducts with fuzzy rough sets. *IEEE Transactions on Fuzzy Systems*, *20*(2), 385–389.
90. Chen, D., & Zhao, S. (2010). Local reduction of decision system with fuzzy rough sets. *Fuzzy Sets and Systems*, *161*(13), 1871–1883.
91. Chen, K., Lu, B., & Kwok, J. (2006a). Efficient classification of multi-label and imbalanced data using min-max modular classifiers. In *Proceedings of the International Joint Conference on Neural Networks* (pp. 1770–1775). IEEE.
92. Chen, Y. (2016). An empirical study of a hybrid imbalanced-class DT-RST classification procedure to elucidate therapeutic effects in uremia patients. *Medical & Biological Engineering & Computing*, *54*(6), 983–1001.
93. Chen, Y., Bi, J., & Wang, J. (2006b). MILES: Multiple-instance learning via embedded instance selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(12), 1931–1947.
94. Chen, Y., & Wang, J. (2004). Image categorization by learning and reasoning with regions. *Journal of Machine Learning Research*, *5*, 913–939.
95. Chen, Z., Lin, T., Xia, X., Xu, H., & Ding, S. (2017). A synthetic neighborhood generation based ensemble learning for the imbalanced data classification. *Applied Intelligence*, 1–17.
96. Cheng, W., & Hüllermeier, E. (2009a). Combining instance-based learning and logistic regression for multilabel classification. *Machine Learning*, *76*(2–3), 211–225.
97. Cheng, W., & Hüllermeier, E. (2009b). A simple instance-based approach to multilabel classification using the mallows model. In *Working notes of the first international workshop on learning from multi-label data* (pp. 28–38).
98. Cheng, X., Zhao, S., Xiao, X., & Chou, K. (2016). iATC-mISF: A multi-label classifier for predicting the classes of anatomical therapeutic chemicals. *Bioinformatics*, *33*(3), 341–346.
99. Cheplygina, V., Tax, D., & Loog, M. (2015). On classification with bags, groups and sets. *Pattern Recognition Letters*, *59*, 11–17.
100. Chiang, T., Lo, H., & Lin, S. (2012). A ranking-based KNN approach for multi-label classification. In *Proceedings of the Asian Conference on Machine Learning*, vol. 25 (pp. 81–96).
101. Cieslak, D., Hoens, T., Chawla, N., & Kegelmeyer, W. (2012). Hellinger distance decision trees are robust and skew-insensitive. *Data Mining and Knowledge Discovery*, *24*(1), 136–158.
102. Cios, K., & Kurgan, L. (2005). Trends in data mining and knowledge discovery. *Advanced techniques in knowledge discovery and data mining* (pp. 1–26).
103. Clark, P., & Boswell, R. (1991). Rule induction with CN2: Some recent improvements. In *Proceedings of the European Working Session on Learning* (pp. 151–163). Springer.
104. Clark, P., & Niblett, T. (1989). The CN2 induction algorithm. *Machine Learning*, *3*(4), 261–283.
105. Cohen, W. (1995). Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning* (pp. 115–123). Elsevier.
106. Cornelis, C., De Cock, M., & Radzikowska, A. (2007). Vaguely quantified rough sets. In *Proceedings of the International Workshop on Rough Sets, Fuzzy Sets, Data Mining, and Granular-Soft Computing* (pp. 87–94). Springer.
107. Cornelis, C., Jensen, R., Hurtado, G., & Ślezak, D. (2010a). Attribute selection with fuzzy decision reducts. *Information Sciences*, *180*(2), 209–224.

108. Cornelis, C., Verbiest, N., & Jensen, R. (2010b). Ordered weighted average based fuzzy rough sets. In *Proceedings of the 5th International Conference on Rough Set and Knowledge Technology* (pp. 78–85). Springer.
109. Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.
110. Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, *13*(1), 21–27.
111. Cox, D. (1958). The regression analysis of binary sequences. *Journal of the Royal Statistical Society. Series B (Methodological)*, 215–242.
112. Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge University Press.
113. Cruz, R., Sabourin, R., & Cavalcanti, G. (2018). Dynamic classifier selection: Recent advances and perspectives. *Information Fusion*, *41*, 195–216.
114. da Silva, N., Coletta, L., Hruschka, E., & Hruschka Jr, E. (2016). Using unsupervised information to improve semi-supervised tweet sentiment classification. *Information Sciences*, *355*, 348–365.
115. Dai, D., & Van Gool, L. (2013). Ensemble projection for semi-supervised image classification. In *Proceedings of the IEEE International Conference on Computer Vision* (pp. 2072–2079).
116. Dara, R., Kremer, S., & Stacey, D. (2002). Clustering unlabeled data with SOMs improves classification of labeled real-world data. In *Proceedings of the 2002 International Joint Conference on Neural Networks*, vol. 3 (pp. 2237–2242). IEEE.
117. Dean, J., & Ghemawat, S. (2008). MapReduce: Simplified data processing on large clusters. *Communications of the ACM*, *51*(1), 107–113.
118. D'eer, L., Verbiest, N., Cornelis, C., & Godo, L. (2015). A comprehensive study of implicator-conjunctor-based and noise-tolerant fuzzy rough sets: Definitions, properties and robustness analysis. *Fuzzy Sets and Systems*, *275*, 1–38.
119. Demiriz, A., Bennett, K., & Embrechts, M. (1999). Semi-supervised clustering using genetic algorithms. *Artificial Neural Networks in Engineering*, 809–814.
120. Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1–30.
121. Dendamrongvit, S., & Kubat, M. (2009). Undersampling approach for imbalanced training sets and induction from multi-label text-categorization domains. In *Proceedings of the PAKDD Workshops* (pp. 40–52). Springer.
122. Derrac, J., Cornelis, C., García, S., & Herrera, F. (2012). Enhancing evolutionary instance selection algorithms by means of fuzzy rough set based feature selection. *Information Sciences*, *186*(1), 73–92.
123. Derrac, J., García, S., & Herrera, F. (2014). Fuzzy nearest neighbor algorithms: Taxonomy, experimental analysis and prospects. *Information Sciences*, *260*, 98–119.
124. Derrac, J., García, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, *1*(1), 3–18.
125. Derrac, J., Verbiest, N., García, S., Cornelis, C., & Herrera, F. (2013). On the use of evolutionary feature selection for improving fuzzy rough set based prototype selection. *Soft Computing*, *17*(2), 223–238.
126. Diao, R., & Shen, Q. (2012). Feature selection with harmony search. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *42*(6), 1509–1523.
127. Dietterich, T., Lathrop, R., & Lozano-Pérez, T. (1997). Solving the multiple instance problem with axis-parallel rectangles. *Artificial Intelligence*, *89*(1), 31–71.
128. Díez-Pastor, J., Rodríguez, J., García-Osorio, C., & Kuncheva, L. (2015). Diversity techniques improve the performance of the best imbalance learning ensembles. *Information Sciences*, *325*, 98–117.
129. Domingos, P., & Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, *29*(2), 103–130.
130. Dong, L. (2006). *A comparison of multi-instance learning algorithms*. Ph.D. thesis, The University of Waikato, New Zealand.

131. Doostparast Torshizi, A., & Petzold, L. (2018). Graph-based semi-supervised learning with genomic data integration using condition-responsive genes applied to phenotype classification. *Journal of the American Medical Informatics Association*, *25*(1), 99–108.
132. Dubois, D., & Prade, H. (1990). Rough fuzzy sets and fuzzy rough sets. *International Journal of General System*, *17*(2–3), 191–209.
133. Dudani, S. (1976). The distance-weighted k-nearest-neighbor rule. *IEEE Transactions on Systems, Man, and Cybernetics*, *6*(4), 325–327.
134. Dundar, M., Krishnapuram, B., Rao, R., & Fung, G. (2007). Multiple instance learning for computer aided diagnosis. In *Advances in neural information processing systems* (pp. 425–432).
135. Edgar, G. (2007). *Measure, topology, and fractal geometry*. Springer Science & Business Media.
136. Efron, B., & Tibshirani, R. (1997). Improvements on cross-validation: The 632+ bootstrap method. *Journal of the American Statistical Association*, *92*(438), 548–560.
137. Elashiri, M., Hefny, H., & Abd Elwhab, A. (2012). Construct fuzzy decision trees based on roughness measures. *Social Informatics and Telecommunications Engineering*. Lecture Notes of the Institute for Computer Sciences, *108*, 199–207.
138. Elghazel, H., Aussem, A., Gharroudi, O., & Saadaoui, W. (2016). Ensemble multi-label text categorization based on rotation forest and latent semantic indexing. *Expert Systems with Applications*, *57*, 1–11.
139. Ezzat, A., Wu, M., Li, X., & Kwoh, C. (2016). Drug-target interaction prediction via class imbalance-aware ensemble learning. *BMC Bioinformatics*, *17*(19), 509.
140. Fan, W., Stolfo, S., Zhang, J., & Chan, P. (1999). AdaCost: Misclassification cost-sensitive boosting. In *Proceedings of the International Conference on Machine Learning*, vol. 99 (pp. 97–105).
141. Fawcett, T. (2006). An introduction to ROC analysis. *Pattern Recognition Letters*, *27*(8), 861–874.
142. Fei, B., & Liu, J. (2006). Binary tree of SVM: A new fast multiclass training and classification algorithm. *IEEE Transactions on Neural Networks*, *17*(3), 696–704.
143. Feng, S., Xiong, W., Li, B., Lang, C., & Huang, X. (2014). Hierarchical sparse representation based multi-instance semi-supervised learning with application to image categorization. *Signal Processing*, *94*, 595–607.
144. Fernández, A., Calderón, M., Barrenechea, E., Bustince, H., & Herrera, F. (2010). Solving multi-class problems with linguistic fuzzy rule based classification systems based on pairwise learning and preference relations. *Fuzzy Sets and Systems*, *161*(23), 3064–3080.
145. Fernández, A., Carmona, C., del Jesus, M., & Herrera, F. (2017). A Pareto-based ensemble with feature and instance selection for learning from multi-class imbalanced datasets. *International Journal of Neural Systems*, *27*(06), 1750028.
146. Fernández, A., del Jesus, M., & Herrera, F. (2009). Hierarchical fuzzy rule based classification systems with genetic rule selection for imbalanced data-sets. *International Journal of Approximate Reasoning*, *50*(3), 561–577.
147. Fernández, A., García, S., & Herrera, F. (2011). Addressing the classification with imbalanced data: Open problems and new challenges on class distribution. In *Proceedings of the International Conference on Hybrid Artificial Intelligence Systems* (pp. 1–10). Springer.
148. Fernández, A., López, V., Galar, M., del Jesus, M., & Herrera, F. (2013). Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches. *Knowledge-Based Systems*, *42*, 97–110.
149. Fernández-Navarro, F., Hervás-Martínez, C., & Gutiérrez, P. (2011). A dynamic over-sampling procedure based on sensitivity for multi-class problems. *Pattern Recognition*, *44*(8), 1821–1833.
150. Fernández-Salido, J., & Murakami, S. (2003). Rough set analysis of a general type of fuzzy data using transitive aggregations of fuzzy similarity relations. *Fuzzy Sets and Systems*, *139*(3), 635–660.

151. Ferri, C., Hernández-Orallo, J., & Salido, M. (2003). Volume under the ROC surface for multi-class problems. In *Proceedings of the European Conference on Machine Learning* (pp. 108–120). Springer.
152. Filev, D., & Yager, R. (1998). On the issue of obtaining OWA operator weights. *Fuzzy Sets and Systems*, *94*(2), 157–169.
153. Fisher, R. (1936). The use of multiple measurements in taxonomic problems. *Annals of Human Genetics*, *7*(2), 179–188.
154. Fisher, W., Camp, T., & Krzhizhanovskaya, V. (2017). Anomaly detection in earth dam and levee passive seismic data using support vector machines and automatic feature selection. *Journal of Computational Science*, *20*, 143–153.
155. Flach, P. (2012). *Machine learning: The art and science of algorithms that make sense of data*. Cambridge University Press.
156. Forestier, G., & Wemmert, C. (2016). Semi-supervised learning using multiple clusterings with limited labeled data. *Information Sciences*, *361*, 48–65.
157. Foulds, J. (2008). *Learning instance weights in multi-instance learning*. Ph.D. thesis, The University of Waikato, New Zealand.
158. Foulds, J., & Frank, E. (2008). Revisiting multiple-instance learning via embedded instance selection. In *Proceedings of the 21st Australian Joint Conference on Artificial Intelligence* (pp. 300–310). Springer.
159. Foulds, J., & Frank, E. (2010). A review of multi-instance learning assumptions. *The Knowledge Engineering Review*, *25*(1), 1–25.
160. Frank, E., & Xu, X. (2003). *Applying propositional learning algorithms to multi-instance data*. Master's thesis, The University of Waikato, New Zealand.
161. Freund, Y., & Schapire, R. (1996). Experiments with a new boosting algorithm. In *Proceedings of the 1996 International Conference on Machine Learning*, vol. 96 (pp. 148–156).
162. Freund, Y., & Schapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, *55*(1), 119–139.
163. Friedman, J. (1996). Another approach to polychotomous classification. Technical report, Department of Statistics, Stanford University.
164. Friedman, J. (1997). On bias, variance, 0/1-loss, and the curse-of-dimensionality. *Data Mining and Knowledge Discovery*, *1*(1), 55–77.
165. Friedman, M. (1937). The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, *32*(200), 675–701.
166. Fu, G., Nan, X., Liu, H., Patel, R., Daga, P., Chen, Y., et al. (2012). Implementation of multiple-instance learning in drug activity prediction. *BMC Bioinformatics*, *13*(15), S3.
167. Fujino, A., Ueda, N., & Saito, K. (2008). Semisupervised learning for a hybrid generative/discriminative classifier based on the maximum entropy principle. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *30*(3), 424–437.
168. Fullér, R., & Majlender, P. (2001). An analytic approach for obtaining maximal entropy OWA operator weights. *Fuzzy Sets and Systems*, *124*(1), 53–57.
169. Funahashi, K. (1989). On the approximate realization of continuous mappings by neural networks. *Neural Networks*, *2*(3), 183–192.
170. Fürnkranz, J., Gamberger, D., & Lavrač, N. (2012). *Foundations of rule learning*. Springer Science & Business Media.
171. Fürnkranz, J., Hüllermeier, E., Mencía, E., & Brinker, K. (2008). Multilabel classification via calibrated label ranking. *Machine Learning*, *73*(2), 133–153.
172. Fürnkranz, J., Hüllermeier, E., & Vanderlooy, S. (2009). Binary decomposition methods for multipartite ranking. *Machine Learning and Knowledge Discovery in Databases*, 359–374.
173. Galar, M., Fernández, A., Barrenechea, E., Bustince, H., & Herrera, F. (2011). An overview of ensemble methods for binary classifiers in multi-class problems: Experimental study on one-vs-one and one-vs-all schemes. *Pattern Recognition*, *44*(8), 1761–1776.
174. Galar, M., Fernández, A., Barrenechea, E., Bustince, H., & Herrera, F. (2012). A review on ensembles for the class imbalance problem: Bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *42*(4), 463–484.

175. Galar, M., Fernández, A., Barrenechea, E., Bustince, H., & Herrera, F. (2013a). Dynamic classifier selection for one-vs-one strategy: Avoiding non-competent classifiers. *Pattern Recognition*, *46*(12), 3412–3424.
176. Galar, M., Fernández, A., Barrenechea, E., Bustince, H., & Herrera, F. (2017). NMC: Nearest matrix classification-a new combination model for pruning one-vs-one ensembles by transforming the aggregation problem. *Information Fusion*, *36*, 26–51.
177. Galar, M., Fernández, A., Barrenechea, E., & Herrera, F. (2013b). EUSBoost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling. *Pattern Recognition*, *46*(12), 3460–3471.
178. Galar, M., Fernández, A., Barrenechea, E., & Herrera, F. (2014). Empowering difficult classes with a similarity-based aggregation in multi-class classification problems. *Information Sciences*, *264*, 135–157.
179. Galar, M., Fernández, A., Barrenechea, E., & Herrera, F. (2015). DRCW-OVO: Distance-based relative competence weighting combination for one-vs-one strategy in multi-class problems. *Pattern Recognition*, *48*(1), 28–42.
180. Ganivada, A., Dutta, S., & Pal, S. (2011a). Fuzzy rough granular neural networks, fuzzy granules, and classification. *Theoretical Computer Science*, *412*(42), 5834–5853.
181. Ganivada, A., & Pal, S. (2011). A novel fuzzy rough granular neural network for classification. *International Journal of Computational Intelligence Systems*, *4*(5), 1042–1051.
182. Ganivada, A., Ray, S., & Pal, S. (2011b). Fuzzy rough granular self organizing map. In *Proceedings of the International Conference on Rough Sets and Knowledge Technology* (pp. 659–668). Springer.
183. Ganivada, A., Ray, S., & Pal, S. (2012). Fuzzy rough granular self-organizing map and fuzzy rough entropy. *Theoretical Computer Science*, *466*, 37–63.
184. Ganivada, A., Ray, S., & Pal, S. (2013). Fuzzy rough sets, and a granular neural network for unsupervised feature selection. *Neural Networks*, *48*, 91–108.
185. Gao, Q., Huang, Y., Gao, X., Shen, W., & Zhang, H. (2015). A novel semi-supervised learning for face recognition. *Neurocomputing*, *152*, 69–76.
186. Gao, Y., Ma, J., & Yuille, A. (2017). Semi-supervised sparse representation based classification for face recognition with insufficient labeled samples. *IEEE Transactions on Image Processing*, *26*(5), 2545–2560.
187. García, S., Fernández, A., Luengo, J., & Herrera, F. (2009). A study of statistical techniques and performance measures for genetics-based machine learning: Accuracy and interpretability. *Soft Computing*, *13*(10), 959–977.
188. García, S., & Herrera, F. (2009). Evolutionary undersampling for classification with imbalanced datasets: Proposals and taxonomy. *Evolutionary Computation*, *17*(3), 275–306.
189. García, S., Luengo, J., & Herrera, F. (2015). *Data preprocessing in data mining*. Springer.
190. García, S., Luengo, J., Sáez, J., Lopez, V., & Herrera, F. (2013). A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. *IEEE Transactions on Knowledge and Data Engineering*, *25*(4), 734–750.
191. García, S., Ramírez-Gallego, S., Luengo, J., Benítez, J., & Herrera, F. (2016). Big data preprocessing: Methods and prospects. *Big Data Analytics*, *1*(1), 9.
192. Geng, X., Tang, Y., Zhu, Y., & Cheng, G. (2014). An improved multi-label classification algorithm BRkNN. *Journal of Information and Computational Science*, *11*(16), 5927–5936.
193. Ghazikhani, A., Monsefi, R., & Yazdi, H. (2013). Online cost-sensitive neural network classifiers for non-stationary and imbalanced data streams. *Neural Computing and Applications*, *23*(5), 1283–1295.
194. Giacinto, G., & Roli, F. (2000). Dynamic classifier selection. In *Proceedings of the International Workshop on Multiple Classifier Systems* (pp. 177–189). Springer.
195. Gibaja, E., & Ventura, S. (2014). Multi-label learning: A review of the state of the art and ongoing research. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *4*(6), 411–444.
196. Godbole, S., & Sarawagi, S. (2004). Discriminative methods for multi-labeled classification. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 22–30). Springer.

197. Gong, C., Tao, D., Maybank, S., Liu, W., Kang, G., & Yang, J. (2016). Multi-modal curriculum learning for semi-supervised image classification. *IEEE Transactions on Image Processing*, *25*(7), 3249–3260.
198. Gónzalez, S., García, S., Lázaro, M., Figueiras-Vidal, A., & Herrera, F. (2017). Class switching according to nearest enemy distance for learning from highly imbalanced data-sets. *Pattern Recognition*, *70*, 12–24.
199. Grandvalet, Y., & Bengio, Y. (2005). Semi-supervised learning by entropy minimization. In *Advances in neural information processing systems* (pp. 529–536).
200. Greco, S., Inuiguchi, M., & Slowinski, R. (2003). Fuzzy rough sets and multiple-premise gradual decision rules. In *Proceedings of the International Workshop on Fuzzy Logic and Applications* (pp. 148–163). Springer.
201. Hady, M., & Schwenker, F. (2008). Co-training by committee: A new semi-supervised learning framework. In *Proceedings of the IEEE International Conference on Data Mining Workshops* (pp. 563–572). IEEE.
202. Hady, M., & Schwenker, F. (2010). Combining committee-based semi-supervised learning and active learning. *Journal of Computer Science and Technology*, *25*(4), 681–698.
203. Haixiang, G., Yijing, L., Yanan, L., Xiao, L., & Jinling, L. (2016). BPSO-Adaboost-KNN ensemble learning algorithm for multi-class imbalanced data classification. *Engineering Applications of Artificial Intelligence*, *49*, 176–193.
204. Han, H., Wang, W., & Mao, B. (2005). Borderline-SMOTE: A new over-sampling method in imbalanced data sets learning. In *Proceedings of the International Conference on Intelligent Computing* (pp. 878–887). Springer.
205. Hand, D., & Till, R. (2001). A simple generalisation of the area under the ROC curve for multiple class classification problems. *Machine Learning*, *45*(2), 171–186.
206. Hand, D., & Vinciotti, V. (2003). Choosing k for two-class nearest neighbour classifiers with unbalanced classes. *Pattern Recognition Letters*, *24*(9), 1555–1562.
207. Hassan, M., Ramamohanarao, K., Karmakar, C., Hossain, M., & Bailey, J. (2010). A novel scalable multi-class ROC for effective visualization and computation. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining* (pp. 107–120). Springer.
208. Hastie, T., & Tibshirani, R. (1998). Classification by pairwise coupling. In *Advances in neural information processing systems* (pp. 507–513).
209. Haykin, S. (2004). *Neural networks: A comprehensive foundation*. Prentice Hall International.
210. He, H., Bai, Y., Garcia, E., & Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *Proceedings of the IEEE International Joint Conference on Neural Networks* (pp. 1322–1328). IEEE.
211. He, H., & Garcia, E. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, *21*(9), 1263–1284.
212. He, J., Gu, H., & Liu, W. (2012). Imbalanced multi-modal multi-label learning for subcellular localization prediction of human proteins with both single and multiple sites. *PloS One*, *7*(6), e37155.
213. He, Q., & Wu, C. (2011). Membership evaluation and feature selection for fuzzy support vector machine based on fuzzy rough sets. *Soft Computing*, *15*(6), 1105–1114.
214. He, Q., Wu, C., Chen, D., & Zhao, S. (2011). Fuzzy rough set based attribute reduction for information systems with fuzzy decisions. *Knowledge-Based Systems*, *24*(5), 689–696.
215. Hernández-González, J., Inza, I., & Lozano, J. (2016). Weak supervision and other non-standard classification problems: A taxonomy. *Pattern Recognition Letters*, *69*, 49–55.
216. Herrera, F., Charte, F., Rivera, A., & del Jesus, M. (2016a). *Multilabel classification: Problem analysis, metrics and techniques*. Springer.
217. Herrera, F., Ventura, S., Bello, R., Cornelis, C., Zafra, A., Sánchez-Tarragó, D., & Vluymans, S. (2016b). *Multiple instance learning - foundations and algorithms*. Springer.
218. Ho, T., Basu, M., & Law, M. (2006). Measures of geometrical complexity in classification problems. *Data Complexity in Pattern Recognition*, 1–23.
219. Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 65–70.

220. Hong, R., Wang, M., Gao, Y., Tao, D., Li, X., & Wu, X. (2014). Image annotation by multiple-instance learning with discriminative feature mapping and selection. *IEEE Transactions on Cybernetics*, *44*(5), 669–680.
221. Hong, T., Liou, Y., & Wang, S. (2009). Fuzzy rough sets with hierarchical quantitative attributes. *Expert systems with Applications*, *36*(3), 6790–6799.
222. Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366.
223. Hosmer Jr., D., Lemeshow, S., & Sturdivant, R. (2013). *Applied logistic regression*. Wiley.
224. Hu, Q., An, S., & Yu, D. (2010). Soft fuzzy rough sets for robust feature evaluation and selection. *Information Sciences*, *180*(22), 4384–4400.
225. Hu, Q., An, S., Yu, X., & Yu, D. (2011). Robust fuzzy rough classifiers. *Fuzzy Sets and Systems*, *183*(1), 26–43.
226. Hu, Q., Xie, Z., & Yu, D. (2007). Hybrid attribute reduction based on a novel fuzzy-rough model and information granulation. *Pattern Recognition*, *40*(12), 3509–3521.
227. Hu, Q., Yu, D., & Xie, Z. (2006). Information-preserving hybrid data reduction based on fuzzy-rough techniques. *Pattern Recognition Letters*, *27*(5), 414–423.
228. Hu, Q., Yu, D., & Xie, Z. (2008). Neighborhood classifiers. *Expert Systems with Applications*, *34*(2), 866–876.
229. Hu, Q., Zhang, L., An, S., Zhang, D., & Yu, D. (2012). On robust fuzzy rough set models. *IEEE Transactions on Fuzzy Systems*, *20*(4), 636–651.
230. Hu, S., Liang, Y., Ma, L., & He, Y. (2009). MSMOTE: Improving classification performance when training data is imbalanced. In *Proceedings of the 2nd International Workshop on Computer Science and Engineering*, vol. 2 (pp. 13–17). IEEE.
231. Huang, K., & Li, Z. (2011). A multilabel text classification algorithm for labeling risk factors in SEC form 10-K. *ACM Transactions on Management Information Systems*, *2*(3), 18:1–18:19.
232. Huang, Y., Hung, C., & Jiau, H. (2006). Evaluation of neural networks and data mining methods on a credit assessment task for class imbalance problem. *Nonlinear Analysis: Real World Applications*, *7*(4), 720–747.
233. Hühn, J., & Hüllermeier, E. (2009). FR3: A fuzzy rule learner for inducing reliable classifiers. *IEEE Transactions on Fuzzy Systems*, *17*(1), 138–149.
234. Hüllermeier, E., & Brinker, K. (2008). Learning valued preference structures for solving classification problems. *Fuzzy Sets and Systems*, *159*(18), 2337–2352.
235. Hüllermeier, E., & Vanderlooy, S. (2010). Combining predictions in pairwise classification: An optimal adaptive voting strategy and its relation to weighted voting. *Pattern Recognition*, *43*(1), 128–142.
236. Huo, L., Tang, P., Zhang, Z., & Tuia, D. (2015). Semisupervised classification of remote sensing images with hierarchical spatial similarity. *IEEE Geoscience and Remote Sensing Letters*, *12*(1), 150–154.
237. Hussain, A., & Cambria, E. (2018). Semi-supervised learning for big social data analysis. *Neurocomputing*, *275*, 1662–1673.
238. Jaiswal, A., Manjunatha, A., Madhu, B., & Murthy, P. (2016). Predicting unlabeled traffic for intrusion detection using semi-supervised machine learning. In *Proceedings of the International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques* (pp. 218–222). IEEE.
239. Japkowicz, N. (2000). The class imbalance problem: Significance and strategies. In *Proceedings of the International Conference on Artificial Intelligence* (pp. 111–117).
240. Japkowicz, N., & Stephen, S. (2002). The class imbalance problem: A systematic study. *Intelligent Data Analysis*, *6*(5), 429–449.
241. Jensen, R., & Cornelis, C. (2010). Fuzzy-rough instance selection. In *Proceedings of the 2010 IEEE International Conference on Fuzzy Systems* (pp. 1–7). IEEE.
242. Jensen, R., & Cornelis, C. (2011). Fuzzy-rough nearest neighbour classification and prediction. *Theoretical Computer Science*, *412*(42), 5871–5884.
243. Jensen, R., Cornelis, C., & Shen, Q. (2009). Hybrid fuzzy-rough rule induction and feature selection. In *Proceedings of the 2009 IEEE International Conference on Fuzzy Systems* (pp. 1151–1156). IEEE.

244. Jensen, R., & Mac Parthaláin, N. (2014). Nearest neighbour-based fuzzy-rough feature selection. In *Proceedings of the International Conference on Rough Sets and Current Trends in Computing* (pp. 35–46). Springer.
245. Jensen, R., & Mac Parthaláin, N. (2015). Towards scalable fuzzy-rough feature selection. *Information Sciences*, *323*, 1–15.
246. Jensen, R., & Shen, Q. (2005). Fuzzy-rough feature significance for fuzzy decision trees. In *Proceedings of the 2005 UK Workshop on Computational Intelligence* (pp. 89–96). Citeseer.
247. Jensen, R., & Shen, Q. (2009). New approaches to fuzzy-rough feature selection. *IEEE Transactions on Fuzzy Systems*, *17*(4), 824–838.
248. Jiang, J., Tsai, S., & Lee, S. (2012). FSKNN: Multi-label text categorization based on fuzzy similarity and k nearest neighbors. *Expert Systems with Applications*, *39*(3), 2813–2821.
249. Jiang, M., Pan, Z., & Li, N. (2017). Multi-label text categorization using L21-norm minimization extreme learning machine. *Neurocomputing*, *261*, 4–10.
250. Jing, X., Wu, F., Li, Z., Hu, R., & Zhang, D. (2016). Multi-label dictionary learning for image annotation. *IEEE Transactions on Image Processing*, *25*(6), 2712–2725.
251. Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *Proceedings of the International Conference on Machine Learning*, vol. 99 (pp. 200–209).
252. Joachims, T. (2003). Transductive learning via spectral graph partitioning. In *Proceedings of the 20th International Conference on Machine Learning* (pp. 290–297).
253. Joshi, M., Kumar, V., & Agarwal, R. (2001). Evaluating boosting algorithms to classify rare classes: Comparison and improvements. In *Proceedings of the IEEE International Conference on Data Mining* (pp. 257–264). IEEE.
254. Jurafsky, D., & Martin, J. (2014). *Speech and language processing*, vol. 3. Pearson London.
255. Kang, F., Jin, R., & Sukthankar, R. (2006). Correlated label propagation with application to multi-label learning. In *Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2 (pp. 1719–1726). IEEE.
256. Kejriwal, L., Darbari, V., & Verma, O. (2017). Multi instance multi label classification of restaurant images. In *Proceedings of the 2017 IEEE 7th International Advance Computing Conference* (pp. 722–727). IEEE.
257. Keller, J., Gray, M., & Givens, J. (1985). A fuzzy k-nearest neighbor algorithm. *IEEE Transactions on Systems, Man and Cybernetics*, *15*(4), 580–585.
258. Keogh, E., & Mueen, A. (2011). Curse of dimensionality. In *Encyclopedia of machine learning* (pp. 257–258). Springer.
259. Khoshgoftaar, T., Seiffert, C., Van Hulse, J., Napolitano, A., & Folleco, A. (2007). Learning with limited minority class data. In *Proceedings of the 6th International Conference on Machine Learning and Applications* (pp. 348–353). IEEE.
260. Kim, J. (2009). Estimating classification error rate: Repeated cross-validation, repeated hold-out and bootstrap. *Computational Statistics & Data Analysis*, *53*(11), 3735–3745.
261. Kingma, D., Mohamed, S., Rezende, D., & Welling, M. (2014). Semi-supervised learning with deep generative models. In *Advances in neural information processing systems* (pp. 3581–3589).
262. Klir, G., & Yuan, B. (1995). *Fuzzy sets and fuzzy logic* (Vol. 4). New Jersey: Prentice Hall.
263. Ko, A., Sabourin, R., & Britto Jr., A. (2008). From dynamic classifier selection to dynamic ensemble selection. *Pattern Recognition*, *41*(5), 1718–1731.
264. Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 1995 International Joint Conference on Artificial Intelligence* (pp. 1137–1143).
265. Krawczyk, B., Woźniak, M., & Schaefer, G. (2014). Cost-sensitive decision tree ensembles for effective imbalanced classification. *Applied Soft Computing*, *14*, 554–562.
266. Krijthe, J., & Loog, M. (2016). The peaking phenomenon in semi-supervised learning. In *Proceedings of the Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition and Structural and Syntactic Pattern Recognition* (pp. 299–309). Springer.

267. Kubat, M., & Matwin, S. (1997). Addressing the curse of imbalanced training sets: One-sided selection. In *Proceedings of the International Conference on Machine Learning*, vol. 97 (pp. 179–186).
268. Kumar, A., & Raj, B. (2016). Audio event detection using weakly labeled data. In *Proceedings of the 2016 ACM on Multimedia Conference* (pp. 1038–1047). ACM.
269. Kumar, S., Gao, X., & Welch, I. (2017). Cluster-than-label: Semi-supervised approach for domain adaptation. In *Proceedings of the IEEE 31st International Conference on Advanced Information Networking and Applications* (pp. 704–711). IEEE.
270. Kuncheva, L. (2004). *Combining pattern classifiers: Methods and algorithms*. Wiley.
271. Kuncheva, L., Bezdek, J., & Duin, R. (2001). Decision templates for multiple classifier fusion: An experimental comparison. *Pattern Recognition*, *34*(2), 299–314.
272. Kuncheva, L., & Whitaker, C. (2003). Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning*, *51*(2), 181–207.
273. Lamata, M., & Pérez, E. (2012). Obtaining OWA operators starting from a linear order and preference quantifiers. *International Journal of Intelligent Systems*, *27*(3), 242–258.
274. LaPierre, N., Rahman, M., & Rangwala, H. (2016). CAMIL: Clustering and assembly with multiple instance learning for phenotype prediction. In *Proceedings of the 2016 IEEE International Conference on Bioinformatics and Biomedicine* (pp. 33–40). IEEE.
275. Laurikkala, J. (2001). Improving identification of difficult small classes by balancing class distribution. In *Proceedings of the Conference on Artificial Intelligence in Medicine in Europe* (pp. 63–66). Springer.
276. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444.
277. Leskovec, J., Rajaraman, A., & Ullman, J. (2014). *Mining of massive datasets*. Cambridge University Press.
278. Lewis, D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. In *Proceedings of the European Conference on Machine Learning* (pp. 4–15). Springer.
279. Li, C., & Shi, G. (2013). Improvement of learning algorithm for the multi-instance multi-label RBF neural networks trained with imbalanced samples. *Journal of Information Science and Engineering*, *29*(4), 765–776.
280. Li, F., Min, F., & Liu, Q. (2008). Intra-cluster similarity index based on fuzzy rough sets for fuzzy c-means algorithm. In *Proceedings of the 3rd International Conference on Rough Sets and Knowledge Technology* (pp. 316–323). Springer.
281. Li, W., & Vasconcelos, N. (2015). Multiple instance learning for soft bags via top instances. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 4277–4285). IEEE.
282. Li, Y., Ji, S., Kumar, S., Ye, J., & Zhou, Z. (2012). Drosophila gene expression pattern annotation through multi-instance multi-label learning. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, *9*(1), 98–112.
283. Li, Y., Kwok, J., & Zhou, Z. (2016). Towards safe semi-supervised learning for multivariate performance measures. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, vol. 2016 (pp. 1816–1822).
284. Li, Y., Tax, D., Duin, R., & Loog, M. (2013). Multiple-instance learning as a classifier combining problem. *Pattern Recognition*, *46*(3), 865–874.
285. Li, Y., & Zhou, Z. (2015). Towards making unlabeled data never hurt. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *37*(1), 175–188.
286. Liang, S., & Srikant, R. (2017). Why deep neural networks for function approximation? In *Proceedings of the International Conference on Learning Representations*.
287. Lichman, M. (2013). *UCI machine learning repository*. urlhttp://archive.ics.uci.edu/ml.
288. Lima, A., & De Castro, L. (2014). A multi-label, semi-supervised classification approach applied to personality prediction in social media. *Neural Networks*, *58*, 122–130.
289. Lin, S., Chang, C., & Hsu, M. (2013). Multiple extreme learning machines for a two-class imbalance corporate life cycle prediction. *Knowledge-Based Systems*, *39*, 214–223.

290. Lin, W., Tsai, C., Hu, Y., & Jhang, J. (2017). Clustering-based undersampling in class-imbalanced data. *Information Sciences*, *409*, 17–26.
291. Lin, W., & Xu, D. (2016). Imbalanced multi-label learning for identifying antimicrobial peptides and their functional types. *Bioinformatics*, *32*(24), 3745–3752.
292. Lin, X., & Chen, X. (2010). Mr. KNN: Soft relevance for multi-label classification. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management* (pp. 349–358). ACM.
293. Liu, B., Hao, Z., & Tsang, E. (2008). Nesting one-against-one algorithm based on SVMs for pattern classification. *IEEE Transactions on Neural Networks*, *19*(12), 2044–2052.
294. Liu, B., Hao, Z., & Yang, X. (2007). Nesting algorithm for multi-classification problems. *Soft Computing-A Fusion of Foundations, Methodologies and Applications*, *11*(4), 383–389.
295. Liu, H., & Motoda, H. (1998). *Feature extraction, construction and selection: A data mining perspective*, vol. 453. Springer Science & Business Media.
296. Liu, H., & Motoda, H. (2002). On issues of instance selection. *Data Mining and Knowledge Discovery*, *6*(2), 115–130.
297. Liu, H., & Motoda, H. (2007). *Computational methods of feature selection*. CRC Press.
298. Liu, H., Wu, X., & Zhang, S. (2016). Neighbor selection for multilabel classification. *Neurocomputing*, *182*, 187–196.
299. Liu, X., Wu, J., & Zhou, Z. (2009). Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *39*(2), 539–550.
300. Liu, Y., & Kirchhoff, K. (2013). Graph-based semi-supervised learning for phone and segment classification. *Proceedings of Interspeech*, *2013*, 1840–1843.
301. Liu, Y., Zhou, Q., Rakus-Andersson, E., & Bai, G. (2012). A fuzzy-rough sets based compact rule induction method for classifying hybrid data. In *Proceedings of the 7th International Conference on Rough Sets and Knowledge Technology* (pp. 63–70). Springer.
302. Liu, Z., Tang, D., Cai, Y., Wang, R., & Chen, F. (2017). A hybrid method based on ensemble WELM for handling multi class imbalance in cancer microarray data. *Neurocomputing*, *266*, 641–650.
303. Lomax, S., & Vadera, S. (2013). A survey of cost-sensitive decision tree induction algorithms. *ACM Computing Surveys*, *45*(2), 16.
304. Loog, M. (2016). Contrastive pessimistic likelihood estimation for semi-supervised classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *38*(3), 462–475.
305. López, V., Fernández, A., del Jesus, M., & Herrera, F. (2013a). A hierarchical genetic fuzzy system based on genetic programming for addressing classification with highly imbalanced and borderline data-sets. *Knowledge-Based Systems*, *38*, 85–104.
306. López, V., Fernández, A., García, S., Palade, V., & Herrera, F. (2013b). An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences*, *250*, 113–141.
307. López, V., Fernández, A., & Herrera, F. (2014). On the importance of the validation technique for classification with imbalanced datasets: Addressing covariate shift when data is skewed. *Information Sciences*, *257*, 1–13.
308. López, V., Fernández, A., Moreno-Torres, J., & Herrera, F. (2012). Analysis of preprocessing versus cost-sensitive learning for imbalanced classification. Open problems on intrinsic data characteristics. *Expert Systems with Applications*, *39*(7), 6585–6608.
309. Luengo, J., Fernández, A., García, S., & Herrera, F. (2011). Addressing data complexity for imbalanced data sets: Analysis of SMOTE-based oversampling and evolutionary undersampling. *Soft Computing*, *15*(10), 1909–1936.
310. Luo, Y., Tao, D., Geng, B., Xu, C., & Maybank, S. (2013). Manifold regularized multitask learning for semi-supervised multilabel image classification. *IEEE Transactions on Image Processing*, *22*(2), 523–536.
311. Mac Parthaláin, N., & Jensen, R. (2011). Fuzzy-rough set based semi-supervised learning. In *Proceedings of the 2011 IEEE International Conference on Fuzzy Systems* (pp. 2465–2472). IEEE.

312. Mac Parthaláin, N., & Jensen, R. (2013). Simultaneous feature and instance selection using fuzzy-rough bireducts. In *Proceedings of the 2013 IEEE International Conference on Fuzzy Systems* (pp. 1–8). IEEE.
313. Maciejewski, T., & Stefanowski, J. (2011). Local neighbourhood extension of SMOTE for mining imbalanced data. In *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining* (pp. 104–111). IEEE.
314. Mahalanobis, P. (1936). On the generalised distance in statistics. *Proceedings of the National Institute of Sciences of India* (pp. 49–55).
315. Mallows, C. (1957). Non-null ranking models. *Biometrika*, *44*(1/2), 114–130.
316. Manning, C., & Schütze, H. (1999). *Foundations of statistical natural language processing*. MIT Press.
317. Maron, O., & Lozano-Pérez, T. (1998). A framework for multiple-instance learning. In *Advances in neural information processing systems* (pp. 570–576).
318. Maron, O., & Ratan, A. (1998). Multiple-instance learning for natural scene classification. In *Proceedings of the 15th International Conference on Machine Learning*, vol. 98 (pp. 341–349).
319. Mason, S., & Graham, N. (2002). Areas beneath the relative operating characteristics (ROC) and relative operating levels (ROL) curves: Statistical significance and interpretation. *Quarterly Journal of the Royal Meteorological Society*, *128*(584), 2145–2166.
320. Mendialdua, I., Martínez-Otzeta, J., Rodriguez-Rodriguez, I., Ruiz-Vazquez, T., & Sierra, B. (2015). Dynamic selection of the best base classifier in one versus one. *Knowledge-Based Systems*, *85*, 298–306.
321. Mera, C., Arrieta, J., Orozco-Alzate, M., & Branch, J. (2015). A bag oversampling approach for class imbalance in multiple instance learning. In *Proceedings of the Iberoamerican Congress on Pattern Recognition* (pp. 724–731). Springer.
322. Mera, C., Orozco-Alzate, M., & Branch, J. (2014). Improving representation of the positive class in imbalanced multiple-instance learning. In *Proceedings of the International Conference Image Analysis and Recognition* (pp. 266–273). Springer.
323. Michalski, R., Carbonell, J., & Mitchell, T. (2013). *Machine learning: An artificial intelligence approach*. Springer Science & Business Media.
324. Mieszkowicz-Rolka, A., & Rolka, L. (2004). Variable precision fuzzy rough sets. In *Transactions on Rough Sets I* (pp. 144–160). Springer.
325. Mieszkowicz-Rolka, A., & Rolka, L. (2008). Fuzzy rough approximations of process data. *International Journal of Approximate Reasoning*, *49*(2), 301–315.
326. Minhas, F., & Ben-Hur, A. (2012). Multiple instance learning of calmodulin binding sites. *Bioinformatics*, *28*(18), i416–i422.
327. Minsky, M., & Papert, S. (1969). *Perceptrons: An introduction to computational geometry*. MIT Press.
328. Mitchell, T. (1997). *Machine learning*. McGraw-Hill.
329. Moreno-Torres, J., Raeder, T., Alaiz-RodríGuez, R., Chawla, N., & Herrera, F. (2012a). A unifying view on dataset shift in classification. *Pattern Recognition*, *45*(1), 521–530.
330. Moreno-Torres, J., Sáez, J., & Herrera, F. (2012b). Study on the impact of partition-induced dataset shift on k-fold cross-validation. *IEEE Transactions on Neural Networks and Learning Systems*, *23*(8), 1304–1312.
331. Moutafis, P., Leng, M., & Kakadiaris, I. (2017). An overview and empirical comparison of distance metric learning methods. *IEEE Transactions on Cybernetics*, *47*(3), 612–625.
332. Naganjaneyulu, S., & Kuppa, M. (2013). A novel framework for class imbalance learning using intelligent under-sampling. *Progress in Artificial Intelligence*, *2*(1), 73–84.
333. Nanculef, R., Flaounas, I., & Cristianini, N. (2014). Efficient classification of multi-labeled text streams by clashing. *Expert Systems with Applications*, *41*(11), 5431–5450.
334. Napierała, K., Stefanowski, J., & Wilk, S. (2010). Learning from imbalanced data in presence of noisy and borderline examples. In *Rough sets and current trends in computing* (pp. 158–167). Springer.

335. Ng, A., & Jordan, M. (2002). On discriminative versus generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in neural information processing systems* (pp. 841–848).
336. Noorbehbahani, F., Fanian, A., Mousavi, R., & Hasannejad, H. (2017). An incremental intrusion detection system using a new semi-supervised stream classification method. *International Journal of Communication Systems*, *30*(4).
337. Novikoff, A. (1962). On convergence proofs for perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata* (pp. 615–622). Polytechnic Institute of Brooklyn.
338. O'Hagan, M. (1988). Aggregating template or rule antecedents in real-time expert systems with fuzzy set logic. In *Proceedings of the 22nd Asilomar Conference on Signals, Systems and Computers*, vol. 2 (pp. 681–689). IEEE.
339. Oreski, D., Oreski, S., & Klicek, B. (2017). Effects of dataset characteristics on the performance of feature selection techniques. *Applied Soft Computing*, *52*, 109–119.
340. Ortigosa-Hernández, J., Inza, I., & Lozano, J. (2017). Measuring the class-imbalance extent of multi-class problems. *Pattern Recognition Letters*, *98*, 32–38.
341. Pan, S., & Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, *22*(10), 1345–1359.
342. Pawlak, Z. (1982). Rough sets. *International Journal of Parallel Programming*, *11*(5), 341–356.
343. Peng, L., Yang, B., Chen, Y., & Abraham, A. (2009). Data gravitation based classification. *Information Sciences*, *179*(6), 809–819.
344. Pérez-Ortiz, M., Gutiérrez, P., Ayllón-Terán, M., Heaton, N., Ciria, R., Briceño, J., et al. (2017). Synthetic semi-supervised learning in imbalanced domains: Constructing a model for donor-recipient matching in liver transplantation. *Knowledge-Based Systems*, *123*, 75–87.
345. Pio, G., Malerba, D., D'Elia, D., & Ceci, M. (2014). Integrating microRNA target predictions for the discovery of gene regulatory networks: A semi-supervised ensemble learning approach. *BMC Bioinformatics*, *15*(1), S4.
346. Platt, J. (1999). Using analytic QP and sparseness to speed training of support vector machines. In *Advances in neural information processing systems* (pp. 557–563).
347. Platt, J., Cristianini, N., & Shawe-Taylor, J. (2000). Large margin DAGs for multiclass classification. In *Advances in neural information processing systems* (pp. 547–553).
348. Popescu, M., & Mahnot, A. (2012). Early illness recognition using in-home monitoring sensors and multiple instance learning. *Methods of Information in Medicine*, *51*(4), 359–367.
349. Prasad, P., & Rao, C. (2011). Extensions to iQuickReduct. In *Proceedings of the International Workshop on Multi-Disciplinary Trends in Artificial Intelligence* (pp. 351–362). Springer.
350. Price, K., Storn, R., & Lampinen, J. (2006). *Differential evolution: A practical approach to global optimization*. Springer Science & Business Media.
351. Qi, X., & Han, Y. (2007). Incorporating multiple SVMs for automatic image annotation. *Pattern Recognition*, *40*(2), 728–741.
352. Qian, Y., Wang, Q., Cheng, H., Liang, J., & Dang, C. (2015). Fuzzy-rough feature selection accelerator. *Fuzzy Sets and Systems*, *258*, 61–78.
353. Qu, Y., Shang, C., Shen, Q., Mac Parthaláin, N., & Wu, W. (2011). Kernel-based fuzzy-rough nearest neighbour classification. In *Proceedings of the 2011 IEEE International Conference on Fuzzy Systems* (pp. 1523–1529). IEEE.
354. Quinlan, J. (1986). Induction of decision trees. *Machine Learning*, *1*(1), 81–106.
355. Quinlan, J. (1987). Simplifying decision trees. *International Journal of Man-Machine Studies*, *27*(3), 221–234.
356. Quinlan, J. (1993). *C4.5: Programs for machine learning*. Morgan Kaufmann.
357. Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., & Lawrence, N. (2009). *Dataset shift in machine learning*. The MIT Press.
358. Radzikowska, A., & Kerre, E. (2002). A comparative study of fuzzy rough sets. *Fuzzy Sets and Systems*, *126*(2), 137–155.

359. Ramentol, E., Caballero, Y., Bello, R., & Herrera, F. (2012a). SMOTE-RSB*: A hybrid preprocessing approach based on oversampling and undersampling for high imbalanced data-sets using SMOTE and rough sets theory. *Knowledge and Information Systems*, *33*(2), 245–265.
360. Ramentol, E., Verbiest, N., Bello, R., Caballero, Y., Cornelis, C., & Herrera, F. (2012b). SMOTE-FRST: A new resampling method using fuzzy rough set theory. In *Proceedings of the 10th International FLINS Conference on Uncertainty Modelling in Knowledge Engineering and Decision Making* (pp. 800–805). World Scientific.
361. Ramentol, E., Vluymans, S., Verbiest, N., Caballero, Y., Bello, R., Cornelis, C., et al. (2015). IFROWANN: Imbalanced fuzzy-rough ordered weighted average nearest neighbor classification. *IEEE Transactions on Fuzzy Systems*, *23*(5), 1622–1637.
362. Read, J., Pfahringer, B., & Holmes, G. (2008). Multi-label classification using ensembles of pruned sets. In *Proceedings of the 8th IEEE International Conference on Data Mining* (pp. 995–1000). IEEE.
363. Read, J., Pfahringer, B., Holmes, G., & Frank, E. (2011). Classifier chains for multi-label classification. *Machine Learning*, *85*(3), 333–359.
364. Reyes, O., Morell, C., & Ventura, S. (2016). Effective lazy learning algorithm based on a data gravitation model for multi-label learning. *Information Sciences*, *340*, 159–174.
365. Rokach, L. (2016). Decision forest: Twenty years of research. *Information Fusion*, *27*, 111–125.
366. Rokach, L., & Maimon, O. (2014). *Data mining with decision trees: Theory and applications*. World Scientific.
367. Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386.
368. Rumelhart, D., Hinton, G., & Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536.
369. Sánchez-Tarragó, D., Cornelis, C., Bello, R., & Herrera, F. (2014). A multi-instance learning wrapper based on the Rocchio classifier for web index recommendation. *Knowledge-Based Systems*, *59*, 173–181.
370. Sarkar, M. (2007). Fuzzy-rough nearest neighbor algorithms in classification. *Fuzzy Sets and Systems*, *158*(19), 2134–2152.
371. Sarkar, M., & Yegnanarayana, B. (1998a). Application of fuzzy-rough sets in modular neural networks. In *Proceedings of 1998 IEEE International Joint Conference on Neural Networks* (pp. 741–746). IEEE.
372. Sarkar, M., & Yegnanarayana, B. (1998b). Fuzzy-rough neural networks for vowel classification. In *Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics*, vol. 5 (pp. 4160–4165). IEEE.
373. Schapire, R. (1990). The strength of weak learnability. *Machine Learning*, *5*(2), 197–227.
374. Schapire, R. (2003). The boosting approach to machine learning: An overview. In *Nonlinear Estimation and Classification* (pp. 149–171). Springer.
375. Schapire, R., & Singer, Y. (1999). Improved boosting algorithms using confidence-rated predictions. *Machine Learning*, *37*(3), 297–336.
376. Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85–117.
377. Schölkopf, B., & Smola, A. (2002). *Learning with kernels: Support vector machines, regularization, optimization, and beyond*. MIT Press.
378. Seiffert, C., Khoshgoftaar, T., Van Hulse, J., & Napolitano, A. (2010). RUSBoost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *40*(1), 185–197.
379. Shannon, C. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, *27*(3), 379–423.
380. Shao, Y., Chen, W., Zhang, J., Wang, Z., & Deng, N. (2014). An efficient weighted lagrangian twin support vector machine for imbalanced data classification. *Pattern Recognition*, *47*(9), 3158–3167.

381. Shenfield, A., & Rostami, S. (2017). Multi-objective evolution of artificial neural networks in multi-class medical diagnosis problems with class imbalance. In *Proceedings of the 2017 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology* (pp. 1–8). IEEE.
382. Shrivastava, H., Huddar, V., Bhattacharya, S., & Rajan, V. (2015). Classification with imbalance: A similarity-based method for predicting respiratory failure. In *Proceedings of the 2015 IEEE International Conference on Bioinformatics and Biomedicine* (pp. 707–714). IEEE.
383. Singh, A., Nowak, R., & Zhu, X. (2009). Unlabeled data: Now it helps, now it doesn't. In *Advances in neural information processing systems* (pp. 1513–1520).
384. Søgaard, A. (2013). Semi-supervised learning and domain adaptation in natural language processing. *Synthesis Lectures on Human Language Technologies*, *6*(2), 1–103.
385. Spyromitros, E., Tsoumakas, G., & Vlahavas, I. (2008). An empirical study of lazy multilabel classification algorithms. In *Proceedings of the Hellenic conference on Artificial Intelligence* (pp. 401–406). Springer.
386. Stefanowski, J., & Wilk, S. (2008). Selective pre-processing of imbalanced data for improving classification performance. Lecture Notes in Computer Science, vol. 5182 (pp. 283–292).
387. Subramanya, A., & Talukdar, P. (2014). Graph-based semi-supervised learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, *8*(4), 1–125.
388. Sun, L., Lu, Y., Yang, K., & Li, S. (2012). ECG analysis using multiple instance learning for myocardial infarction detection. *IEEE Transactions on Biomedical Engineering*, *59*(12), 3348–3356.
389. Sun, S. (2013). A survey of multi-view machine learning. *Neural Computing and Applications*, *23*(7–8), 2031–2038.
390. Sun, Y., Kamel, M., Wong, A., & Wang, Y. (2007). Cost-sensitive boosting for classification of imbalanced data. *Pattern Recognition*, *40*(12), 3358–3378.
391. Sun, Y., Wong, A., & Kamel, M. (2009). Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence*, *23*(04), 687–719.
392. Tahir, M., Kittler, J., & Bouridane, A. (2012a). Multilabel classification using heterogeneous ensemble of multi-label classifiers. *Pattern Recognition Letters*, *33*(5), 513–523.
393. Tahir, M., Kittler, J., & Yan, F. (2012b). Inverse random under sampling for class imbalance problem and its application to multi-label classification. *Pattern Recognition*, *45*(10), 3738–3750.
394. Tang, B., & He, H. (2017). GIR-based ensemble sampling approaches for imbalanced learning. *Pattern Recognition*, *71*, 306–319.
395. Tang, X., & Han, M. (2010). Semi-supervised Bayesian artmap. *Applied Intelligence*, *33*(3), 302–317.
396. Tepvorachai, G., & Papachristou, C. (2008). Multi-label imbalanced data enrichment process in neural net classifier training. In *Proceedings of the IEEE International Joint Conference on Neural Networks* (pp. 1301–1307). IEEE.
397. Tian, X., Gasso, G., & Canu, S. (2012). A multiple kernel framework for inductive semi-supervised SVM learning. *Neurocomputing*, *90*, 46–58.
398. Tichỳ, L., Chytrỳ, M., & Botta-Dukát, Z. (2014). Semi-supervised classification of vegetation: Preserving the good old units and searching for new ones. *Journal of Vegetation Science*, *25*(6), 1504–1512.
399. Ting, K. (2000). A comparative study of cost-sensitive boosting algorithms. In *Proceedings of the 17th International Conference on Machine Learning* (pp. 983–990).
400. Ting, K. (2002). An instance-weighting method to induce cost-sensitive trees. *IEEE Transactions on Knowledge and Data Engineering*, *14*(3), 659–665.
401. Tomás, J., Spolaôr, N., Cherman, E., & Monard, M. (2014). A framework to generate synthetic multi-label datasets. *Electronic Notes in Theoretical Computer Science*, *302*, 155–176.
402. Torra, V. (1999). On the learning of weights in some aggregation operators: The weighted mean and OWA operators. *Mathware and Soft Computing*, *6*(2/3), 249–265.
403. Triguero, I., del Río, S., López, V., Bacardit, J., Benítez, J., & Herrera, F. (2015a). ROSEFW-RF: The winner algorithm for the ECBDL14 big data competition: An extremely imbalanced big data bioinformatics problem. *Knowledge-Based Systems*, *87*, 69–79.

404. Triguero, I., Derrac, J., García, S., & Herrera, F. (2012). A taxonomy and experimental study on prototype generation for nearest neighbor classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *42*(1), 86–100.
405. Triguero, I., García, S., & Herrera, F. (2015b). SEG-SSC: A framework based on synthetic examples generation for self-labeled semi-supervised classification. *IEEE Transactions on Cybernetics*, *45*(4), 622–634.
406. Triguero, I., García, S., & Herrera, F. (2015c). Self-labeled techniques for semi-supervised learning: Taxonomy, software and empirical study. *Knowledge and Information Systems*, *42*(2), 245–284.
407. Triguero, I., Peralta, D., Bacardit, J., García, S., & Herrera, F. (2015d). MRPR: A MapReduce solution for prototype reduction in big data classification. *Neurocomputing*, *150*, 331–345.
408. Triguero, I., & Vens, C. (2016). Labelling strategies for hierarchical multi-label classification techniques. *Pattern Recognition*, *56*, 170–183.
409. Tsang, E., Chen, D., Yeung, D., Wang, X., & Lee, J. (2008). Attributes reduction using fuzzy rough sets. *IEEE Transactions on Fuzzy systems*, *16*(5), 1130–1141.
410. Tsoumakas, G., & Katakis, I. (2007). Multi-label classification: An overview. *International Journal of Data Warehousing and Mining*, *3*(3), 1–13.
411. Turner, M., Chakrabarti, C., Jones, T., Xu, J., Fox, P., Luger, G., et al. (2013). Automated annotation of functional imaging experiments via multi-label classification. *Frontiers in Neuroscience*, *7*, 1–13.
412. Van Le, V., Van Tran, L., & Van Tran, H. (2016). A novel semi-supervised algorithm for the taxonomic assignment of metagenomic reads. *BMC Bioinformatics*, *17*(1), 22.
413. Verbiest, N. (2014). *Fuzzy rough and evolutionary approaches to instance selection*. Ph.D. thesis, Ghent University, Belgium.
414. Verbiest, N., Cornelis, C., & Herrera, F. (2013a). FRPS: A fuzzy rough prototype selection method. *Pattern Recognition*, *46*(10), 2770–2782.
415. Verbiest, N., Cornelis, C., & Herrera, F. (2013b). 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* (pp. 180–190). Springer.
416. Verbiest, N., Cornelis, C., & Jensen, R. (2012a). Fuzzy rough positive region based nearest neighbour classification. In *Proceedings of the 2012 IEEE International Conference on Fuzzy Systems* (pp. 1–7). IEEE.
417. Verbiest, N., Ramentol, E., Cornelis, C., & Herrera, F. (2012b). Improving SMOTE with fuzzy rough prototype selection to detect noise in imbalanced classification data. In *Proceedings of the 13th Ibero-American Conference on Artificial Intelligence* (pp. 169–178). Springer.
418. Veropoulos, K., Campbell, C., & Cristianini, N. (1999). Controlling the sensitivity of support vector machines. In *Proceedings of the International Joint Conference on AI* (pp. 55–60).
419. Villar, P., Fernández, A., Carrasco, R., & Herrera, F. (2012). Feature selection and granularity learning in genetic fuzzy rule-based classification systems for highly imbalanced data-sets. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *20*(03), 369–397.
420. Vluymans, S. (2014). *Instance selection for imbalanced data*. Master's thesis, Ghent University, Belgium.
421. Vluymans, S., Cornelis, C., Herrera, F., & Saeys, Y. (2018a). Multi-label classification using a fuzzy rough neighborhood consensus. *Information Sciences*, *433–434*, 96–114.
422. Vluymans, S., D'eer, L., Saeys, Y., & Cornelis, C. (2015). Applications of fuzzy rough set theory in machine learning: A survey. *Fundamenta Informaticae*, *142*(1–4), 53–86.
423. Vluymans, S., Fernández, A., Saeys, Y., Cornelis, C., & Herrera, F. (2018b). Dynamic affinity-based classification of multi-class imbalanced data with one-vs-one decomposition: A fuzzy rough approach. *Knowledge and Information Systems*, *56*(1), 55–84.
424. Vluymans, S., Mac Parthaláin, N., Cornelis, C., & Saeys, Y. (2016a). Fuzzy rough sets for self-labelling: An exploratory analysis. In *Proceedings of the 2016 IEEE International Conference on Fuzzy Systems* (pp. 931–938). IEEE.

425. Vluymans, S., Sánchez-Tarragó, D., Saeys, Y., Cornelis, C., & Herrera, F. (2016b). Fuzzy multi-instance classifiers. *IEEE Transactions on Fuzzy Systems*, *24*(6), 1395–1409.
426. Vluymans, S., Sánchez-Tarragó, D., Saeys, Y., Cornelis, C., & Herrera, F. (2016c). Fuzzy rough classifiers for class imbalanced multi-instance data. *Pattern Recognition*, *53*, 36–45.
427. Wan, J., Yang, M., Gao, Y., & Chen, Y. (2014). Pairwise costs in semisupervised discriminant analysis for face recognition. *IEEE Transactions on Information Forensics and Security*, *9*(10), 1569–1580.
428. Wan, S., Duan, Y., & Zou, Q. (2017). HPSLPred: An ensemble multi-label classifier for human protein subcellular location prediction with imbalanced source. *Proteomics*, *17*(17–18).
429. Wang, C., Qi, Y., Shao, M., Hu, Q., Chen, D., Qian, Y., et al. (2017a). A fitting model for feature selection with fuzzy rough sets. *IEEE Transactions on Fuzzy Systems*, *25*(4), 741–753.
430. Wang, C., Xu, Z., Wang, S., & Zhang, H. (2017b). Semi-supervised classification framework of hyperspectral images based on the fusion evidence entropy. *Multimedia Tools and Applications*, 1–19.
431. Wang, H., Ding, C., & Huang, H. (2010a). Multi-label classification: Inconsistency and class balanced k-nearest neighbor. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence* (pp. 1264–1266).
432. Wang, J., Jebara, T., & Chang, S.-F. (2013a). Semi-supervised learning using greedy max-cut. *Journal of Machine Learning Research*, *14*, 771–800.
433. Wang, J., & Zucker, J. (2000). Solving multiple-instance problem: A lazy learning approach. In *Proceedings of the 17th International Conference on Machine Learning* (pp. 1119–1125).
434. Wang, Q., Yuan, Y., Yan, P., & Li, X. (2013b). Saliency detection by multiple-instance learning. *IEEE Transactions on Cybernetics*, *43*(2), 660–672.
435. Wang, S., Chen, H., & Yao, X. (2010b). Negative correlation learning for classification ensembles. In *Proceedings of the 2010 International Joint Conference on Neural Networks* (pp. 1–8). IEEE.
436. Wang, S., McKenna, M., Nguyen, T., Burns, J., Petrick, N., Sahiner, B., et al. (2012). Seeing is believing: Video classification for computed tomographic colonography using multiple-instance learning. *IEEE Transactions on Medical Imaging*, *31*(5), 1141–1153.
437. Wang, S., & Yao, X. (2009). Diversity analysis on imbalanced data sets by using ensemble models. In *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining* (pp. 324–331). IEEE.
438. Wang, S., & Yao, X. (2012). Multiclass imbalance problems: Analysis and potential solutions. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *42*(4), 1119–1130.
439. Wang, X., Liu, X., Japkowicz, N., & Matwin, S. (2013c). Resampling and cost-sensitive methods for imbalanced multi-instance learning. In *Proceedings of the 2013 IEEE 13th International Conference on Data Mining Workshops* (pp. 808–816). IEEE.
440. Wang, X., Matwin, S., Japkowicz, N., & Liu, X. (2013d). Cost-sensitive boosting algorithms for imbalanced multi-instance datasets. In *Proceedings of the Canadian Conference on Artificial Intelligence* (pp. 174–186). Springer.
441. Wang, X., Zhang, W., Zhang, Q., & Li, G. (2015). MultiP-SChlo: Multi-label protein subchloroplast localization prediction with Chou's pseudo amino acid composition and a novel multi-label classifier. *Bioinformatics*, *31*(16), 2639–2645.
442. Wang, Y., Li, X., & Ding, X. (2016). Probabilistic framework of visual anomaly detection for unbalanced data. *Neurocomputing*, *201*, 12–18.
443. Wei, Y., Xia, W., Lin, M., Huang, J., Ni, B., Dong, J., et al. (2016). HCP: A flexible CNN framework for multi-label image classification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *38*(9), 1901–1907.
444. Weidmann, N., Frank, E., & Pfahringer, B. (2003). A two-level learning method for generalized multi-instance problems. In *Proceedings of the European Conference on Machine Learning* (pp. 468–479). Springer.
445. Weinberger, K., & Saul, L. (2009). Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, *10*, 207–244.

446. Weiss, K., Khoshgoftaar, T., & Wang, D. (2016). A survey of transfer learning. *Journal of Big Data*, *3*(9), 1–40.
447. Wilcoxon, F. (1945). Individual comparisons by ranking methods. *Biometrics Bulletin*, *1*(6), 80–83.
448. Wilson, D., & Martinez, T. (1997). Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research*, *6*, 1–34.
449. Wu, T., Lin, C., & Weng, R. (2004). Probability estimates for multi-class classification by pairwise coupling. *Journal of Machine Learning Research*, *5*, 975–1005.
450. Xia, X., Yang, X., Li, S., Wu, C., & Zhou, L. (2011). RW.KNN: A proposed random walk KNN algorithm for multi-label classification. In *Proceedings of the 4th Workshop on Workshop for Ph.D. Students In Information & Knowledge Management* (pp. 87–90). ACM.
451. Xie, B., Wang, M., & Tao, D. (2011). Toward the optimization of normalized graph laplacian. *IEEE Transactions on Neural Networks*, *22*(4), 660–666.
452. Xiong, S., Azimi, J., & Fern, X. (2014). Active learning of constraints for semi-supervised clustering. *IEEE Transactions on Knowledge and Data Engineering*, *26*(1), 43–54.
453. Xu, J. (2011). An empirical comparison of weighting functions for multi-label distance-weighted k-nearest neighbour method. In *Proceedings of the 1st International Conference on Artificial Intelligence, Soft Computing and Applications* (pp. 13–20).
454. Xu, X. (2003). *Statistical learning in multiple instance problems*. Ph.D. thesis, The University of Waikato, New Zealand.
455. Xu, X., & Frank, E. (2004). Logistic regression and boosting for labeled bags of instances. In *Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, vol. 4 (pp. 272–281). Springer.
456. Xu, Y., Min, H., Wu, Q., Song, H., & Ye, B. (2017). Multi-instance metric transfer learning for genome-wide protein function prediction. *Scientific Reports*, *7*, 1–15.
457. Xu, Y., Yang, F., & Shen, H. (2016). Incorporating organelle correlations into semi-supervised learning for protein subcellular localization prediction. *Bioinformatics*, *32*(14), 2184–2192.
458. Xu, Z. (2005). An overview of methods for determining OWA weights. *International Journal of Intelligent Systems*, *20*(8), 843–865.
459. Xu, Z. (2006). Dependent OWA operators. In *Proceedings of the International Conference on Modeling Decisions for Artificial Intelligence* (pp. 172–178). Springer.
460. Yager, R. (1988). On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Transactions on Systems, Man, and Cybernetics*, *18*(1), 183–190.
461. Yager, R., & Filev, D. (1999). Induced ordered weighted averaging operators. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, *29*(2), 141–150.
462. Yager, R., & Kacprzyk, J. (2012). *The ordered weighted averaging operators: Theory and applications*. Springer Science & Business Media.
463. Yang, X., Kuang, Q., Zhang, W., & Zhang, G. (2017, in press). AMDO: An over-sampling technique for multi-class imbalanced problems. *IEEE Transactions on Knowledge and Data Engineering*.
464. Yao, Y. (1998). Relational interpretations of neighborhood operators and rough set approximation operators. *Information Sciences*, *111*(1–4), 239–259.
465. Yao, Y., Mi, J., & Li, Z. (2014). A novel variable precision (θ, σ)-fuzzy rough set model based on fuzzy granules. *Fuzzy Sets and Systems*, *236*, 58–72.
466. Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting on Association for Computational Linguistics* (pp. 189–196). Association for Computational Linguistics.
467. Yen, S., & Lee, Y. (2006). Under-sampling approaches for improving prediction of the minority class in an imbalanced dataset. In *Intelligent control and automation* (pp. 731–740).
468. Yijing, L., Haixiang, G., Xiao, L., Yanan, L., & Jinling, L. (2016). Adapted ensemble classification algorithm based on multiple classifier system and feature selection for classifying multi-class imbalanced data. *Knowledge-Based Systems*, *94*, 88–104.
469. Yoon, K., & Kwek, S. (2005). An unsupervised learning approach to resolving the data imbalanced issue in supervised learning problems in functional genomics. In *Proceedings of the 5th International Conference on Hybrid Intelligent Systems* (pp. 303–308). IEEE.

470. Younes, Z., Abdallah, F., & Denœux, T. (2008). Multi-label classification algorithm derived from k-nearest neighbor rule with label dependencies. In *Proceedings of the 2008 16th European Signal Processing Conference* (pp. 1–5). IEEE.
471. Younes, Z., Abdallah, F., & Denœux, T. (2009). An evidence-theoretic k-nearest neighbor rule for multi-label classification. In *Proceedings of the International Conference on Scalable Uncertainty Management* (pp. 297–308). Springer.
472. Younes, Z., Abdallah, F., & Denœux, T. (2010a). Evidential multi-label classification approach to learning from data with imprecise labels. In *Proceedings of the International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems* (pp. 119–128). Springer.
473. Younes, Z., Abdallah, F., & Denœux, T. (2010b). Fuzzy multi-label learning under veristic variables. In *Proceedings of the 2010 IEEE International Conference on Fuzzy Systems* (pp. 1–8). IEEE.
474. Younes, Z., Abdallah, F., Denœux, T., & Snoussi, H. (2011). A dependent multilabel classification method derived from the k-nearest neighbor rule. *EURASIP Journal on Advances in Signal Processing*.
475. Yu, H., Hong, S., Yang, X., Ni, J., Dan, Y., & Qin, B. (2013). Recognition of multiple imbalanced cancer types based on DNA microarray data using ensemble classifiers. *BioMed Research International, 2013*.
476. Yu, Y., Pedrycz, W., & Miao, D. (2014). Multi-label classification by exploiting label correlations. *Expert Systems with Applications*, *41*(6), 2989–3004.
477. Zadeh, L. (1965). Fuzzy sets. *Information and Control*, *8*(3), 338–353.
478. Zafra, A., Gibaja, E., & Ventura, S. (2011). Multiple instance learning with multiple objective genetic programming for web mining. *Applied Soft Computing*, *11*(1), 93–102.
479. Zafra, A., Romero, C., Ventura, S., & Herrera-Viedma, E. (2009). Multi-instance genetic programming for web index recommendation. *Expert Systems with Applications*, *36*(9), 11470–11479.
480. Zarinabad, N., Wilson, M., Gill, S., Manias, K., Davies, N., & Peet, A. (2017). Multiclass imbalance learning: Improving classification of pediatric brain tumors from magnetic resonance spectroscopy. *Magnetic Resonance in Medicine*, *77*(6), 2114–2124.
481. Zemmal, N., Azizi, N., Dey, N., & Sellami, M. (2016). Adaptive semi supervised support vector machine semi supervised learning with features cooperation for breast cancer classification. *Journal of Medical Imaging and Health Informatics*, *6*(1), 53–62.
482. Zeng, X., & Martinez, T. (2000). Distribution-balanced stratified cross-validation for accuracy estimation. *Journal of Experimental & Theoretical Artificial Intelligence*, *12*(1), 1–12.
483. Zhai, J. (2011). Fuzzy decision tree based on fuzzy-rough technique. *Soft Computing*, *15*(6), 1087–1096.
484. Zhang, K., & Song, H. (2013). Real-time visual tracking via online weighted multiple instance learning. *Pattern Recognition*, *46*(1), 397–411.
485. Zhang, M., Li, Y., Liu, X., & Geng, X. (2017a). Binary relevance for multi-label learning: An overview. *Frontiers of Computer Science*, 1–12.
486. Zhang, M., & Zhou, Z. (2007). ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognition*, *40*(7), 2038–2048.
487. Zhang, M., & Zhou, Z. (2009). Multi-instance clustering with applications to multi-instance prediction. *Applied Intelligence*, *31*(1), 47–68.
488. Zhang, M., & Zhou, Z. (2014). A review on multi-label learning algorithms. *IEEE Transactions on Knowledge and Data Engineering*, *26*(8), 1819–1837.
489. Zhang, S. (2011). Shell-neighbor method and its application in missing data imputation. *Applied Intelligence*, *35*(1), 123–133.
490. Zhang, T., & Oles, F. (2000). The value of unlabeled data for classification problems. In *Proceedings of the 17th International Conference on Machine Learning* (pp. 1191–1198). Citeseer.
491. Zhang, X., Mei, C., Chen, D., & Li, J. (2016a). Feature selection in mixed data: A method using a novel fuzzy rough set-based information entropy. *Pattern Recognition*, *56*, 1–15.

492. Zhang, Y., & Yang, Y. (2015). Cross-validation for selecting a model selection procedure. *Journal of Econometrics*, *187*(1), 95–112.
493. Zhang, Z., Krawczyk, B., García, S., Rosales-Pérez, A., & Herrera, F. (2016b). Empowering one-vs-one decomposition with ensemble learning for multi-class imbalanced data. *Knowledge-Based Systems*, *106*, 251–263.
494. Zhang, Z., Luo, X., García, S., Tang, J., & Herrera, F. (2017b). Exploring the effectiveness of dynamic ensemble selection in the one-versus-one scheme. *Knowledge-Based Systems*, *125*, 53–63.
495. Zhang, Z., Luo, X., González, S., García, S., & Herrera, F. (2018, in press). DRCW-ASEG: One-versus-one distance-based relative competence weighting with adaptive synthetic example generation for multi-class imbalanced datasets. *Neurocomputing*.
496. Zhao, M., Chan, R., Chow, T., & Tang, P. (2014). Compact graph based semi-supervised learning for medical diagnosis in Alzheimer's disease. *IEEE Signal Processing Letters*, *21*(10), 1192–1196.
497. Zhao, S., Tsang, E., & Chen, D. (2009). The model of fuzzy variable precision rough sets. *IEEE Transactions on Fuzzy Systems*, *17*(2), 451–467.
498. Zhao, S., Tsang, E., Chen, D., & Wang, X. (2010). Building a rule-based classifier - a fuzzy-rough set approach. *IEEE Transactions on Knowledge and Data Engineering*, *22*(5), 624–638.
499. Zhao, X., Li, X., Chen, L., & Aihara, K. (2008). Protein classification with imbalanced data. *Proteins: Structure, Function, and Bioinformatics*, *70*(4), 1125–1132.
500. Zhao, Z., Fu, G., Liu, S., Elokely, K., Doerksen, R., Chen, Y., et al. (2013). Drug activity prediction using multiple-instance learning via joint instance and feature selection. *BMC Bioinformatics*, *14*(14), S16.
501. Zheng, E., Li, P., & Song, Z. (2006). Cost sensitive support vector machines. *Control and Decision*, *21*(4), 473.
502. Zhou, S., Chen, Q., & Wang, X. (2013). Active deep learning method for semi-supervised sentiment classification. *Neurocomputing*, *120*, 536–546.
503. Zhou, Y., & Goldman, S. (2004). Democratic co-learning. In *Proceedings of the 16th IEEE International Conference on Tools with Artificial Intelligence* (pp. 594–602). IEEE.
504. Zhou, Z. (2012). *Ensemble methods: Foundations and algorithms*. CRC Press.
505. Zhou, Z., Jiang, K., & Li, M. (2005). Multi-instance learning based web mining. *Applied Intelligence*, *22*(2), 135–147.
506. Zhou, Z., & Li, M. (2005). Tri-training: Exploiting unlabeled data using three classifiers. *IEEE Transactions on Knowledge and Data Engineering*, *17*(11), 1529–1541.
507. Zhou, Z., & Liu, X. (2006). Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering*, *18*(1), 63–77.
508. Zhou, Z., & Liu, X. (2010). On multi-class cost-sensitive learning. *Computational Intelligence*, *26*(3), 232–257.
509. Zhou, Z., Sun, Y., & Li, Y. (2009). Multi-instance learning by treating instances as Non-IID samples. In *Proceedings of the 26th International Conference on Machine Learning* (pp. 1249–1256). ACM.
510. Zhou, Z., Zhang, M., Huang, S., & Li, Y. (2012). Multi-instance multi-label learning. *Artificial Intelligence*, *176*(1), 2291–2320.
511. Zhu, B., Baesens, B., & vanden Broucke, S. (2017a). An empirical comparison of techniques for the class imbalance problem in churn prediction. *Information Sciences*, *408*, 84–99.
512. Zhu, J., Shi, J., Liu, X., & Chen, X. (2014). Co-training based semi-supervised classification of alzheimer's disease. In *Proceedings of the 19th International Conference on Digital Signal Processing* (pp. 729–732). IEEE.
513. Zhu, T., Lin, Y., & Liu, Y. (2017b). Synthetic minority oversampling technique for multiclass imbalance problems. *Pattern Recognition*, *72*, 327–340.
514. Zhu, X., Goldberg, A., Brachman, R., & Dietterich, T. (2009). *Introduction to semi-supervised learning*. Morgan and Claypool Publishers.