Fuzzy Rough and Evolutionary Approaches to Instance Selection

Datapuntselectie gebaseerd op de Vaagruwverzamelingenleer en Evolutionaire Algoritmen

Nele Verbiest

Dissertation submitted to the Faculty of Sciences of Ghent University
in fulfillment of the requirements for the degree of Doctor of Computer Science

Supervisors:

Prof. Dr. Chris Cornelis
Dr. Yvan Saeys

March 2014

Faculty of Sciences

Contents

Dankwoord

Ik heb het grote geluk gehad dat ik de voorbije vier jaar onbezorgd heb kunnen werken aan dit proefschrift, dankzij de onvoorwaardelijke steun en vriendschap die ik van vele mensen heb gekregen.
Eerst en vooral wens ik mijn promotor, Prof. Dr. Chris Cornelis te bedanken. Als lesgever van het vak computationele intelligentie maakte hij mij enthousiast voor wetenschappelijk onderzoek. Later heeft hij dit enthousiasme nog meer aangewakkerd als promotor van mijn master thesis. Het heeft lang geduurd eer hij mij ervan heeft kunnen overtuigen een doctoraat te starten, maar ik heb er nooit spijt van gehad, integendeel. Dankzij Chris ben ik in contact gekomen met vele onderzoekers overal ter wereld en heb ik aan verschillende goede universiteiten kunnen werken. Chris, bedankt voor alle kennis, voor alle tijd, voor alle goeie raad, voor alle kansen die je me hebt gegeven en voor de leuke tijden en boeiende gesprekken op conferenties en in Granada.
Veel dank ook aan de co-promotor van dit proefschrift, Dr. Yvan Saeys. Bedankt voor de interessante discussies die inspiratie brachten voor mijn methoden, voor het nalezen van mijn proefschrift en voor de goede samenwerking bij het doceren van computationele intelligentie.
Bedankt aan Prof. Dr. Hans De Meyer om promotor te zijn van het BOF-project en om praktische beslommeringen met betrekking tot mijn buitenlandse verblijven voor zijn rekening te nemen. Danku voor de interesse in mijn onderzoek en om als jurylid van mijn doctoraat de thesis zo grondig te lezen.
Ik wens ook Prof. Dr. Martine De Cock, hoofd van onze onderzoeksgroep, te danken voor alle steun de voorbije jaren. Als begeleider van de twee vakken die zij doceert heb ik enorm veel kunnen leren, bedankt. Dankzij haar ben ik ook in contact gekomen met Prof. Ankur Teredesai van de universiteit van Washington, Tacoma, waardoor ik daar een onderzoeksverblijf heb kunnen doen. De leuke

uitstapjes die ik met haar en haar gezin in Seattle heb gedaan hebben van mijn verblijf daar een mooie tijd gemaakt.
Verder veel dank aan alle andere leden van onze onderzoeksgroep. Prof. Dr. Etienne Kerre, dankzij het vak vaagheids- en onzekerheidsmodellen dat door u met het grootst mogelijke enthousiasme werd gedoceerd heb ik een tak in de wiskunde gevonden die mij uitermate boeit, dankuwel daarvoor. Bedankt Dr. Glad Deschrijver voor alle hulp met latex problemen, vage problemen en voor de goede samenwerking bij het doceren van vaagheids- en onzekerheidsmodellen. Dankzij de begeleidster van mijn masterthesis, Patricia, en de vele activiteiten die zij organiseerde op de S9 voelde ik mij snel thuis in de vakgroep. Bedankt Steven Schockaert, een beter voorbeeld als bureaugenoot had ik mij niet kunnen wensen. Bedankt Jeroen, om van mijn eerste conferentie een leuke conferentie te maken. Dankjewel, Marjon, om zo'n boeiende bureaugenoot te zijn en samen de laatste loodjes van onze doctoraten te doorspartelen. Gustavo, thanks for the great times at S9 and for helping me with my poor Spanish, I hope we meet some day in California. German, dankjewel voor alle hulp bij de praktische beslommeringen van mijn doctoraat. Lynn, bedankt voor de leuke tijden op conferentie en om samen de geheimen van vaagruwverzamelingen te doorgronden. Mushthofa, Elie and Golnoosh, it was great to have you as a colleagues in our research group, thank you. Sofie, Bart en Kim, bedankt om zulke fijne collega's te zijn.
Ook buiten onze onderzoeksgroep heb ik schitterende collega's. Een heel grote dankuwel aan Karel voor de drie heerlijke jaren als bureaugenoot. Catherine, dankzij jouw organisatietalent was er altijd wel iets te beleven bij TWIST, bedankt. Charlotte, het was leuk om samen met jou senior onder de doctoraatsstudenten te zijn! Herman, dankuwel om de middagpauze een stuk ecologischer aan te pakken. Machteld, bedankt voor de zoete bezoekjes aan ons bureau. Verder ook een heel grote dankuwel aan Veerle, Michaël, Bart M., Steven F., Steven K., Benoit, Ludger, Annick, Eva, Johan, Holger, Bart V.R., Bart J., Dieter, Niels, Bashir en Roy.
Ik wens ook uitdrukkelijk een aantal ex-twisters te bedanken. Jullie waren een mooi voorbeeld toen ik als doctoraatsstudent begon, ik heb veel van jullie geleerd en sommigen onder jullie zijn echte vrienden geworden. Dankuwel Virginie, Bert, Davy, Gilles, Jan, Heide, Nico, Stéphanie, Bart en Yun.
During the four years that I have been working on my Ph. D., I got the chance to explore the world and to work at different universities. I spent several months at the University of Granada. I would like to thank Prof. Francisco Herrera for giving me the opportunity to work together with him and his co-workers at his department. Prof. Herrera is an expert in the field and I am grateful and

proud that I could work with him. He has offered us innovative ideas, made sure that our papers were correct in every detail and taught me the importance of a good experimental study and evaluation. Although he is probably the most busy person I have ever met, he spent a lot of time discussing with me and Chris on our research. Muchas gracias, Paco, por todo lo que me ha enseñado y para los tiempos encantadoras en Granada y Hyderabad.

The times at the University of Granada were particularly interesting because many of the Ph. D. students are working in the same field as me. I could rely on them for many discussions and practical problems.

A big thank you to Joaquín, for answering my e-mails with questions within the minute. I'm not sure if I would ever have been able to work with the software platform Keel if it wasn't for the good instructions he gave me. Thank you for the good collaboration on our common research projects, it was an absolute pleasure to work together with you. Thank you Victoria, for helping me with the imbalanced problems, and for making me feel at home in Granada. Thank you Isaac, for helping me with practical issues and for all the help with access to the Granada cluster. Many other Ph. D. students made my stay in Granada very pleasant, a big thank you to all of them. Many thanks also to Enislay Ramentol from the University of Camaguey, Cuba, for the collaboration on imbalanced problems.

In 2013 I stayed three months at the university of Washington, Tacoma, to work on health care problems in data mining. Thank you very much, Prof. Dr. Ankur Teredesai, to invite me to your institute and to give me the opportunity to work on this project. During my stay I met very interesting people, I saw very interesting talks from many good speakers from the industry and I worked on hard but interesting problems. Thank you, Prof. Dr. Senjuti Basu Roy and Dr. Si-Chi Chin for the interesting discussions and the good collaboration. Thanks to all the students of the institute for the interesting seminars and nice discussions. Thank you, Kiyana, Jayshree, Ani, Naren, Sid, Muaz, Deepthi, Swapna for driving me around in Tacoma, for going to Portland with me, for the nice poker games, for the cake-facial and for introducing me to great Indian food.

After my stay in Tacoma, I stayed one month at the University of Córdoba, Spain, to work on ensembles of prototype selection methods. Muchas gracias, Prof. Nicolas García-Pedrájas, for inviting me to your research group and for the interesting suggestions on our work.

A big thank you to Dr. Richard Jensen from the Aberystwyth University in Wales for his collaboration on the classification part of this thesis. Also, thank you for reading the thesis in such great detail.

Also, thank you to Prof. Dr. Jesús Medina from the University of Cádiz for the

collaboration on Multi-Adjoint Fuzzy Rough Sets. Although out of the scope of this thesis, I have learned a lot from working on this theoretical research field. Thanks for your nice visits to Granada and for being the perfect host when Chris and I visited you at the University of Cádiz.
Naast al deze collega's kan ik ook privé rekenen op heel wat steun. Bedankt aan mijn super-vrienden: Sofie, Lore en Matthias, Stefanie en Darel, Danielle, Elien, Vero en Frederik, Tineke en Frederik, Emmy, Sofie en Hans, Gert en Peter, Lies en Jelle, Bas en Jan, Frederik, Ineke en Thomas, Paul en Cassandra, een ongelooflijk dikke merci voor alles, jullie zijn stuk voor stuk schatten.
Tenslotte zou ik nog een hele thesis kunnen volschrijven over de steun en liefde die ik van mijn familie heb gekregen. Ik ga het hierbij houden: dankuwel Fien, Dries en Anke, al mijn lieve neven, nichten, nonkels, tantes en grootouders, voor de interesse in mijn onderzoek en jullie onvoorwaardelijke steun en liefde. Bedankt, mama en papa, voor de voorbije geweldige 27 jaar. Dankzij jullie heb ik elk talent en elke kans die ik heb gekregen ten volle kunnen benutten, dankjewel.

Nele
Juni 2014

Dit onderzoek werd financieel gesteund door het Bijzonder Onderzoeksfonds van de UGent. Verschillende buitenlandse verblijven werden gefinancieerd door het Fonds Wetenschappelijk Onderzoek Vlaanderen. Experimenten in deze thesis werden uitgevoerd op de centrale HPC infrastructuur van UGent, beheerd door het Vlaams Supercomputer Centrum en gefinancierd door UGent, de Hercules stichting en de Vlaamse Overheid, afdeling EWI. Andere experimenten werden uitgevoerd op de Hercules rekeninfrastructuur van de Universiteit Granada.

1. Introduction

In 2006, Clive Humby said at the ANA Senior marketer's summit that *Data is the new oil* and this might not be far from the truth. In its raw format oil is not valuable, but once it is refined and changed into petrol, plastic, chemicals etc., it is profitable. The same holds for data: tables containing numbers are not helpful on their own, but the knowledge abstracted from data is indispensable to companies, research, governments and other interest groups.
Data mining, the process of extracting information from data, has become an important research topic over the last decades, confirmed by numerous publications and conferences in the field[1]. The information revealed by data mining processes is limitless. Consider for instance a dataset consisting of supermarket customers described by characteristics such as age, sex and living place. Through clustering [102, 110, 173] techniques, one can detect groups of customers that are highly related to each other. If one has records about the weekly expenses of customers, one can predict the weekly expenses of new customers based on their characteristics through regression [16, 71, 90, 136]. If it is known which customers in the data are interested in a certain product, classification [1] algorithms can predict if a new customer will be interested in that product or not.
There are many other examples of information that can be derived from data, but in this work we focus on classification. Formally, given a dataset of instances described by conditional features and a decision feature (class), classifiers aim to predict the class of a new instance given its conditional features. Most classifiers first build a model based on the data and then feed the new instance to the model to predict its class. For instance, Support Vector Machines (SVMs, [25, 128, 175, 186]) construct a function that models a separating border between the different classes in the data, and the value of that function for the

[1] The Microsoft Academic Search repository has over 70000 publications on data mining, of which 40000 were published in the last ten years

new instance then determines to what class it most likely belongs. Decision trees [130, 16, 93] generate rules from the data following a tree structure that predict the class of a new instance. Another example is Naive Bayes (NB, [43]), this classifier assumes that all features are independent and calculates probabilities from the given data such that the probability that a certain instance belongs to a certain class given its conditional features can be predicted. An example of a somewhat different classifier is K Nearest Neighbors (KNN, [32]). This classifier has no modeling phase. A new instance is classified directly by looking up the closest instances in the data and classifying it to the class mostly occurring among those nearest neighbors.

Before the data can be used to build the classification model and to classify new instances, it generally needs preprocessing. For instance, there can be missing values [10, 64, 66, 67] in the data that need to be imputed. Some classifiers are not able to handle continuous data, so discretization [59, 89, 103] should be applied. Preprocessing can also be used to speed up or improve the classification process afterward. A good example is Feature Selection (FS, [68, 92, 104, 138]), where redundant or misleading features are removed from the data.

This thesis focuses on instance selection [27, 55, 98, 159, 183], the preprocessing technique where a subset of the instances in the data is selected before using it for classification. Although this technique is less frequently applied in applications than FS, many researchers have proposed instance selection techniques and have experimentally demonstrated that instance selection can indeed improve and speed up classification.

Instance selection has mostly been applied in the context of KNN classification. In that case, it is referred to as Prototype Selection (PS, [55]). The reason why researchers mostly focus on PS for KNN classification is that it has a direct influence on the classification as the KNN classifier has no buffering classification model between removing instances and classifying new ones. In this thesis we principally focus on PS, but we also study if instance selection can be used for SVMs, in that case we refer to it as Training Set Selection (TSS, [27, 183]).

The algorithms proposed in this thesis are based on evolutionary algorithms on the one hand and fuzzy rough set theory on the other hand. Evolutionary algorithms [63] are search strategies to solve optimization algorithms inspired on biological phenomena. Different evolutionary algorithms use different strategies to create new candidate solutions and to guide a search to improve their quality. For instance, ant colony optimization [44] algorithms can find good paths through graphs, inspired by the behavior of ants that leave pheromones on the route between their colony and food. Particle swarm optimization algorithms [88, 127] move candidate solutions (particles) around in the search-space, mak-

ing sure that the solution is locally optimal but also heading towards the global best solution. Structural optimization problems can be solved using artificial bee colony algorithms [86]. These algorithms are based on the behavior of honey bees in a swarm, where bees inform each other using waggle dances about where to find nectar.

In this work we consider Genetic Algorithms (GAs, [63]), search strategies inspired on biological evolution. The key component of GAs is the fitness function that evaluates candidate solutions. These solutions to the optimization problem are contained in a population that evolves over several generations. A new generation is formed by recombination and mutation of individuals in the previous population. GAs have been successfully applied in many fields of data mining like regression [174], classification [7, 35], FS [22, 36, 99, 177] and instance selection [21, 97, 98]. In this work we further elaborate on the latter two.

Apart from evolutionary algorithms, we also use fuzzy rough set theory [29, 45, 46] to tackle instance selection. It is the hybridization of rough set theory [124] and fuzzy set theory [181]. The former, rough set theory, models inconsistencies and incompleteness in data. Consider for example a classification dataset and a class A within that dataset. An inconsistency occurs when two instances have the same feature values, but one instance belongs to class A and the other does not. In that case, the instances do not belong to the *lower approximation* of A, but do belong to the *upper approximation*. More specifically, the lower approximation of A consists of all instances that belong to A and for which there are no instances with the same features but a different class from A. In other words, these are instances that definitely belong to A. The lower approximation consists of instances that belong to A or instances that have exactly the same features as some instance in A. These are instances for which some inconsistency exists.

Rough set theory has been used extensively in data mining. However, the model has one important drawback, being that it cannot handle continuous features in the dataset. This problem can be alleviated using discretization techniques but this generally comes with information loss. Therefore, many authors have studied how rough set theory can be extended using fuzzy set theory [29, 45, 46] which extends traditional set theory in the sense that instances can belong to a set to a certain degree between 0 and 1. The keystone of fuzzy rough set theory is the fuzzy relation, which expresses gradual indiscernibility between instances. A fuzzy rough set consists of the lower and upper fuzzy rough approximation, two fuzzy sets to which instances belong to a certain extent.

Fuzzy rough set theory has been widely used in several data mining topics. Most research has been done on FS with fuzzy rough sets [14, 30, 84, 85], mainly focusing on preserving the predictive power of datasets with the least

features possible. Some preliminary research has been done on using fuzzy rough set theory for instance selection [80] and its combination with FS [36, 120]. Apart from using fuzzy rough set theory for preprocessing, it has also been used successfully to tackle classification directly, for instance in rule induction [42, 65, 77, 82, 106, 157, 166, 189], improving K nearest neighbor classification [15, 79, 81, 109, 129, 142], enhancing decision trees [13, 47, 83, 185] and boosting SVMs [25, 26, 128, 175, 186].

In this thesis we study instance selection using fuzzy rough set theory and genetic algorithms. Using genetic algorithms for instance selection has been proven to be very successful, this is why we further elaborate on this research direction in this work. Using fuzzy rough set theory for instance selection is not explored much, but we think that fuzzy rough set theory is a good model to assess the quality of instances, as it is designed to model inconsistencies in the data.

After introducing preliminaries necessary for the understanding of the thesis in Chapter 2, we start off by introducing a new type of fuzzy rough sets, called Ordered Weighted Average (OWA, [176]) fuzzy rough sets [31] in Chapter 3. This model is motivated by the fact that the traditional fuzzy rough set model is not robust against noisy data: changing one value in the data can change the membership values of instances to the fuzzy rough lower and upper approximation drastically. Moreover, existing robust extensions of fuzzy rough set models are not able to preserve the theoretical properties of the traditional fuzzy rough set model, or need parameter tuning to perform well in data mining applications. Using OWA operators to soften the strict minimum and maximum operators in the formulas for the fuzzy rough lower and upper approximations, we are able to construct a robust fuzzy rough set model that preserves the most important theoretical properties.

In Chapter 4 we develop a PS technique based on fuzzy rough set theory, called Fuzzy Rough Prototype Selection (FRPS, [159, 160]). The main idea is that the quality of instances is assessed using fuzzy rough set theory, and that a threshold to decide which instances to retain is tuned automatically. In the experimental study we derive that using the OWA fuzzy rough set model improves upon the traditional fuzzy rough set model. Additionally, we conclude that FRPS significantly improves the state of the art in PS, and that FRPS is faster than the genetic approaches to PS.

Motivated by the good performance of FRPS, we study its combination with FS in Chapter 5. A possible research path could be to use a fuzzy rough approach for the FS part, but as most FS techniques based on fuzzy rough set theory aim to maintain the predictive power of the data rather than improving it, we decide not to do so. Instead, we use a genetic approach to FS, as this technique has shown

to perform well in experimental studies. We consider different settings, where FS is applied first followed by PS and the other way around. Additionally, we propose a simultaneous approach, called SIM-FRPS-SSGA [38], that carries out a genetic algorithm for FS and applies FRPS at certain stages of the algorithm. The experimental approach shows that the simultaneous approach improves the sequential application of its components. Moreover, SIM-FRPS-SSGA significantly improves the baselines in FS and PS.

In Chapter 4 and 5 we developed preprocessing techniques for KNN classification. In Chapter 6, we study if these techniques are also advantageous for SVM classification. As mentioned before, TSS does not affect SVM classification directly, as the SVM classification model buffers between the selection of instances and the classification of new instances. This is reflected in the results of the experimental study: the effect of TSS on SVMs is more limited than the influence of PS on KNN. We adjust the PS algorithms for SVMs by calculating the training accuracy in the evaluation functions using SVMs instead of KNN. Some of the techniques that work well for KNN do not perform well for SVMs, like FRPS. On the other hand, genetic approaches adjusted for SVMs are able to significantly improve SVM classification. The experimental study additionally shows that our FS approaches decrease the performance of SVMs.

We turn back to PS for KNN classification in Chapter 7 and elaborate on genetic approaches to PS. We observe that genetic PS algorithms only use the best prototype subset found during the course of the search algorithm, while many good but suboptimal solutions are found throughout the algorithm. We design a framework were multiple prototype subsets are used to classify test data. In order to classify a new instance, the prototype subsets that perform well in the neighborhood of that instance are used to classify it. We apply this framework to the Generational Genetic Algorithm (GGA, [97, 98]) and call the approach GGA improved by ensembles (GGA-ENS). The experimental study shows that GGA-ENS significantly outperforms GGA in the traditional setting with PS, and that GGA-ENS only requires a small additional running time cost. Moreover, GGA-ENS outperforms GGA for a smaller number of evaluations, and applying GGA with 10 000 evaluations has similar results as using GGA-ENS with 1 000 evaluations, meaning that GGA-ENS attains the same results as GGA with a fraction of the running time required.

In Chapter 8 we study how FRPS can be used for imbalanced data, which is data with one or more classes underrepresented. Classifying this type of data needs special attention, as traditional classification techniques typically classify the majority instances correctly and neglect minority instances. We adjust FRPS for imbalanced data and call the new technique Fuzzy Rough Imbalanced Prototype

Selection (FRIPS, [163]). This technique tunes the threshold in such a way that both the minority and the majority instances are classified well. An experimental study shows that FRIPS improves FRPS for imbalanced data. Additionally, we study if using FRPS after balancing the data using the Synthetic Minority Oversampling TEchnique (SMOTE, [23]) is valuable. We conclude that SMOTE-FRPS does indeed improve SMOTE and other state-of-the-art preprocessing techniques for imbalanced data.

In the setting of FRPS followed by KNN, an instance is either removed or either retained in the data. In Chapter 9 we explore a different research path, where instances are weighted according to their quality based on fuzzy rough set theory. We improve the Fuzzy Nearest Neighbor (FNN, [87]) algorithm, which classifies instances using a nearest neighbor approach and weighting the importance of the neighbors depending on the distance between these neighbors and the instance to be classified. We improve this approach by additionally weighting the neighbors according to their quality, based on fuzzy rough set theory. We propose two methods, POSitive region Nearest Neighbor (POSNN, [161]) and Quality, Frequency and Similarity Nearest Neighbor (QFSNN, [162]), in which the former multiplies the weights whereas the latter tunes the weights given to the components automatically.

We conclude the thesis with an overview of the results obtained and suggestions for further research in the last concluding chapter.

2. Preliminaries

In this chapter we provide preliminaries that are necessary for the understanding of the rest of this thesis. In Section 2.1 we present the two classifiers that are used in this work and techniques to evaluate classifiers. In Section 2.2 we give a general background on evolutionary algorithms. In Section 2.3 we review the basics of fuzzy rough set theory, and in Section 2.4 we explore the state-of-the-art in instance selection.

2.1 Classification

Classification and regression problems can be modeled by means of a decision system $(U, \mathcal{A} \cup \{d\})$, which consists of the set U containing instances that are described by the conditional attributes $\mathcal{A}$ and a decision attribute d. In some applications, multiple decision attributes are given, but in this work we only consider one decision attribute. The value of an instance $x \in U$ for an attribute $b \in \mathcal{A} \cup \{d\}$ is denoted by $b(x)$. The value $d(x)$ is continuous for regression problems and takes values in a finite set for classification problems. As this work only considers classification problems we assume that $d(x)$ takes values in a finite set from now on.
Classification methods aim to predict the class $d(t)$ of a new target instance t, based on the knowledge in the given training data U. That is, the conditional attribute values of t are given and $d(t)$ needs to be determined making use of the training instances U and their attribute and class values.
Many classification techniques are available. In this work, we consider two of the most well-known techniques, K Nearest Neighbor (KNN) classification and Support Vector Machine (SVM) classification. The first is a lazy learning technique, i.e. new instances can be classified immediately without training a model. SVM classification requires a model to be trained, but is very fast in the testing phase when new instances need to be classified. These classifiers are

discussed in Section 2.1.1 and 2.1.2, respectively. In Section 2.1.3 we discuss how classifiers can be evaluated.

2.1.1 K Nearest Neighbor (KNN) classification

One of the most simple and easy-to-understand classification methods is KNN [32]. It is model-free, which means that there is no training phase. In order to classify an instance t, the K training instances that are closest to the instance t are looked up, and t is classified to the class that occurs most among these K nearest neighbors. In case of ties, a random choice is made among the classes that occur most frequently.
In order to determine the K nearest neighbors, a distance measure needs to be chosen. We define the distance between two instances x and y as follows:

$$\text{dist}(x, y) = \sum_{a \in \mathcal{A}} \text{dist}_a(x, y), \tag{2.1}$$

where the distance between two instances x and y for one attribute a is defined by:

$$\text{dist}_a(x, y) = (a(x) - a(y))^2 \tag{2.2}$$

for a continuous attribute and by

$$\text{dist}_a(x, y) = \begin{cases} 1, & \text{if } a(x) \neq a(y) \\ 0, & \text{otherwise} \end{cases} \tag{2.3}$$

for a discrete attribute. Note that $\sqrt{dist(x, y)}$ corresponds to the Euclidean distance if there are no discrete attributes.
As all attributes get equal weights in the distance function, it is important that the data is normalized before applying KNN. Otherwise, the determination of the nearest neighbors would be highly dependent on the scales of the attributes. KNN has many advantages, the main one being that no assumptions need to be made about the data. On the other hand, KNN needs high storage requirements and has low efficiency caused by multiple computations of distances between the test and training instances.

2.1.2 Support Vector Machine (SVM) classification

For each instance $x \in U = \{x_1, \ldots, x_n\}$ we denote by $\mathbf{x}$ its feature vector, that is,

$$\mathbf{x} = (a_1(x), \ldots, a_m(x))^t.$$

We first consider the two-class case, where $d(x)$ is either -1 or 1 for each $x \in U$. In Section 2.1.2.1 we consider a very simple form of support vector machines, separating hyperplanes, where it is assumed that the two classes can be separated linearly. In Section 2.1.2.2 where support vector classifiers are presented we omit this restriction and allow that instances can fall on the wrong side of the separating border. In Section 2.1.2.3 we take this approach one step further and use kernels to allow for non-linear borders. In Section 2.1.2.4 we discuss how the SVM optimization problem can be tackled efficiently, and in Section 2.1.2.5 we discuss how the probabilities returned by the SVM can be rescaled. Finally, in Section 2.1.2.6 we discuss how classification problems with more than two classes can be approached using SVMs.

2.1.2.1 Separating hyperplanes

The simplest form of SVMs are separating hyperplanes. A hyperplane $H_{\boldsymbol{\beta}}$ represented by a function f is defined by

$$H_{\boldsymbol{\beta}} \leftrightarrow f(\mathbf{x}) = \boldsymbol{\beta_0} + \boldsymbol{\beta}^t\mathbf{x} = 0, \tag{2.4}$$

where $\boldsymbol{\beta}$ and $\boldsymbol{\beta_0}$ are m-dimensional vectors. A hyperplane is a separating hyperplane if for all instances $x \in U$ it holds that $f(\mathbf{x}) < 0$ if $d(x) = -1$ and $f(\mathbf{x}) \geq 0$ if $d(x) = 1$. We say that the data U is linearly separable if such a separating hyperplane exists. In that case there might exist multiple separating hyperplanes. The optimal separating hyperplane is selected, it maximizes the sum of the distances between the hyperplane and the closest instances from each class (called the support vectors), also referred to as the margin. An example of linearly separable data is given in Figure 2.1.1. There are infinitely many separating hyperplanes, the one indicated with a solid line is optimal.
It can be shown that this problem is equivalent to solving the optimization problem that finds $\boldsymbol{\beta}$ and $\boldsymbol{\beta_0}$ such that

$$\frac{1}{2}||\boldsymbol{\beta}|| \tag{2.5}$$

is minimized, subject to

$$\forall x \in U : d(x)(\boldsymbol{\beta}^t\mathbf{x} + \boldsymbol{\beta_0}) \geq 1. \tag{2.6}$$

This can be reformulated as the minimization problem that finds $\boldsymbol{\beta}$ and $\boldsymbol{\beta_0}$ such that the following Lagrange primal function

$$L_P = \frac{1}{2}||\boldsymbol{\beta}||^2 - \sum_{i=1}^{n} \alpha_i(d(x_i)(\boldsymbol{\beta}^t\mathbf{x_i} + \boldsymbol{\beta_0}) - 1) \tag{2.7}$$

is minimal, where $\forall i \in \{1, \ldots, n\}, \alpha_i \geq 0$ are the Lagrange multipliers. Setting the derivatives w.r.t. $\boldsymbol{\beta}$ and $\boldsymbol{\beta_0}$ equal to zero yields the following equations:

$$\boldsymbol{\beta} = \sum_{i=1}^{n} \alpha_i d(x_i) \mathbf{x_i} \tag{2.8}$$

$$0 = \sum_{i=1}^{n} \alpha_i d(x_i). \tag{2.9}$$

This means that $\boldsymbol{\beta}$ can be calculated if we know the optimal values for the Lagrange multipliers. Substituting the above first-order conditions in the Lagrange primal function yields the Wolfe dual function

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j \mathbf{x_i}^t \mathbf{x_j} \tag{2.10}$$

that needs to be maximized subject to the conditions $\forall i \in \{1, \ldots, n\} \alpha_i \geq 0$ and $0 = \sum_{i=1}^{n} \alpha_i d(x_i)$. This is a convex optimization problem, and the solution needs to satisfy the Karush-Kuhn-Tucker (KKT) conditions. They are given by Equations (2.6), (2.8), (2.9), $\forall i \in \{1, \ldots, n\} \alpha_i \geq 0$ and

$$\forall i \in \{1, \ldots, n\} \alpha_i (y_i (\boldsymbol{\beta}^t \mathbf{x_i} + \boldsymbol{\beta_0}) - 1) = 0 \tag{2.11}$$

From these KKT conditions, it follows that the optimal parameter vector $\boldsymbol{\beta}$ is a linear combination of the feature vectors $\mathbf{x_i}$ for which $\alpha_i > 0$, and it can be shown that these correspond to the support vectors. This means that the separating hyperplane mostly takes into account points close to the decision boundary and gives zero weights to all other points.
After the optimal hyperplane $H_{\boldsymbol{\beta}} \leftrightarrow f(\mathbf{x}) = 0$ is found, a new test instance t can be classified by applying f to it. If $f(\mathbf{t}) \geq 0$, t should be classified to the positive class and else to the negative class.

2.1.2.2 Support vector classifiers

In many cases the data is not linearly separable as classes may overlap. Support vector classifiers deal with this problem by allowing some of the instances to fall on the wrong side of the separating hyperplane. An example is given in Figure 2.1.2, where it is impossible to find a line that perfectly separates the classes.

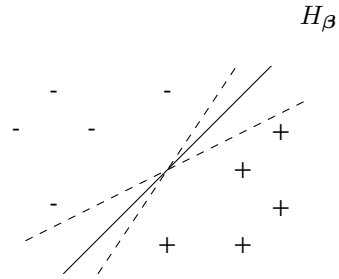


Figure 2.1.1: Linearly separable data and separating hyperplanes, the separating hyperplane indicated with a solid line optimizes the margin.

Support vector classifiers associate a slack variable ξ_i to each instance $x_i \in U$, where $\xi_i \geq 0$. The condition in Equation (2.6) is modified to:

$$\forall x_i \in U : d(x_i)(\boldsymbol{\beta}^t \mathbf{x_i} + \boldsymbol{\beta_0}) \geq 1 - \xi_i. \tag{2.12}$$

The value ξ_i indicates how much each point x_i falls at the wrong side of the margin. The sum of these values is bounded:

$$\sum_{i=1}^{n} \xi_i \leq Q, \tag{2.13}$$

with Q the maximum number of instances that are allowed to fall on the wrong side of the separating hyperplane. The problem can be formulated as the optimization problem that finds $\boldsymbol{\beta}$ and $\boldsymbol{\beta_0}$ such that

$$\frac{1}{2}||\boldsymbol{\beta}||^2 + C\sum_{i=1}^{n} \xi_i \tag{2.14}$$

is minimized, subject to

$$\forall i \in \{1, \ldots, n\} : \xi_i \geq 0 \tag{2.15}$$

and Equation (2.12). The cost parameter C takes over the role of Q. A large value of C only allows few misclassifications, while a small value of C allows for more misclassifications. The above problem is equivalent to finding $\boldsymbol{\beta}, \boldsymbol{\beta_0}$ and

$\xi_1, \ldots, \xi_n$ such that

$$L_P = \frac{1}{2}||\boldsymbol{\beta}||^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i(d(x_i)(\boldsymbol{\beta}^t\mathbf{x_i} + \boldsymbol{\beta_0}) - (1-\xi_i)) - \sum_{i=1}^{n}\mu_i\xi_i \quad (2.16)$$

is minimal, where $\forall i \in \{1, \ldots, n\}, \alpha_i, \mu_i \geq 0$ are the Lagrange multipliers. The optimization problem is subject to

$$\boldsymbol{\beta} = \sum_{i=1}^{n}\alpha_i d(x_i)\mathbf{x_i} \quad (2.17)$$

$$0 = \sum_{i=1}^{n}\alpha_i d(x_i) \quad (2.18)$$

$$\forall i \in \{1, \ldots, n\} : \alpha_i = C - \mu_i. \quad (2.19)$$

It follows that the solution for $\boldsymbol{\beta}$ is again a linear combination of the instances $x_i \in U$ with $\alpha_i > 0$. It can be shown that these points are exactly on the boundary of the margin around the separating hyperplane, the support vectors.

2.1.2.3 Support vector machines

Both separating hyperplanes and support vector classifiers separate the classes linearly. In some cases this is not meaningful, as illustrated in Figure 2.1.3. In order to have more flexible borders, kernel functions can be used. Assume that we have a function h that maps instance vectors $\mathbf{x}$ from the original feature space to another feature space of a higher dimension. The SVM can then be applied to the transformed training data in the new feature space.
SVMs only use $\mathbf{x}$ in the inner product with other vectors $\mathbf{y}$, which means that calculating $h(\mathbf{x})$ or $h(\mathbf{y})$ separately is not necessary, it suffices to know the value of $h(\mathbf{x})h(\mathbf{y})$. A kernel function K can be used to calculate these values. It is a function that corresponds to the inner product of $h(\mathbf{x})$ and $h(\mathbf{y})$ in the new feature space:

$$\forall \mathbf{x}, \mathbf{y} \in U : K(\mathbf{x}, \mathbf{y}) = h(\mathbf{x})^t h(\mathbf{y}). \quad (2.20)$$

The optimization problem is then the maximization problem that finds $\alpha_1, \ldots, \alpha_n$ such that the Lagrange function

$$L_D = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j d(x_i)d(x_j)K(\mathbf{x_i}, \mathbf{x_j}) \quad (2.21)$$

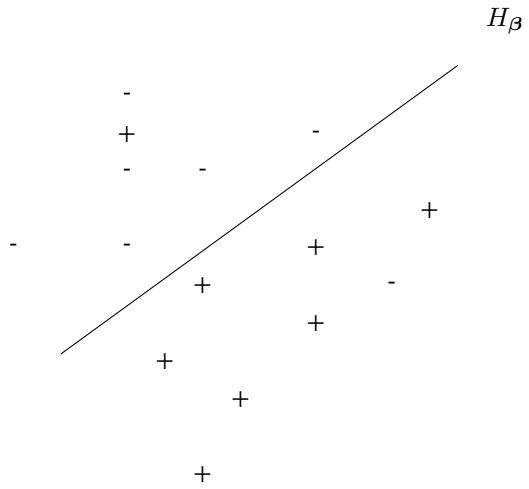


Figure 2.1.2: Example of data that is not linearly separable. The data is separated by the hyperplane H_β and two instances fall on the wrong side of H_β.

is maximized under the constraints

$$\sum_{i=1}^{n} \alpha_i d(x_i) \tag{2.22}$$

and

$$\forall i \in \{1, \ldots, n\} 0 \leq \alpha_i \leq C. \tag{2.23}$$

The simplest kernel function is the linear kernel:

$$K(\mathbf{x}, \mathbf{y}) = \mathbf{x}^t \mathbf{y} \tag{2.24}$$

When this kernel is used, the SVM model is equivalent to the support vector classifier.
A kernel that is very often used is the Radial Basis Function Kernel (RBF), defined as follows for $x, y \in U$:

$$K(\mathbf{x}, \mathbf{y}) = exp(-\frac{||\mathbf{x} - \mathbf{y}||^2}{2\delta^2}). \tag{2.25}$$

The parameter $\delta > 0$ is called the bandwidth and determines how smooth the decision boundary of the SVM is: a low value of δ results in a very irregular

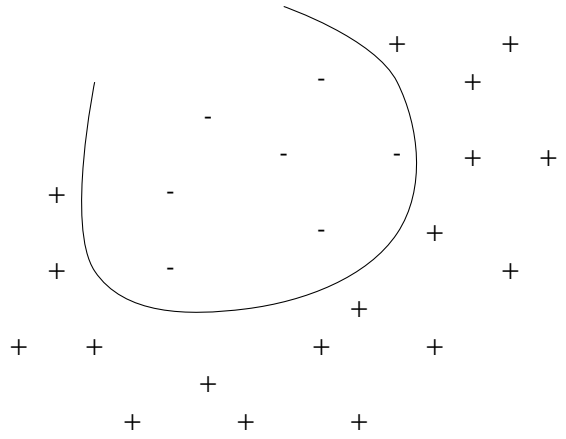


Figure 2.1.3: Data where separating the data linearly is not meaningful. Using kernels more flexibility for the separating boundaries is allowed.

boundary, while a high value of δ results in an SVM with a smooth decision boundary.

2.1.2.4 Sequential Minimal Optimization (SMO)

The Quadratic Program (QP, [122]) to be solved in Equation (2.21) is solved when the matrix $Q_{ij} = d(x_i)d(x_j)K(\mathbf{x_i}, \mathbf{x_j})$ is positive semi-definite and when the KKT conditions are fulfilled:

$$\begin{array}{c} \forall i \in \{1, \ldots, n\} : \\ \alpha_i = 0 \Rightarrow d(x_i)f(\mathbf{x_i}) \geq 1 \\ 0 < \alpha_i < C \Rightarrow d(x_i)f(\mathbf{x_i}) = 1 \\ \alpha_i = C \Rightarrow d(x_i)f(\mathbf{x_i}) \leq 1. \end{array} \tag{2.26}$$

Solving this QP using traditional QP solving techniques is not feasible for SVM problems, as the matrix that needs to be stored is quadratic in the number of instances. Sequential Minimal Optimizaton (SMO, [125]) is a much faster alternative for traditional QP solving techniques. The main idea is that the problem is decomposed in several smaller optimization problems that can be solved analytically. At each step, two Lagrange multipliers are jointly optimized. These two Lagrange multipliers are chosen such that at each step in the process

the smallest possible optimization problem at that point is solved. After the optimal values are found for the two Lagrange multipliers, the SVM is updated to reflect the new optimal values.
The main advantage of this strategy is that solving the optimization problem for two Lagrange multipliers can be done analytically. As a result, each sub-problem can be solved so fast that the entire QP can be solved quickly.

2.1.2.5 Platt's scaling

As discussed previously, the separating hyperplanes are represented by a function f that takes values in $(-\infty, +\infty)$. That is, a test instance t is classified to the negative class if $f(\mathbf{t}) \in (-\infty, 0]$ and to the positive class if $f(\mathbf{t}) \in (0, \infty)$ However, it is more useful to obtain probabilities. Therefore, Platt's method [126, 153] estimates the probabilities $P(d(t) = 1|f)$. A sigmoid model is used as follows:

$$P(d(t) = 1|f) = \frac{1}{1 + exp(Af + B)}. \tag{2.27}$$

The parameters A and B are estimated using maximum likelihood estimation on the training data. This scaling can also be seen as training the model to find a better threshold: instead of using the standard 0 as threshold to classify test instances, the model is trained to find a better threshold.

2.1.2.6 The multi-class case

The discussed methods apply to two-class problems. A traditional approach to handle multi-class problems is pairwise coupling [52, 73, 123], also referred to as all-versus-all, where the multi-class problem is decomposed in all possible multiple two-class problems and the majority voting principle is applied. For instance, when there are k classes, for each pair of classes i and j with $i, j \leq k$ and $i \neq j$, a binary SVM is constructed. A new instance is classified by all classifiers, and each class gets a vote if the new instance is classified to that class. The class with the highest number of votes is the final class returned for that instance.
Another approach is the so-called one-versus-all technique [52]. In this case, k training datasets are considered, where in each dataset one class is the positive class and the remaining classes form the negative class. The SVM is trained on each of these training datasets and the target instance t is classified by each SVM. Each SVM returns a probability value p expressing the confidence that t should be classified to the positive class. Finally, t is classified to the class for which this

probability is maximal.
There is no agreement in the literature about which strategy to use. Some authors argue that pairwise coupling is more practical than one-versus-all as the training process is quicker and moreover pairwise coupling is more accurate[4]. Other authors disagree and claim that the one-versus-all strategy is equally accurate [133].

2.1.3 Evaluation of classification

In this section we discuss three important steps that are taken when evaluating classifiers. The classifier is applied to multiple datasets, using a validation scheme (Section 2.1.3.1). The classification performance is then measured using an evaluation measure (Section 2.1.3.2). Once these evaluation measures are obtained for each dataset, classifiers can be compared among each other using statistical tests (Section 2.1.3.3).

2.1.3.1 Validation Schemes

A classifier is usually evaluated in a setting with both train and test data. For instance, support vector machines use the train data to find support vectors, and the KNN technique uses the entire train data as model. In order to evaluate the classifier built on the train data, test data is needed. The classifier labels each example in the test data and this classification can then be evaluated using evaluation measures.
The oldest validation methods used the entire data set both for training and testing. Obviously the resulting performance will be too optimistic [100], as the classes of the instances are known in the model.
A more reasonable way to evaluate a classification method on a data set is hold out evaluation [40], which splits the data into two parts. The train data is usually bigger than the test data, typically the train data doubles the size of the test data. Although this design is simple and easy to use, it has some disadvantages, the main one being that the data is not fully explored, that is, the evaluation is only carried out on a small fraction of the data. Moreover, it can happen that the instances included in the test data are too easy or too difficult to classify, resulting in a misleading high or low performance. Another problem that might occur is that instances essential to build the model are not included in the training set. This problem can be partially alleviated by running the hold-out evaluation several times, but still it might happen that some essential instances are never included in the train data or that instances hard to classify are never evaluated

in the test data.
In order to deal with this problem, a more systematic approach to repeat the hold out evaluation was developed. The widely used evaluation design called K-fold cross validation [149], splits the data into K equal parts, and each of these parts is classified by a model built on the remaining $K-1$ parts. The main advantage of this technique is that each data point is evaluated exactly once.
The choice of K is a trade-off between bias and variance [91]. For low values of K, the sizes of the training sets in the K-fold cross validation procedure are smaller, and the classifications are more biased depending on how the performance of the classifier changes with the instances included in the train data and with sample size. For instance, when $K = 2$, the two training sets are completely different and only cover one half of the original data, so the quality of the predictions can differ drastically for the two train sets. When $K = 5$, all train sets have 60 percent of the data in common, so the bias will be lower. For high values of K, the variance of the procedure is higher due to the stronger dependence on the train data, as all training sets are very similar to one another. Typical good values for K are 5 or 10. When K equals the data size, K-fold cross validation is referred to as Leave-One-Out cross validation. In this case, each instance is classified by building a model on all remaining instances and applying the resulting model on the instance.
Finally, we note that the data should be divided in folds carefully. Stratification processes [41, 117, 118, 119, 145, 184] make sure that the global data distribution is reflected in the separate folds.

2.1.3.2 Evaluation Measures

Once a validation scheme is chosen, the classifier can build its model on the training data and classify the test instances based on this model. Afterwards, evaluation measures are needed to describe how well this classification is done. We study how we can assess the quality of a given classifier based on a list of the real classes of instances and the predicted classes[1].
A representation that is very useful when evaluating a classifier is the so-called confusion matrix M. The dimension of the squared matrix M is k, the number of classes, and the entry M_{ij} denotes how many times an instance with real class c_i was classified as c_j $(i, j \in \{1, \ldots, k\})$. Based on the confusion matrix, many metrics can be defined.

[1]Note that we only consider evaluation measures for discrete classifiers, i.e. a classifier that returns a class as label, as opposed to probabilistic classifiers that return for each class the probability that the instance belongs to it.

We first discuss metrics that are only defined for binary classification problems. In the binary case, where there are two classes namely the positive (P) and the negative (N) class, we denote by True Positives (TP) and True Negatives (TN) the number of correctly classified positive and negative instances respectively. False Negatives (FN) stands for the number of instances that are predicted negative but that are actually positive, while the False Positives (FP) are the number of instances that are falsely classified to the positive class.
The *recall* (also referred to as true positive rate or sensitivity) is the number of true positives compared to the number of truly positive instances:

$$\text{recall} = \frac{TP}{TP + FN}. \tag{2.28}$$

It reflects how well the positive instances are classified. The false alarm (denoted by *falarm* here, also known as the false positive rate) is the number of false positives compared to the number of negative instances:

$$\text{falarm} = \frac{FP}{TN + FP}. \tag{2.29}$$

It is clear that a balance between the recall and false alarm values should be aimed for. The Receiver Operating Characteristics (ROC, [111]) curve plots the recall against the false alarm for probabilistic classifiers. The Area Under the Curve (AUC) is then the surface under this ROC curve, and expresses how good the balance between recall and false alarm is. The Approximate Area Under the Curve (AAUC):

$$AAUC = \frac{recall - falarm}{2} \tag{2.30}$$

measure can be seen as an approximation of the AUC for discrete classifiers.
The *precision* is the number of true positives compared to the number of instances predicted positive:

$$\text{precision} = \frac{TP}{TP + FP}. \tag{2.31}$$

This measure expresses how many of the instances predicted positive are indeed positive.
Another metric is the *specificity* (also called true negative rate), defined as the number of correctly classified negative instances divided by the number of truly negative instances:

$$\text{specificity} = \frac{TN}{FP + TN}. \tag{2.32}$$

This is the analogue of the recall for the negative instances, and reflects how well negative instances are classified.
All metrics above are defined for binary classification problems, but they can easily be used for multi-class problems. A common practice is to calculate the measure for each class separately and then to average the metrics over all classes (one vs. all).
The most well-known evaluation measure for multiclass problems is the classification accuracy, denoted by *acc*. It is defined as the ratio of correctly classified instances, which can also be expressed as the sum of the diagonal elements in the confusion matrix:

$$\text{acc} = \sum_{i=1}^{k} M_{ii}. \tag{2.33}$$

It is a general measure that gives an idea of the overall performance of the classifier.
Another metric that can handle multi-class problems is Cohen's kappa [12], which is an agreement measure that compensates for classifications that may be due to chance, defined as follows:

$$\kappa = \frac{n \sum_{i=1}^{k} M_{ii} - \sum_{i=1}^{k} M_{i.} M_{.i}}{n^2 - \sum_{i=1}^{k} M_{i.} M_{.i}}, \tag{2.34}$$

where $M_{.i}$ is the sum of the elements in the i^{th} column of M and $M_{i.}$ the sum of the elements in the i^{th} row of M.
There is no simple answer to the question which evaluation metric to use, and in general there is no classifier that is optimal for each evaluation metric. When evaluating general classification problems, the accuracy is mostly sufficient, together with an analysis of Cohen's kappa. In case of imbalanced problems, i.e. data where one or more classes are underrepresented, the accuracy mostly reflects the accuracy of the classification of the majority class. Therefore, the accuracy is less appropriate when evaluating imbalanced problems, and the AAUC is more appropriate, as it takes into account the class imbalance. When considering real-world problems, one should be careful when selecting appropriate evaluation metrics. For instance, when there is a high cost related to classifying instances to the negative class, a high false alarm is problematic. When it is more important to not misclassify positive instances, a high recall is important. In some cases it is recommended to use multiple evaluation metrics, and a balance between them should be aimed for.

2.1.3.3 Non-parametric statistical tests

When evaluating a new classifier, it is important to compare it to the state of the art. Deciding if a new algorithm is better than existing ones is not a trivial task. It can happen that an algorithm is better than another algorithm on average, but this good average performance can be due to some outliers. In general, there is no algorithm that is the best in all situations, as suggested by the *no free lunch* theorem [170]. For these and other reasons, it is crucial to use appropriate statistical tests to verify that a new classifier indeed outperforms the state of the art.

A major distinction between different statistical tests is whether they are parametric [144] or non-parametric [37, 56, 108, 144]. All considered statistical tests, both parametric and non-parametric, assume independent instances. Parametric tests, in contrast to non-parametric tests, are based on an underlying parametric distribution of the (transformed) results of the considered evaluation measure. In order to use parametric statistical tests meaningfully, these distributional assumptions on the observations must be fulfilled.

Depending on the setting, pairwise or multiple comparison tests should be performed. A pairwise test aims to detect a significant difference between two classifiers, while a multiple comparison test aims to detect significant differences between multiple classifiers.

All statistical tests, both parametric and non-parametric, follow the same pattern. They assume a null hypothesis, stating that there is no difference between the classifiers' performance, e.g., expressed by the mean or median difference in performance. Before the test is carried out, a significance level is fixed which is an upper bound for the probability of falsely rejecting the null hypothesis. The statistical test can reject the null-hypothesis at this significance level, which means that with high confidence at least one of the classifiers significantly outperforms the others, or not reject it at this significance level, which means that there is no strong or sufficient evidence to believe that one classifier is better than the others. In the latter case, it is not guarenteed that there are *no* differences between the classifiers, the only conclusion then is that the test cannot find significant differences at this significance level. This might be because there are indeed no differences in performance, or because the power (the probability of correctly rejecting the null hypothesis) of the test is too low, caused by an insufficient amount of datasets.

The decision to reject the null hypothesis is based on whether the observed test statistic is bigger than some critical value or equivalently, whether the corresponding p-value is smaller than the prespecified significance level. Recall that

the p-value returned by a statistical test is the probability that a more extreme observation than the observed one is true.
In the remainder of this section, the number of classifiers is denoted by k, the number of cases by n and the significance level by α. The calculated evaluation measure of case i based on classifier j is denoted Y_{ij}. We only discuss non-parametric statistical tests as the assumptions for parametric statistical tests are in general not fulfilled or too few cases are available to check the validity of these assumptions in our case. That is, we mostly work with about 40 datasets and the results are generally not normally distributed. Strictly speaking, the null hypothesis of the non-parametric tests presented here state that the distribution of the performance of all classifiers is the same. Different tests then differ in what alternatives they aim to detect.
We make a distinction between pairwise and multiple comparisons.

2.1.3.4 Pairwise comparisons: Wilcoxon's signed-ranks test

Wilcoxon's signed-ranks test [167] uses the differences $Y_{i1} - Y_{i2}$. Under the null hypothesis, the distribution of these differences is symmetric around the median and hence we must have that the distribution of the positive differences is the same as the distribution of the negative differences. The Wilcoxon signed rank test aims to detect a deviation from this to reject the null hypothesis. The procedure assigns a rank to each difference according to the absolute value of these differences, where the mean of ranks is assigned to cases with ties. Next, the sum of the ranks of the positive differences R^+ and the sum of the negative differences R^- are calculated. When few instances are available, to reject the null hypothesis, $\min(R^+, R^-)$ should be less than or equal than a critical value depending on the significance level and the number of instances, or equivalently, when the corresponding p-value is lower than the significance level, see [144] for tables. When a sufficient number of instances is available (as a rule of thumb, more than 50 instances), one can rely on the asymptotic approximation of the distribution of R^+ or R^-.

2.1.3.5 Multiple comparisons

In this section we first discuss the Friedman test that detects if there are significant differences among a set of methods, and then explain the Holm post-hoc procedure that tests if the method with the highest rank significantly outperforms others.

Friedman's test Friedman's test [49, 50] ranks the classifiers for each instance, according to the evaluation measure (let r_i^j denote the corresponding rank). Next the average rank R_j for each classifier over the different instances is calculated, $R_j = \sum_{i=1}^{n} r_i^j / n$. The best method gets rank 1, the second best method gets rank 2 and so on. Under the null hypothesis, all classifiers are equivalent and hence the average ranks of the different classifiers should be similar. The Friedman test aims to detect a deviation from this.
For a sufficient number of instances and classifiers (as a rule of thumb $n > 10$ and $k > 5$), the test statistic of the Friedman test approximately follows a chi-square distribution with $k - 1$ degrees of freedom. For a small number of data sets and classifiers, exact critical values have been computed [144, 182]. If the test statistic exceeds the corresponding critical value, it means that there are significant differences between the methods, but no other conclusion whatsoever can be made.

Detecting pairwise differences: Holm's post-hoc procedure When the null hypothesis (stating that all classifiers perform equivalently) is rejected, the average ranks calculated by Friedman's test itself can be used to get a meaningful ranking of which methods perform best. However, post-hoc procedures are still needed to evaluate which pairwise differences are significant. It is tempting to use multiple pairwise comparisons to get more information. However, this will lead to an accumulation of the Type I error coming from the combination of pairwise comparisons, also referred to as the Family Wise Error Rate (FWER, [121]), which is the probability of making at least one false discovery among the different hypotheses.
Therefore we consider post-hoc procedures based on adjusted p-values of the pairwise comparisons to control the FWER. Recall that the p-value returned by a statistical test is the probability that a more extreme observation than the current one is observed, assuming the null hypothesis holds. This simple p-value reflects this probability of one comparison, but does not take into account the remaining comparisons. Adjusted p-values (APV) deal with this problem and after the adjustment, these APVs can be compared with the nominal significance level α. The post-hoc procedures that we discuss first are all designed for multiple comparisons with a control method, that is, we compare one algorithm against the $k - 1$ remaining ones. In the following, p_j denotes the p-value obtained for the jth null hypothesis, stating that the control method and the jth method are performing equally well. The p-values are ordered from smallest to largest: $p_1 \leq \ldots \leq p_{k-1}$, and the corresponding null hypotheses are rewritten accordingly

as $H_1, \ldots, H_{k-1}$.
The Holm [76] procedure is the most popular post-hoc procedure and starts with the lowest p-value. If $p_1 \leq \alpha/(k-1)$, the first null hypothesis is rejected and the next comparison is made. If $p_2 \leq \alpha/(k-2)$, also the second null hypothesis H_2 is rejected and the next null hypothesis is verified. This process continues until a null hypothesis cannot be rejected anymore. In that case, all remaining null hypotheses are retained as well. The adjusted p-values for the Holm procedure are $\min\left[\max\{(k-j)p_j : 1 \leq j \leq i\}, 1\right]$.

2.2 Evolutionary algorithms

In the 1950s, researchers started to study how optimization problems can be modeled and solved inspired by biological evolution. Nowadays, these Evolutionary Algortithms (EAs, [63]) are widespread in research and have been applied successfully in a wide range of domains.
There are many variants of EAs, but they all share the underlying idea that a population of individuals is improved by means of natural selection, founded on the survival of the fittest principle. The general scheme of EAs is depicted in Figure 2.2.1. Candidate solutions to the optimization problem form a population of individuals. The quality of these solutions is measured by means of a fitness function. The higher the value of an individual for the fitness function, the better the individual and the more likely the individual survives and is selected as parent to create the next generation of individuals. This new generation of individuals is formed by applying recombination and mutation operators to the fittest individuals in the previous generation. The newly created individuals compete to survive with the old ones to be included in the next generation. This process continues until a certain termination criterion is met and the fittest individual is returned as solution.
There are two factors that are at the base of EAs' strength. The first is the diversity and novelty among the candidate solutions during the course of the generations, achieved by using good mutation and recombination operators. The second is the strive for good quality candidate solutions, achieved by selection operators that improve the fitness of individuals generation after generation.

Different subgroups of EAs only deviate from the general scheme in Figure 2.2.1 by technical details. An important distinction that needs to be made is the way in which individuals are represented. The subgroup of EAs that we consider are Genetic Algorithms (GAs) where individuals are represented by binary strings.

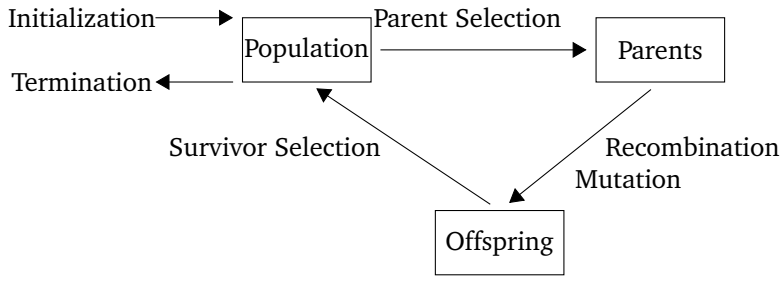


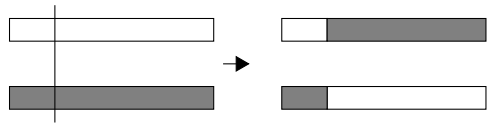
Figure 2.2.1: General scheme of evolutionary algorithms

In the following we discuss the separate components of GAs in more detail.

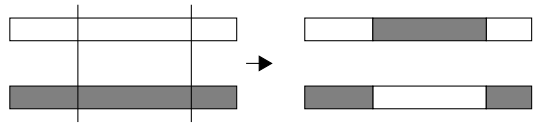
- **Representation**
  Candidate solutions to optimization problems take different forms like sets, integers or matrices. In order to present a solving strategy, it is important that these candidate solutions can be translated to a general form. GAs represent each candidate solution by a binary string. Each position in the binary string is called a *gene* or also a *locus*. The candidate solution in the original optimization problem is called the phenotype, the representation in the GA is called the genotype and the transformation from phenotype to genotype is called *representation*. All candidate solutions in the original optimization problem form the phenospace, while the possible solutions in the GA representation form the genospace.
  For instance, when the goal is to find an optimal subset of a set P, the subsets $S \subseteq P$ can be presented by binary strings of length $|P|$, where each gene corresponds to an element in P, and the gene is 1 if the element is included in S and 0 else. The subset itself is the phenotype, its binary representation is called the genotype. The phenospace consists of all subsets $S \subseteq P$ and the genospace consists of all binary strings of length $|P|$.

- **Fitness function**
  The fitness function reflects the quality of the individuals. This fitness function drives the improvement of the individuals over time. In most cases the fitness function needs the inverse representation function, that is, given the genotype of an individual, the fitness function needs to translate this genotype to the phenotype of the individual, in order to assess its quality.

- **Population**
  The population consists of genotypes of individuals and changes over time: the individuals do not change or adapt but the population does. The initial population mostly consists of randomly generated individuals.
- **Parent selection**
  In each generation parents are selected for recombination. Individuals with a higher fitness are more likely to be selected, but individuals with a low fitness are also given a low chance of being selected. Parent selection is one of the main drives for quality improvement of the population.
- **Crossover**
  Crossover is the process where two parent genotypes are recombined into one or two new genotypes for offspring. There are two random components in crossover. The first is which parents to use for crossover, the second is which parts of these parents to recombine. In Figure 2.2.2 two examples of crossover operators are depicted. The first is one-point crossover, where the first part of the first genotype is attached to the second part of the second genotype, and the first part of the second genotype is attached to the second part of the first genotype. The second is two-point crossover, where a part in the middle of the genotype is interchanged between the two genotypes.
  The principle of crossover is that the features of both parents are combined. It is hoped that the good characteristics of both parents are recombined in the same genotype.
- **Mutation**
  Mutations are unary changes of the individuals: the genotype of the individual undergoes random changes and the resulting mutant is offspring for the next generation. A common mutation is to change the gene of a genotype with a (low) probability. Mutations ensure that the entire genospace can be searched and avoid the GA to get stuck in local optima, but on the other hand the low mutation probability ensures that good solutions are not drastically changed during the course of the GA.
- **Survivor Selection**
  After offspring is produced, it needs to be decided which individuals to keep in the population. Survivor selection chooses the fittest individuals and pushes them to the next generation. Some algorithms only select the fittest individuals from the offspring, while others merge the offspring and the previous population and select the fittest individuals among them. Survivor selection is mostly deterministic, while parent selection has a



(a) Single-point crossover



(b) Two-point crossover

Figure 2.2.2: Two examples of crossover operators

random component.

- **Termination**
  Deciding when to halt the GA can be based on different criteria. One possibility is to require that the fitness of the best individual reaches a certain threshold. In some cases it is possible that this limit is never reached, therefore an additional criterion should be imposed. Another option is to limit the number of fitness evaluations or the number of generations, or to stop the GA when the diversity within the population drops below a certain level.

The procedure of GAs typically consists of two phases: exploration and exploitation. During the exploration phase, the population is very diverse and the entire genospace is explored. The fitness increases very fast in that phase. During the exploitation phase, the GA only searches in specific regions of the genospace in the neighborhood of good individuals.

2.3 Fuzzy Rough Set Theory

We recall the basic concepts of fuzzy rough set theory that are needed for the understanding of the remainder of this work. Fuzzy rough set theory is built upon two other theories, namely fuzzy set theory and rough set theory, discussed in Section 2.3.1 and 2.3.2 respectively. The hybridization of both theories that

we use in this work is presented in Section 2.3.3, and in Section 2.3.4 we relate fuzzy rough set theory to classification problems.

2.3.1 Fuzzy Set Theory

We first recall the most important concepts of fuzzy set theory. We define fuzzy sets, introduce fuzzy logical operators and finally discuss the notion of fuzzy indiscernibility relations.

Fuzzy sets

In 1965, Lotfi Askar Zadeh introduced fuzzy set theory [181]. He observed that traditional crisp sets that are at the basis of logic are not able to describe everything in reality. For instance, it is hard to define the set of *smart people*. One could say that people with an IQ higher than 120 belong to the set of smart people, but this distinction is artificial. Someone with an IQ of 119 can almost not be distinguished from someone with an IQ of 120, and therefore someone with an IQ of 119 should also belong to the set of smart people if someone of 120 does.
Zadeh proposed fuzzy sets to overcome this problem. He defined a fuzzy set A as a mapping from the universe U to the interval $[0, 1]$. The value $A(x)$ for $x \in U$ is called the *membership degree* of x in A. Using this model, elements in the universe can belong to a set to a certain degree. In the example, one could say that someone with an IQ of 120 belongs to degree 0.9 to the set of smart people, and someone with an IQ of 119 to degree 0.89. Finding a good fuzzy set to model concepts can be challenging and subjective, but it is more meaningful than trying to make an artificial crisp distinction between elements.
Note that fuzzy sets are an extension of crisp sets: any crisp set A can be modeled by means of a fuzzy set as follows:

$$\forall x \in U : A(x) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{else.} \end{cases} \tag{2.35}$$

The cardinality of a fuzzy set A is defined as the sum of the membership values of all elements in the universe to A:

$$|A| = \sum_{x \in U} A(x). \tag{2.36}$$

Fuzzy logical operators

Extending crisp sets to fuzzy sets requires new logical operators. For instance, in crisp set theory the proposition *an element belongs to the set A and to the set B* is either true or false. If we want to extend this proposition to fuzzy set theory, we need fuzzy logical operators that extend the logical conjunction $\wedge$, in order to express to what extent an instance x belongs to A and B given the membership degrees $A(x)$ and $B(x)$.
The conjunction $\wedge$ and the disjunction $\vee$ are extended by means of a so-called t-norm $\mathcal{T}$ and t-conorm $\mathcal{S}$ respectively, which are mappings $\mathcal{T}, \mathcal{S} : [0,1]^2 \to [0,1]$ that satisfy the following conditions:

- $\mathcal{T}$ and $\mathcal{S}$ are increasing in both arguments
- $\mathcal{T}$ and $\mathcal{S}$ are commutative
- $\mathcal{T}$ and $\mathcal{S}$ are associative
- $\forall x \in U : \mathcal{T}(x,1) = x$ and $\forall x \in U : \mathcal{S}(x,0) = x$.

The most important examples of t-norms are the minimum operator $\mathcal{T}_M$, which is the largest t-norm, the product operator $\mathcal{T}_P$ and the Łukasiewicz t-norm $\mathcal{T}_L$:

$$\begin{aligned} &\forall x, y \in [0,1] : \mathcal{T}_M(x,y) = \min(x,y) \\ &\forall x, y \in [0,1] : \mathcal{T}_P(x,y) = xy \\ &\forall x, y \in [0,1] : \mathcal{T}_L(x,y) = \max(0, x+y-1). \end{aligned} \tag{2.37}$$

Well-known examples of t-conorms are the maximum operator $\mathcal{S}_M$, which is the smallest t-conorm, the probabilistic sum $\mathcal{S}_P$ and the Łukasiewicz t-conorm $\mathcal{S}_L$:

$$\begin{aligned} &\forall x, y \in [0,1] : \mathcal{S}_M(x,y) = \max(x,y) \\ &\forall x, y \in [0,1] : \mathcal{S}_P(x,y) = x+y-xy \\ &\forall x, y \in [0,1] : \mathcal{S}_L(x,y) = \min(1, x+y). \end{aligned} \tag{2.38}$$

The implication $\to$ is extended by fuzzy implicators, which are mappings $\mathcal{I} : [0,1]^2 \to [0,1]$ that satisfy

- $\mathcal{I}$ is decreasing in the first and increasing in the second argument
- $\mathcal{I}$ satisfies $\mathcal{I}(1,0) = 0$ and $\mathcal{I}(1,1) = \mathcal{I}(0,1) = \mathcal{I}(0,0) = 1$.

The most used implicator is the Łukasiewicz implicator $\mathcal{I}_L$, defined by

$$\forall x, y \in [0,1] : \mathcal{I}_L(x,y) = \min(1, 1-x+y). \tag{2.39}$$

Fuzzy relations

A special type of fuzzy sets are binary fuzzy relations in U, these are fuzzy sets R in U^2 and express to what extent x and y are related to each other. In the context

of fuzzy rough set theory, we use relations to model indiscernibility between instances, therefore we refer to them as indiscernibility relations. We require that R is at least a fuzzy tolerance relation, that is, R is reflexive ($\forall x \in U : R(x,x) = 1$) and symmetric ($\forall x, y \in U : R(x,y) = R(y,x)$). These two conditions correspond to the reflexivity and symmetry conditions of the equivalence relation. The third condition for an equivalence relation, transitivity, is translated to $\mathcal{T}$-transitivity for a certain t-norm $\mathcal{T}$:

$$\forall x, y, z \in U : \mathcal{T}(R(x,y), R(y,z)) \leq R(x,z). \tag{2.40}$$

In this case we call R a $\mathcal{T}$-similarity relation. Note that when R is $\mathcal{T}_M$-transitive, R is $\mathcal{T}$-transitive for all t-norms $\mathcal{T}$. In this case we call R a similarity relation.

2.3.2 Rough Set Theory

Rough set theory was initiated by Zdzisław Pawlak ([124]) in the early eighties to handle the problem of incomplete information. He considered a universe U consisting of elements, an equivalence relation R on U and a concept $A \subseteq U$ within the universe. The problem of incomplete information states that it might not be possible to discern the concept A based on the equivalence relation R, that is, there can exist two elements x and y in U that are equivalent for R but for which x belongs to A and y does not. This is illustrated in Figure 2.3.1: the universe is partitioned in squares using the equivalence relation, and the concept A does not follow the lines of the squares, which means that A cannot be described using R.

This problem of incomplete information occurs often in real life. Consider for instance the problem of spam classification. Assume that the universe consists of spam and non-spam e-mails, and say that the concept is spam. The equivalence relation is defined based on a predefined list of 10 words that occur often in spam e-mail, and we say that two e-mails are equivalent (i.e., indiscernible) if they contain the same words among the list of 10 words. Some equivalence classes will be completely contained in the spam group, and some will be completely contained in the non-spam group. However, it is very likely that there exist two e-mails that contain the same words among the list of 10 words, but for which one is spam and the other non-spam. In this case, the equivalence relation is not able to distinguish between spam and non-spam.

Pawlak addressed this problem by approximating the concept A. The lower approximation contains all the equivalence classes that are contained in A, and the upper approximation contains the equivalence classes for which at least one

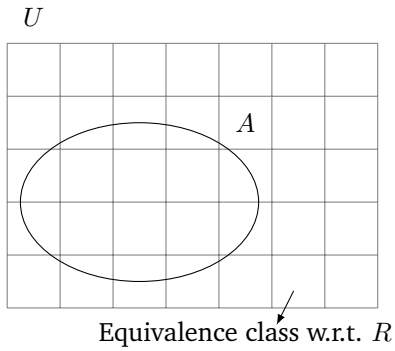


Figure 2.3.1: A universe U partitioned by an equivalence relation R and a concept $A \subseteq U$ that cannot be defined using R.

element is in A, as indicated in Figure 2.3.2.
Formally, the lower approximation of A by means of R is defined as follows:

$$R \downarrow A = \{x \in U | \forall y \in U : (x, y) \in R \rightarrow y \in A\}. \tag{2.41}$$

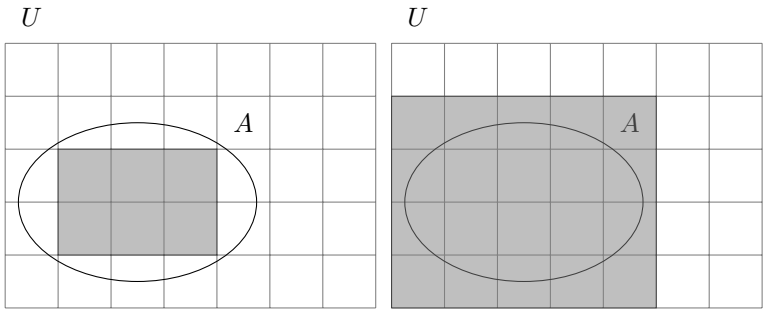
Note that an equivalence relation is reflexive, therefore x is only contained in $R \downarrow A$ if $x \in A$. The upper approximation is defined by:

$$R \uparrow A = \{x \in U | \exists y \in U : (x, y) \in R \wedge y \in A\}. \tag{2.42}$$

The lower approximation in the spam example consists of all e-mails that are spam and for which all e-mails indiscernible from it are also spam. The upper approximation consists of e-mails that are spam and e-mails that are non-spam but for which there exists an e-mail indiscernible from it that is spam.

2.3.3 Fuzzy Rough Set Theory

Fuzzy set theory enables us to model vague information, while rough set theory models incomplete information. These two theories are not competing but complement each other. Many models to hybridize rough sets and fuzzy sets have been proposed [105, 171, 172, 178], but we restrict ourselves to the implicator/t-norm model [29, 45, 46].
A fuzzy rough set is the pair of lower and upper approximations of a fuzzy set A in a universe U on which a fuzzy relation R is defined. The fuzzy rough



(a) The lower approximation (b) The upper approximation

Figure 2.3.2: The concept A is approximated by means of the lower and upper approximation

model is obtained by fuzzifying the definitions of the crisp lower and upper approximation. Recall that the condition for an element $x \in U$ to belong to the crisp lower approximation is that

$$\forall y \in U : (x, y) \in R \rightarrow y \in A. \tag{2.43}$$

The equivalence relation R is now a fuzzy relation and A is a fuzzy set. The values $R(x, y)$ and $A(y)$ are connected by a fuzzy implication $\mathcal{I}$, so $\mathcal{I}(R(x, y), A(y))$ expresses to what extent elements that are similar to x belong to A. The membership value of an element $x \in U$ to the lower approximation is high if these values $\mathcal{I}(R(x, y), A(y))$ are high for all $y \in U$:

$$\forall x \in U : (R \downarrow A)(x) = \min_{y \in U} \mathcal{I}(R(x, y), A(y)). \tag{2.44}$$

The fuzzy lower approximation can be derived similarly, except that now the logical operator $\wedge$ is replaced by a t-norm:

$$\forall x \in U : (R \uparrow A)(x) = \max_{y \in U} \mathcal{T}(R(x, y), A(y)). \tag{2.45}$$

This upper approximation expresses to what extent there exist instances that are similar to x and belong to A.

2.3.4 Using (fuzzy) rough set theory for decision systems

Rough sets and fuzzy rough sets are particularly useful to model a decision system $(U, \mathcal{A} \cup \{d\})$. The universe U corresponds to the set of instances. The indiscernibility relation R can be modeled by means of the conditional attributes $\mathcal{A}$: instances x and y for which the conditional attributes are highly related will have a high membership degree $R(x, y)$ or will be equivalent for R. The concepts to be approximated correspond to the decision classes that are derived from the decision attribute.

We first show how rough set theory can be deployed to approximate the decision classes in the decision system. Assume that we can derive an equivalence relation R from the attributes that partitions the universe U, and that the instances are divided in three classes as depicted in Figure 2.3.3. The lower approximations of the classes, indicated in gray in Figure 2.3.3 are particularly interesting: the instances in these regions are typical for their classes: based on the equivalence relation R, and hence also based on the attributes in $\mathcal{A}$, the classes of those instances can unambiguously be determined. Formally, the union of the lower approximations of all classes is called the positive region, defined as follows:

$$POS = \bigcup_{x \in U} R \downarrow [x]_d, \tag{2.46}$$

where $[x]_d = \{y \in U | d(x) = d(y)\}$ is the decision class of x.

The positive region can easily be extended to the fuzzy case:

$$\forall x \in U : POS(x) = \max_{y \in U} (R \downarrow [y]_d)(x). \tag{2.47}$$

As $[y]_d(x)$ only takes values in $\{0, 1\}$ for classification problems and $\mathcal{I}(a, 0) = 0$ for all $a \in [0, 1]$, the positive region can also be written as follows [30]:

$$\forall x \in U : POS(x) = (R \downarrow [x]_d)(x), \tag{2.48}$$

and can be interpreted as the extent to which instances indiscernible from x belong to the class of x.

This fuzzy rough positive region is very useful in many aspects of machine learning. For instance, fuzzy rough feature selection techniques select features such that the membership degrees of the instances to the fuzzy rough positive region are maximized [30], or instance selection techniques select instances with a high membership degree to the fuzzy rough positive region [80].

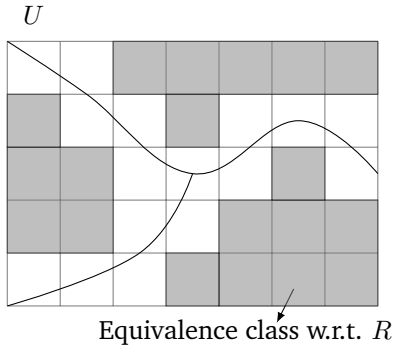


Figure 2.3.3: Decision system where the universe, consisting of three classes, is partitioned by an equivalence relation R. The positive region is indicated in gray

2.4 Instance Selection

Data preprocessing is an important step in data mining. Real-world datasets can be noisy, huge, have missing values, have irrelevant features, contain superfluous instances, and so on. Instance selection deals with some of these issues. It is the process where a subset of instances is selected before classification takes place. We distinguish three types of instance selection methods based on their purpose. Instance selection methods that focus on improving the accuracy of the classifier that is applied afterwards are called edition techniques. Methods whose main goal is to reduce the storage requirements are called condensation algorithms. Some instance selection methods achieve both goals simultaneously, they are called hybrid methods.
Another distinction that should be made is between wrapper algorithms and filter algorithms. Wrapper instance selection algorithms use a classifier to evaluate candidate subsets during the instance selection process, while filter algorithms do not. Wrapper algorithms are usually slower but more accurate than filter techniques [55].
In this section we present the state-of-the-art of instance selection techniques. We limit ourselves to instance selection techniques that were designed for KNN or SVM classification. In Section 2.4.1, we discuss instance selection techniques that were designed as preprocessing techniques for KNN classification, and we refer to them as Prototype Selection (PS, [55]) techniques. In Section 2.4.2, we discuss two techniques that were developed as preprocessors for SVM classification,

referred to as Training Set Selection (TSS) algorithms. Note that filter PS techniques can be used in combination with SVMs, but we describe them in a separate section as they were designed for KNN specifically. An overview of all techniques is given in Figure 2.4.1.

2.4.1 Prototype Selection

A lot of research has been done on PS, and as a result many methods have been proposed in the literature. In the following we discuss the state-of-the-art in PS algorithms. We always denote the original training data by U and the selected subset of prototypes by S.

2.4.1.1 Filter PS techniques

The first large group of methods are methods that are all variations and improved versions of two basic methods, Edited Nearest Neighbors (ENN) and Condensed Nearest Neighbors (CNN).

- **Edited Nearest Neighbor (ENN, [168])**
  The ENN method considers all instances in U and marks $x \in U$ if x is wrongly classified when applying the KNN rule with $U \setminus \{x\}$ as pool of possible nearest neighbors. After all instances are considered, the marked instances are removed.
- **Condensed Nearest Neighbor (CNN, [72])**
  The CNN method initializes S with one element of each class, and then proceeds by classifying all instances in U using KNN with S as pool of possible nearest neighbors. Each time an instance $x \in U$ is classified incorrectly, it is added to S. The process is repeated until all instances in U are classified correctly using S as pool of possible nearest neighbors or until $S = U$.

CNN only selects the instances that are necessary to classify the instances in U correctly, which means that it mainly removes superfluous instances, that is, CNN is a condensation method. On the other hand, ENN removes those instances that deteriorate the classification of instances in U, so ENN is an editing method. Both methods are simple basic methods that have been improved in several ways. We first discuss PS methods that aim to improve ENN.

- **All-K Nearest Neighbor (All-KNN, [154])**
  ENN depends on K, the number of neighbors that KNN looks up to classify the instances in U. The All-KNN PS method depends on a parameter K_{max}

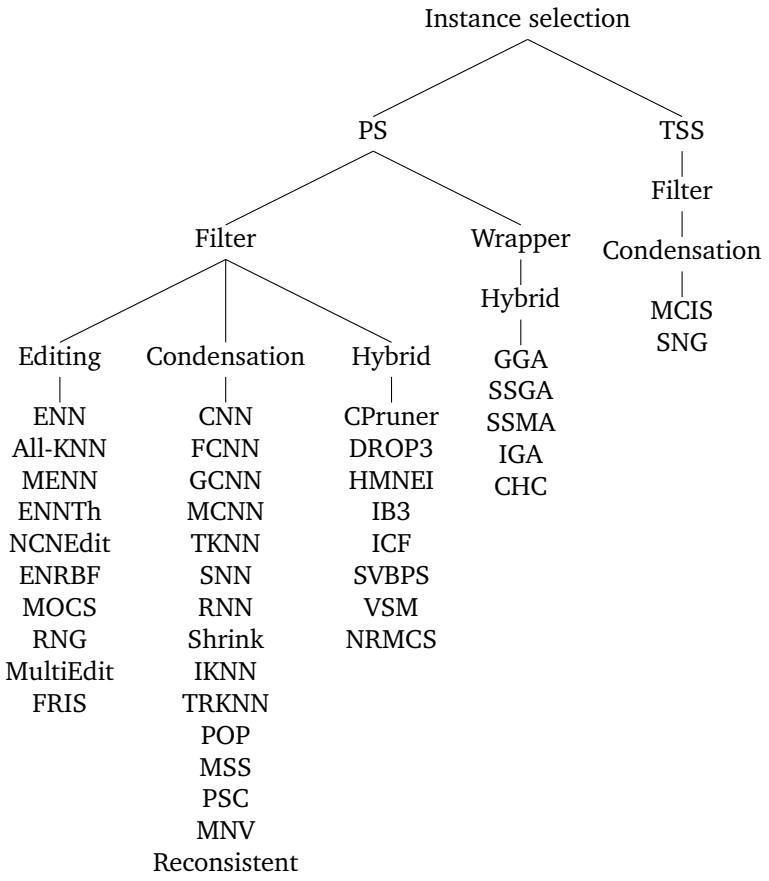


Figure 2.4.1: Overview of the state-of-the-art instance selection methods for KNN (PS) and for SVM (TSS) classification.

and proceeds as follows; it carries out ENN with all $K = 1, \ldots, K_{max}$ and removes the instances that were removed by at least one of those ENN runs. As a result, All-KNN always removes more instances than ENN.

- **Modified Edited Nearest Neighbor (MENN, [74])**
  The MENN PS method only slightly modifies ENN. During the ENN process, instead of considering the K nearest neighbors, $K + l$ neighbors are considered, where l is the number of instances that are at the same distance of the furthest nearest neighbor.

The second group of PS methods are based on the ideas of CNN and have in common that they start with an empty prototype set S and add instances if they are necessary for the classification of instances in U.

- **Fast Condensed Nearest Neighbor algorithm (FCNN, [5])**
  The FCNN algorithm initializes S with the centroid instances of all classes, these are the instances that are closest to the geometric mean of the instances of each class. For each instance $x \in S$, the Voronoi cell Vor(x) consists of the instances in U that are closer to x than to any other instance. Among the instances in $Vor(x)$ that are from a different class than x, the closest one is added to S. This process is repeated for all instances in S until no instances are removed in an iteration.

- **Modified Condensed Nearest Neighbor (MCNN, [39])**
  The MCNN algorithm proceeds as CNN, but instead of adding an instance to S when it is misclassified, it is marked. After classifying all instances in U, the instances closest to the centroids of each class among the misclassified instances are added to S. This process is repeated until there are no misclassifications anymore.

- **Reduced Nearest Neighbor (RNN, [62])**
  RNN is a decremental version of the CNN algorithm. It starts with $S = U$ and removes each instance x from S if such a removal does not cause any other instances in U to be misclassified by the instances remaining in S using the 1NN rule.

Apart from these algorithms that are closely related to ENN or CNN, there are other algorithms that use the KNN rule in the course of the algorithm:

- **Combined-Pruner (C-Pruner, [187])**
  The C-Pruner algorithm defines the coverage set of an instance x as the instances that have the same class as x and for which x is one of the nearest neighbors. C-Pruner considers three types of instances. An instance is superfluous if it is classified correctly by its nearest neighbors. A critical instance is an instance that has at least one instance in its coverage set

that is not superfluous or that has at least one instance in its coverage set that is not superfluous if the instance itself would be deleted. A noisy instance is an instance that is not superfluous and that has more instances in its nearest neighbor set than in its coverage. C-pruner deletes noisy instances and superfluous instances that are not critical. The order of removal is important. An instance x is removed earlier than an instance y if the number of nearest neighbors of x that have the same class as x is larger than the number of nearest neighbors of y that have the same class as y. Ties can occur, in that case the instance for which the distance to the nearest instance of a different class is smaller should be removed first. If ties persist, a random selection is made.

- **Decremental Reduction Optimization Procedure (DROP3, [169])**
  There are many versions of the DROP algorithm, we consider the DROP3 algorithm. It first carries out ENN in order to remove noise from the data. Next, it removes an instance x if instances that have x as nearest neighbor and that are classified correctly by their nearest neighbors, are also classified correctly when removing x. Instances that are closest to an instance from a different class are removed first.

- **Model Class Selection, (MOCS, [18])**
  MOCS marks an instance for removal if the number of times it is one of the nearest neighbors of another instance y and its class equals the class of that instance y is lower than the number of times it is one of the nearest neighbors of another instance y, and its class is different from the class of that instance y.

- **Instance-Based Selection (IB3, [2])**
  The IB3 algorithm is the best-performing PS algorithm among a range of instance based algorithms. It gradually adds instances to the prototype subset S and updates the classification performance of the instances each time an element is added. This classification performance of an element $y \in S$ is the number of times y contributes to a correct classification. When S is the current subset and a new element x is added, the nearest neighbors of x in $S \setminus \{x\}$ are calculated. If x is classified correctly by these nearest neighbors, the number of times these neighbors contribute to a correct classification is increased by one. Note that the classification performance of an element $x \in S$ only depends on instances that are added to S later than x. As a result it is a measure to assess how well new instances will be classified by x. An instance in S is called acceptable if the number of times it contributes to a correct classification is significantly higher than its class

frequency and unacceptable if this number is significantly lower than its class frequency. The overall algorithm follows the next steps. The prototype subset S is initialized with a random element. Then, for each element $x \in U$, the following procedure is carried out. The element y in S that is most similar to x among the acceptable instances in S is selected. If there are no acceptable instances in S, a random element is chosen. If the classes of x and y are different, y is added to S, and the classification performance of the instances in S is updated. Instances that become unacceptable are removed from S.

- **Iterative Case Filtering (ICF, [17])** The ICF algorithm defines the reachability set of an element x as the instances that are closer to x than any instance of a different class of x. The coverage set of x consists of instances for which x is in their reachability set. Instances in the reachability set of x contribute to the correct classification, while x contributes to the correct classification of instances in the coverage set of x. ICF starts by applying ENN on the training set, the resulting set is S. Next, the coverage and reachability set are calculated for each instance in S. Instances for which the reachability set is larger than the coverage set are marked for removal, and all marked instances are removed. This process, excluding ENN, is repeated until no changes occur in S.
- **Modified Selective Subset (MSS, [8])** MSS calculates for each instance the distance to the closest instance from a different class. Next, all instances are considered one by one, instances with a smaller distance are considered first. An instance x is added to the prototype subset if there exists an instance in $U \setminus S$ that is closer to x than to the closest instance from a different class.
- **Reconsistent, [107]**
  The Reconsistent algorithm constructs two sets M and N that are merged to S at the end of the algorithm. Both sets are constructed based on the definition of a special neighborhood for each instance x, that is obtained by first adding the nearest neighbor of x, and then adding instances such that the centroid of the current neighborhood is closest to x. This is repeated until a neighbor from a different class would be added. The set N is initialized as U and for each instance in S the neighborhood is determined. Then, the instance $x \in S$ determined with the largest neigborhood. Instances in this neighborhood are removed from S, and if these instances are also included in neighborhoods of other instances they are removed from these neighborhoods. This is repeated until all

neighborhoods are empty. The set M is initialized as the empty set, and classifies all instances in U using M as pool of candidate nearest neighbors. If an instance is classified incorrectly, it is added to M. Then M is updated in the same way as S.

The last algorithms that we discuss are not based on the KNN rule. The following three algorithms use graphs to model the decision system:

- **Hit Miss Network Edition Iterative (HMNEI, [112])**
  HMNEI builds a *hit and miss network*, which is a graph where the nodes are the instances in U and where the directed edges connect an instance with its nearest neighbor from each class. For an instance x, the incoming edges from instances of the same class are hits, while incoming edges from instances of a different class are misses. HMNEI marks instances for removal if they have more misses than hits, as these instances are on the decision borders. In order to make sure that the dataset does not become too imbalanced, instances are only removed if there are enough instances of that class remaining. The procedure is repeated until no changes occur in an iteration.

- **Relative Neighborhood Graph Editing (RNG, [141])**
  The RNG algorithm calculates the neighborhood graph as follows. Each instance is a node, and two instances x and y are connected by an edge if for any other instance z, the distance between x and y is smaller than either the distance between x and z or the distance between y and z. For each instance the majority class among the neighbors in the graph is calculated. Instances for which this majority class is different from their own class are removed.

The following algorithm considers all features separately and removes points that are internal for all features:

- **Patterns by Ordered Projection (POP, [134])** The goal of POP is to preserve the decision boundaries. POP determines for each instance $x \in U$ its weakness. For each attribute $a \in A$ the weakness of x can be raised by one. Only if the weakness of an instance equals the total number of attributes $|A|$, it is not included in the instance subset that POP returns.
  Determining if the weakness of an instance $x \in U$ should be raised for an attribute $a \in A$ depends on the nature of a. First the numerical attributes are processed: if the instance is an internal point for a numerical attribute its weakness is raised by one. Next, the categorical attributes are handled. The POP algorithm wants to return a representative set and therefore it should return at least one example of every different categorical attribute.

For each value of each categorical attribute, the weakness of all instances except one is raised, namely an instance for which the weakness is minimal among all instances that have this specific value for the considered attribute. These are instances that are likely to be selected based on their numerical attributes, and in this way it is ensured that for each attribute value there will be an instance with that attribute value.

And finally, the next method is based on fuzzy rough set theory:

- **Fuzzy Rough Instance Selection (FRIS, [80])**
  The FRIS algorithm removes instances for which the membership to the fuzzy rough positive region is lower than a certain threshold.

2.4.1.2 Wrapper PS algorithms

An important class of PS techniques are GAs. The individuals are candidate subsets of prototypes, represented by binary strings of length $|U|$. A gene is 1 if the instance corresponding to it is included in the candidate subset and 0 else. The fitness of individuals is based on two important components. The first is the leave-one-out accuracy acc_S, this is the accuracy that is obtained when classifying instances $x \in U$ using the set S as pool of nearest neighbors if $x \notin S$ and $S \setminus \{x\}$ else. The second component is the reduction red_S, which is the percentage of removed instances. These two components are balanced using a factor $\alpha \in [0, 1]$:

$$\text{fitness}(S) = \alpha \text{acc}_S + (1 - \alpha)\text{red}_S. \tag{2.49}$$

We consider five genetic PS algorithms that share the same fitness function and representation. All algorithms halt when a certain number of fitness evaluations has been carried out.

- **Generational Genetic Algorithm (GGA, [97, 98])**
  The GGA algorithm follows the general scheme of evolutionary algorithms. The population is initialized randomly. Parent selection happens stochastically, that is, individuals with a higher fitness have a higher chance of being selected, but also individuals with a low fitness value can be selected as parent. Once the parents are selected, parents are matched randomly and offspring is generated using two-point crossover. Mutation only happens with a small probability, and the probability of a 0 to 1 mutation is smaller than the probability of a 1 to 0 mutation in order to force the algorithm to obtain higher reduction rates. Survivor selection is done by selecting the entire generated offspring and adding the fittest individual from the previous population, this is also referred to as elitism.

- **Steady State Genetic Algorithm (SSGA, [21])**
  The SSGA algorithm differs from the GGA algorithm by its selection operators. In each generation, only two parents are chosen to generate offspring. After the offspring is generated using crossover and mutation, the worst individuals in the population are replaced by the two new individuals.
- **Intelligent Genetic Algorithm for Edition (IGA, [75])**
  The IGA algorithm uses the same parent selection, survivor selection and mutation operators as the GGA algorithm. The only point where IGA and GGA differ is the crossover operator. IGA uses intelligent crossover, where for each position in the genotype, it is decided if the gene from the first of the gene from the second parent should be selected for the offspring. All combinations are considered and the best one is selected.
- **Steady State Memetic Algorithm (SSMA, [54])**
  The SSMA algorithm is similar to the SSGA algorithm, the difference is that before replacing the two worst individuals an optimization step is carried out. This optimization happens if the fitness of the offspring is higher than the fitness of the individual in the current population with the lowest fitness. Else, the optimization only happens with a low probability. The optimization is achieved by changing one 1 in the genotype to a 0. The position with the highest gain in accuracy is chosen.
- **Adaptive Search for Instance Selection (CHC, [21])**
  The CHC algorithm selects all parents for recombination. The parents are randomly paired, and only parents that differ enough can produce offspring. This incest prevention takes place by measuring the hamming distance between the parents. If this distance is smaller than a certain threshold, no crossover takes place. If there are nearly no parent pairs left, the threshold can be decreased. The old population and the offspring are merged and the fittest individuals survive. No mutation takes place, but the population is re-initialized if the population converges. This initialization is done by considering the fittest individual found so far and changing a fixed percentage of randomly selected genes from 1 to 0.

The following two optimization algorithms are closely related to genetic PS algorithms.

- **Random Mutation Hill-Climbing (RMHC, [146])**
  The RMHC algorithm starts off with a subset S containing a fixed percentage of instances of U (typically 90%). For a predefined number of iterations, the RMHC algorithm removes a random element from S and adds a random element from $U \setminus S$ to it. If the leave-one-out-accuracy

acc_S decreases, the previous solution is recovered, otherwise the algorithm continues with the new solution.

- **Encoding Length Explore (Explore, [20])**
  The explore algorithm defines a cost function based on the leave-one-out-accuracy and the number of instances selected, but differs from the fitness function in the way these components are aggregated. The Explore algorithm initializes S with one element. Then for each element in $x \in U$, x is added to S if the cost of S decreases. After repeating this for all instances, all instances in $y \in S$ are processed one by one. If removing y from S decreases the cost, the element is removed.

An extensive experimental evaluation [55] has shown that the wrapper approaches are amongst the most accurate PS methods. Especially GGA, CHC, SSGA and RMHC have high accuracy rates and at the same time remove about 90 percent of the instances. The RNG filter method performs very well and is faster than the wrapper techniques but removes less instances. The condensation methods are in general not able to maintain the performance of KNN classification, but some of them remove up to 90 percent of the instances.

2.4.2 Training Set Selection

Not much research has been done on TSS for SVMs, only two methods have been proposed specifically for SVMs in the literature so far.

- **Multi-Class Instance Selection (MCIS, [27])**
  The first technique is MCIS, which can only be used in a one-versus-all setting. When the number of classes is k, the one-versus-all scheme considers k problems, where the i-th problem considers the i-the class as positive and the remaining classes negative. For each of these problems, a subset of instances S is selected, and the SVM is trained on S instead of on the entire training set. The MCIS algorithm clusters only the positive class and then removes instances of the positive class that are close to the centers of the clusters and selects instances of the negative class that are closest to the centers of the clusters. In this way, instances near the boundary between the positive and negative class are selected.
- **Sparsifying Neural Gas (SNG, [183])**
  Another technique developed to improve SVMs is the SNG algorithm, which is restricted to two-class problems. The intention of SNG is to only select instances that will likely become support vectors in the final SVM classification. To this goal, a combination of learning vector quantization

techniques and the growing neural gas algorithm [51] is used.

3. Ordered Weighted Average Fuzzy Rough Sets

In many applications, the datasets at hand can contain attribute or class noise [190]. Attribute noise can be erroneous attribute values, missing or incomplete values, while class noise can come from contradictory examples, where two instances have the same attribute values but a different class label, and misclassifications where the instances are incorrectly annotated.
Unfortunately, the traditional fuzzy rough set model is not suited to handle this noise, as the fuzzy rough lower and upper approximations are based on the strict minimum and maximum operators. As a result, small changes in the dataset can lead to drastic changes in the values of the fuzzy rough lower and upper approximation.
Robust fuzzy rough set models extend the fuzzy rough set model and aim to overcome this problem. A straightforward way to make the fuzzy rough set model more robust is to elaborate on the strict minimum and maximum operators that cause the robustness problem. The β precision fuzzy rough set model [78, 139, 140] restricts the minimum and maximum operators to the least extreme values using a cutoff parameter β. A similar approach is taken in [115, 116], where variable precision fuzzy rough sets are introduced. Again, the set over which the minimum or maximum is taken is restricted, and additionally the fuzzy implication and t-norm are replaced by inclusion measures. The fuzzy variable precision model [188] does not change the minimum and maximum operators but cuts of the membership values of the instances to the concept in order to omit extreme values. Vaguely quantified fuzzy rough sets [28] use vague quantifiers to express that an element belongs to the lower (resp. upper) approximation if *most* (resp. *some*) instances related to it belong to the concept. Theoretical properties that the traditional fuzzy rough set model satisfy are not always satisfied by the robust models. Moreover, some of the aforementioned

robust models still exhibit some crispness using cut-off values, require parameter tuning or do not embody the concept of fuzzy rough set theory.
This motivates us to introduce a new robust fuzzy rough set model, called Ordered Weighted Average (OWA) fuzzy rough sets [31]. We solely elaborate on softening the minimum and maximum operators used in the traditional fuzzy rough set model and as such, we preserve the fundamental ideas of traditional fuzzy rough set theory.
We give the definition and discuss the properties of OWA fuzzy rough sets in Section 3.1. To verify if our new fuzzy rough set model is more robust than the traditional model, we design and execute an experimental evaluation to assess the robustness in Section 3.2.

3.1 Definition and properties

We work in a universe U on which an indiscernibility relation R is defined. Recall that the fuzzy rough lower and upper approximation of a fuzzy set A are defined as follows:

$$\begin{aligned} \forall x \in U : (R \downarrow A)(x) &= \min_{y \in U} \mathcal{I}(R(x,y), A(y)) \\ \forall x \in U : (R \uparrow A)(x) &= \max_{y \in U} \mathcal{T}(R(x,y), A(y)), \end{aligned} \tag{3.1}$$

with $\mathcal{I}$ an implicator and $\mathcal{T}$ a t-norm. As these definitions are based on the strict minimum and maximum operators, small changes in the values of R or A can alter the outcome of the lower and upper approximation drastically.
A straightforward way to improve the definitions of the fuzzy rough lower and upper approximation is to replace the strict minimum and maximum operators that are at the source of the problem. The so-called Ordered Weighted Average (OWA, [176]) aggregator is very well suited to this goal. It is similar to the weighted average aggregator, except that the weights are not associated with the values but with the ordered positions of the values.
Assume that the values $V = \{v_1, \ldots, v_p\}$ need to be aggregated, and that a weight vector $W = \langle w_1, \ldots, w_p \rangle$ for which $\sum_{i=1}^{p} w_i = 1$ and for all $i \in \{1, \ldots, p\}$, $w_i \in [0,1]$ holds is provided. If for all i in $\{1, \ldots, p\}$ c_i is the i-th largest value in V, then the OWA_W aggregation of the values in V is given by:

$$\text{OWA}_W(V) = \sum_{i=1}^{p} (w_i c_i),$$

that is, the values in V are ordered decreasingly, and the weights W are associated to the values of V in that order.
The OWA's main strength is its flexibility, since it enables us to model a wide range of aggregation strategies. For example, when $W = \langle 0, \ldots, 0, 1\rangle$ is used as the weight vector, OWA_W corresponds to the maximum, and when the weight vector equals $W = \langle 1/p, \ldots, 1/p\rangle$, the average is retrieved.
The strict minimum and maximum operators in the fuzzy rough set model can be replaced by OWA operators that behave like the minimum and maximum operator. The weight vector $W_{\min}$ used to model the minimum should associate small weights to high values and large weights to small values. As such, the $\text{OWA}_{W_{\min}}$ aggregator is a soft or gradual extension of the strict minimum. The opposite holds for the aggregator $\text{OWA}_{W_{\max}}$ that models the maximum. The vector W_{max} should associate high values with high weights and low values with low weights.
The weight vectors associated with the OWA operator can be analyzed by means of the *orness* and *andness* degrees:

$$\text{orness}(\langle w_1, \ldots, w_p\rangle) = \frac{1}{(p-1)} \sum_{i=1}^{p} (p-i) w_i \tag{3.2}$$

$$\text{andness}(\langle w_1, \ldots, w_p\rangle) = 1 - \text{orness}. \tag{3.3}$$

An OWA operator for which the weight vector has a high orness degree resembles the maximum operator, while OWA operators for which the weight vectors have a high andness degree behave like the minimum operator. It holds that $\text{orness}(\langle 1, 0, \ldots, 0\rangle) = 1$ and $\text{andness}(\langle 0, \ldots, 0, 1\rangle) = 1$. To ensure that $\text{OWA}_{W_{\max}}$ behaves like the maximum operator, we can require that the orness degree of $W_{\max}$ is higher than 0.5, and similarly, we can require that the andness degree of $W_{\min}$ is higher than 0.5.
We give three examples of such weight vectors. The first weight vectors are *additive* and are defined as follows:

$$W_{min}^{add} = \langle \frac{2}{p(p+1)}, \frac{4}{p(p+1)}, \ldots, \frac{2p}{p(p+1)}\rangle \tag{3.4}$$

$$W_{max}^{add} = \langle \frac{2p}{p(p+1)}, \frac{2(p-1)}{p(p+1)}, \ldots, \frac{2}{p(p+1)}\rangle. \tag{3.5}$$

Note that these weights are normalized versions of the weight vectors $\langle 1, 2, \ldots, p\rangle$ and $\langle p, p-1, \ldots, 1\rangle$ respectively.

The second set of OWA weight vectors are *inverse* additive weights:

$$W_{min}^{inv} = \langle \frac{1}{p\sum_{i=1}^{p}\frac{1}{i}}, \frac{1}{(p-1)\sum_{i=1}^{p}\frac{1}{i}}, \ldots, \frac{1}{1\sum_{i=1}^{p}\frac{1}{i}} \rangle \tag{3.6}$$

$$W_{max}^{inv} = \langle \frac{1}{1\sum_{i=1}^{p}\frac{1}{i}}, \frac{1}{2\sum_{i=1}^{p}\frac{1}{i}}, \ldots, \frac{1}{p\sum_{i=1}^{p}\frac{1}{i}} \rangle. \tag{3.7}$$

The last set of OWA weight vectors are *exponential*:

$$W_{min}^{exp} = \langle \frac{1}{2^p\sum_{i=1}^{p}\frac{1}{2^i}}, \frac{1}{2^{p-1}\sum_{i=1}^{p}\frac{1}{2^i}}, \ldots, \frac{1}{2\sum_{i=1}^{p}\frac{1}{2^i}} \rangle \tag{3.8}$$

$$W_{max}^{exp} = \langle \frac{1}{2\sum_{i=1}^{p}\frac{1}{2^i}}, \frac{1}{2^2\sum_{i=1}^{p}\frac{1}{2^i}}, \ldots, \frac{1}{2^p\sum_{i=1}^{p}\frac{1}{2^i}} \rangle. \tag{3.9}$$

These weights change faster than the additive weights or inverse weights and the corresponding OWA fuzzy rough model resembles more the traditional fuzzy rough model.
In the following theorems we show that the orness degrees of the three weight vectors that soften the maximum defined above are higher than 0.5.

Theorem 3.1.1. *orness*$(W_{max}^{add}) \geq 0.5$

Proof.

$$\begin{aligned}
\text{orness}(W_{max}^{add}) &= \tfrac{1}{p-1}\sum_{i=1}^{p}(p-i)\tfrac{2(p-i+1)}{p(p+1)} \\
&= \tfrac{2}{(p-1)p(p+1)}\sum_{i=1}^{p}(p-i)(p-i+1) \\
&= \tfrac{2}{(p-1)p(p+1)}\sum_{i=1}^{p}((p^2+p)-i(2p+1)+i^2) \\
&= \tfrac{2p(p^2+p)}{(p-1)p(p+1)} - \tfrac{2(2p+1)p(p+1)}{(p-1)p(p+1)2} - \tfrac{2p(p+1)(2p+1)}{(p-1)p(p+1)6} \\
&= 2/3 \\
&\geq 0.5
\end{aligned}$$

□

Theorem 3.1.2. *orness*$(W_{max}^{exp}) \geq 0.5$ *for* $p \geq 3$.

Proof. We prove this theorem using induction. Note that the condition orness$(W_{max}^{exp}) \geq 0.5$ is equivalent with:

$$\sum_{i=1}^{p}(p-i)\frac{1}{2^i} \geq (p-1)0.5\sum_{i=1}^{p}\frac{1}{2^i}. \tag{3.10}$$

First, we verify that this equation holds for $p = 3$:

$$(2\frac{1}{2} + 1\frac{1}{4} + 0\frac{1}{8}) = \frac{5}{4} \geq (\frac{1}{2} + \frac{1}{4} + \frac{1}{8}) = \frac{7}{8}.$$

Next, we assume that the theorem holds for p and prove that it holds for $p+1$. The left-hand-side of Equation (3.10) can be rewritten as follows for $p+1$:

$$\begin{aligned}\sum_{i=1}^{p+1}(p+1-i)\tfrac{1}{2^i} &= \sum_{i=1}^{p}(p+1-i)\tfrac{1}{2^i} + (p+1-(p+1))\tfrac{1}{2^i} \\ &= \sum_{i=1}^{p}(p-i)\tfrac{1}{2^i} + \sum_{i=1}^{p}\tfrac{1}{2^i}.\end{aligned}$$

We can now use the assumption that Equation (3.10) holds for p:

$$\begin{aligned}\sum_{i=1}^{p+1}(p+1-i)\tfrac{1}{2^i} &\geq 0.5(p-1)\sum_{i=1}^{p}\tfrac{1}{2^i} + \sum_{i=1}^{p}\tfrac{1}{2^i} \\ &= 0.5p\sum_{i=1}^{p}\tfrac{1}{2^i} + 0.5\sum_{i=1}^{p}\tfrac{1}{2^i} \\ &\geq 0.5p\sum_{i=1}^{p}\tfrac{1}{2^i} + 0.5p\tfrac{1}{2^{p+1}} \\ &= 0.5p\sum_{i=1}^{p+1}\tfrac{1}{2^i}.\end{aligned}$$

□

Theorem 3.1.3. *orness*$(W_{max}^{inv}) \geq 0.5$ *for* $p \geq 3$.

Proof. We prove this theorem using induction again. The condition orness$(W_{max}^{inv}) \geq 0.5$ is equivalent with:

$$\sum_{i=1}^{p}(p-i)\frac{1}{i} \geq (p-1)0.5\sum_{i=1}^{p}\frac{1}{i}. \tag{3.11}$$

First, we verify that this equation holds for $p = 3$:

$$(2\frac{1}{1} + 1\frac{1}{2} + 0\frac{1}{3}) = 5 \geq (\frac{1}{1} + \frac{1}{2} + \frac{1}{3}) = \frac{11}{6}.$$

Next, we assume that the theorem holds for p and prove that it holds for $p+1$. The left-hand-side of Equation (3.11) can be rewritten as follows for $p+1$:

$$\begin{aligned}\sum_{i=1}^{p+1}(p+1-i)\tfrac{1}{i} &= \sum_{i=1}^{p}(p+1-i)\tfrac{1}{i} + (p+1-(p+1))\tfrac{1}{i} \\ &= \sum_{i=1}^{p}(p-i)\tfrac{1}{i} + \sum_{i=1}^{p}\tfrac{1}{i}.\end{aligned}$$

We can now use the assumption that Equation (3.11) holds for p:

$$\begin{aligned}\sum_{i=1}^{p+1}(p+1-i)\tfrac{1}{i} &\geq 0.5(p-1)\sum_{i=1}^{p}\tfrac{1}{i} + \sum_{i=1}^{p}\tfrac{1}{i} \\ &= 0.5p\sum_{i=1}^{p}\tfrac{1}{i} + 0.5\sum_{i=1}^{p}\tfrac{1}{i} \\ &\geq 0.5p\sum_{i=1}^{p}\tfrac{1}{i} + 0.5p\tfrac{1}{p+1} \\ &= 0.5p\sum_{i=1}^{p+1}\tfrac{1}{i}.\end{aligned}$$

□

As the weight vectors that model the minimum are the opposite of the weight vectors that model the maximum, it immediately follows that the andness of these minimum weight vectors is more than 0.5.
The OWA fuzzy rough model uses these weights to replace the minimum and maximum operators:

$$\begin{aligned}\forall x \in U : (R \downarrow^{\mathsf{OWA}} A)(x) &= \underset{y\in U}{\mathsf{OWA}_{W_{min}}}\mathcal{I}(R(x,y), A(y)) \\ \forall x \in U : (R \uparrow^{\mathsf{OWA}} A)(x) &= \underset{y\in U}{\mathsf{OWA}_{W_{max}}}\mathcal{T}(R(x,y), A(y)),\end{aligned} \tag{3.12}$$

where W_{min} is a vector of weights that models a soft minimum aggregator, and W_{max} a vector of weights that models a soft maximum aggregator.
In the next example, we illustrate how more values are taken into account when calculating the OWA lower and upper approximation.

Example 3.1. *Consider a universe U consisting of 101 instances*

$$U = \{y_1, \ldots, y_{100}, x\}$$

and let A be a fuzzy set in U such that for all $i = 1, \ldots, 100$ it holds that $A(y_i) = i/100$ and $A(x) = 1$. Let R be a symmetric indiscernibility relation such that for all $i = 1, \ldots, 100$ it holds that $R(y_i, x) = i/100$ and $R(x, x) = 1$. Then we have the following observations for the traditional fuzzy rough set model:

- $(R \downarrow A)(x) = 1$*, but if* $A(y_{100}) = 0$*,* $(R \downarrow A)(x) = 0$
- $(R \downarrow A)(x) = 1$*, but if* $R(y_1, x) = 1$*,* $(R \downarrow A)(x) = 0.01$

If we use additive weights, the following observations hold for the OWA fuzzy rough set model:

- $(R \downarrow^{OWA} A)(x) = 1$*, but if* $A(y_{100}) = 0$*,* $(R \downarrow A)(x) = 0.9902$
- $(R \downarrow^{OWA} A)(x) = 1$*, but if* $R(y_1, x) = 1$*,* $(R \downarrow A)(x) = 0.9999$

This example illustrates that one change in the membership values of A or R can result in major changes in the traditional fuzzy rough lower approximation, and that the OWA fuzzy rough lower approximation is not sensitive to these small changes.

An important advantage of OWA fuzzy rough sets is that the monotonicity w.r.t. the fuzzy set A and the indiscernibility relation R is maintained. These properties are not fulfilled for other robust fuzzy rough set models like the variable precision fuzzy rough set model ([115, 116]) or the vaguely quantified fuzzy rough set model ([28]).

Theorem 3.1.4. *Let $A_1 \subseteq A_2, A$ be fuzzy sets in U, and $R_1 \subseteq R_2, R$ be indiscernibililty relations in U. Then*

$$R \downarrow^{OWA} A_1 \subseteq R \downarrow^{OWA} A_2$$

$$R \uparrow^{OWA} A_2 \subseteq R \uparrow^{OWA} A_2$$

$$R_1 \downarrow^{OWA} A \supseteq R_2 \downarrow^{OWA} A$$

$$R_1 \uparrow^{OWA} A \subseteq R_2 \uparrow^{OWA} A.$$

Proof. The proofs easily follow from the monotonicity properties of implicators and t-norms, and the fact that if all arguments in the OWA aggregation increase (resp. decrease), the value returned by OWA aggregation increases (resp. decreases). □

The time complexity of the OWA fuzzy rough set model is larger than the time complexity of the traditional fuzzy rough set model, due to the sort operation required for the OWA aggregation. If n is the number of instances and m the number of attributes, the asymptotic time complexity of calculating the traditional lower or upper approximation of one instance is $\mathcal{O}(nm)$, whereas the asymptotic time complexity of the OWA fuzzy rough set model is $\mathcal{O}(mn\log(n))$.

3.2 Robustness

In the previous section we showed by means of a toy example that there are less drastic changes in the values of the OWA fuzzy rough lower and upper approximation when the relation R and the concept A are slightly altered than in the traditional fuzzy rough model. In this section we want to study if this holds in general.
We set up an experiment involving 40 KEEL [3] and UCI [6] classification datasets.

Table 3.1: Data used in the experimental evaluation

	# inst.	# feat.	# class.		# inst.	# feat.	# class.
appendicitis	106	7	2	housevotes	232	16	2
australian	690	14	2	iris	150	4	3
automobile	150	25	6	led7digit	500	7	10
balance	625	4	3	lymphography	148	18	4
bands	365	19	2	mammographic	830	5	2
breast	277	9	2	monk-2	432	6	2
bupa	345	6	2	movement_libras	360	90	15
car	1728	6	4	newthyroid	215	5	3
cleveland	297	13	5	pima	768	8	2
contraceptive	1473	9	3	saheart	462	9	2
crx	653	15	2	sonar	208	60	2
dermatology	358	34	6	spectfheart	267	44	2
ecoli	336	7	8	tae	151	5	3
flare	1066	11	6	tic-tac-toe	958	9	2
german	1000	20	2	vehicle	846	18	4
glass	214	9	7	vowel	990	13	11
haberman	306	3	2	wine	178	13	3
hayes-roth	160	4	3	wisconsin	683	9	2
heart	2270	13	2	yeast	1484	8	10
hepatitis	80	19	2	zoo	101	16	7

The number of instances, number of features and number of classes of these instances are listed in Table 3.1. As the positive region plays an important role in

many data mining techniques, we measure how robust the fuzzy rough positive region is against class and attribute noise. Assume that the data set is given by the decision system $(U, \mathcal{A} \cup \{d\})$ where the attribute values are normalized such that for each numerical conditional attribute $a \in \mathcal{A}$ and for each instance $x \in U$ it holds that $a(x) \in [0, 1]$. Recall that the fuzzy rough positive region is given by

$$\forall x \in U : POS(x) = (R \downarrow [x]_d)(x) \tag{3.13}$$

for classification problems. We use the Łukasiewicz implicator $\mathcal{I}_L$ and define the fuzzy tolerance relation R as follows:

$$\forall x, y \in U : R(x, y) = \frac{\sum\limits_{a \in \mathcal{A}} R_a(x, y)}{|\mathcal{A}|}, \tag{3.14}$$

where $R_a(x, y) = 1 - |a(x) - a(y)|$ for continuous attributes, and $R_a(x, y)$ is 1 if x and y have the same values for a and 0 otherwise for a discrete attribute a.
We study the influence of attribute noise and class noise separately. For a certain noise level $n \in [0, 1]$, each attribute (resp. class) value in the new dataset has a chance n to be altered to another value in the attribute (resp. class) range. Denote by $POS(x)$ the membership degree of $x \in U$ to the fuzzy rough positive region in the original dataset, by $POS_n^a(x)$ the membership degree of $x \in U$ to the fuzzy rough positive region in the dataset with attribute noise level n and by $POS_n^c(x)$ the membership degree of $x \in U$ to the fuzzy rough positive region in the dataset with class noise level n . Consider the following error measures:

$$\text{error}_n^a = \frac{\sum\limits_{x \in U} |POS(x) - POS_n^a(x)|}{|U|} \tag{3.15}$$

$$\text{error}_n^c = \frac{\sum\limits_{x \in U} |POS(x) - POS_n^c(x)|}{|U|}. \tag{3.16}$$

The larger these values, the more differences there are between the membership degrees to the fuzzy rough positive regions of the instances in the original and the new dataset and the less robust the model is.
For each dataset, we calculate the error measures described above. We compare the traditional fuzzy rough model against the OWA fuzzy rough model where either exponential or additive weights are used.
The average error values over the 40 datasets for noise levels 0.01 until 0.30 in steps of size 0.01 are depicted in Figure 3.2.1 for attribute noise and in Figure

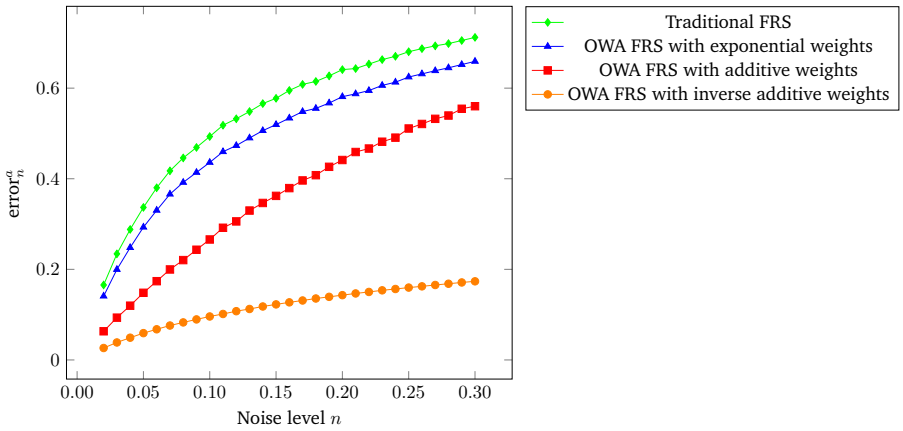


Figure 3.2.1: Average values of $error_n^a$ over 40 datasets

3.2.2 for class noise. For all noise levels, the OWA fuzzy rough model is more robust than the traditional fuzzy rough model. There is a clear order on the robustness of the OWA fuzzy rough set model: the faster the weights change and hence the more the OWA fuzzy rough set resembles the traditional fuzzy rough set model, the less robust. The most robust model uses the inverse additive weights. This can be explained by the fact that if the weights associated to the arguments of the OWA aggregation are less extreme, changing arguments has less effect on the final value. Using weights that correspond to the average would probably lead to an even more robust model, but in that case the orness and andness constraints for the OWA weights would be violated.

3.3 Conclusion

The strict minimum and maximum operators in the traditional fuzzy rough set model make it not suitable to handle noisy data. Existing approaches to improve the robustness of fuzzy rough set theory against noise deviate from the original idea of fuzzy rough set theory, violate theoretical properties of the traditional fuzzy rough set model and and are not always able to handle noise appropriately. By replacing the minimum and maximum operators by OWA aggregators, the OWA fuzzy rough set model further fuzzifies the traditional fuzzy rough set

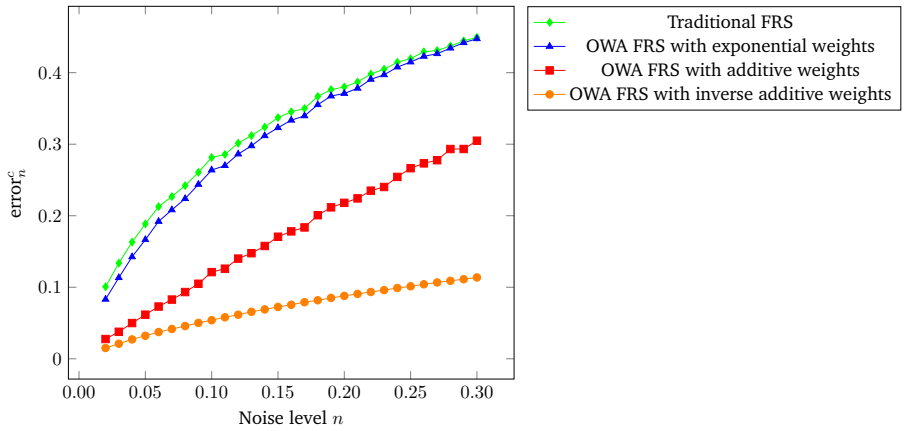


Figure 3.2.2: Average values of $error_n^c$ over 40 datasets

model. The OWA fuzzy rough set model has good theoretical properties and an experimental evaluation shows that it improves the robustness.

4. Fuzzy Rough Prototype Selection

Fuzzy rough set theory has been proven to be very useful in several machine learning fields like feature selection, classification, rule extraction and many more. The use of fuzzy rough set theory in PS is almost unexplored. However, fuzzy rough set theory intuitively seems to be an excellent tool for PS. It is a model developed to deal with data that is inconsistent (i.e. indiscernible instances have different classes) and vagueness (instances can be indiscernible to some extent). By means of the fuzzy rough lower and upper approximation, fuzzy rough set theory can model the quality or typicality of instances within their respective classes, and hence it is an ideal tool to detect border points and noisy instances.

Only one method, called Fuzzy Rough Instance Selection (FRIS, [80]) has so far been presented in the literature. It calculates the membership values of instances to the fuzzy rough positive region and removes those for which the values are lower than a certain threshold. The authors noted that removing one instance can greatly affect the membership values of other instances to the fuzzy rough positive region, and therefore they introduced alternative versions of their algorithms that iteratively re-calculate the membership values of instances to the fuzzy rough positive region. These algorithms unfortunately come with a high computational cost.

Although the idea of using fuzzy rough set theory for instance selection is valuable, there are two weaknesses associated with FRIS. The first issue is the selection of the threshold to decide if instances should be retained or not. The second problem is the aforementioned problem that the membership values of instances to the fuzzy rough positive region need to be recalculated each time an instance is removed.

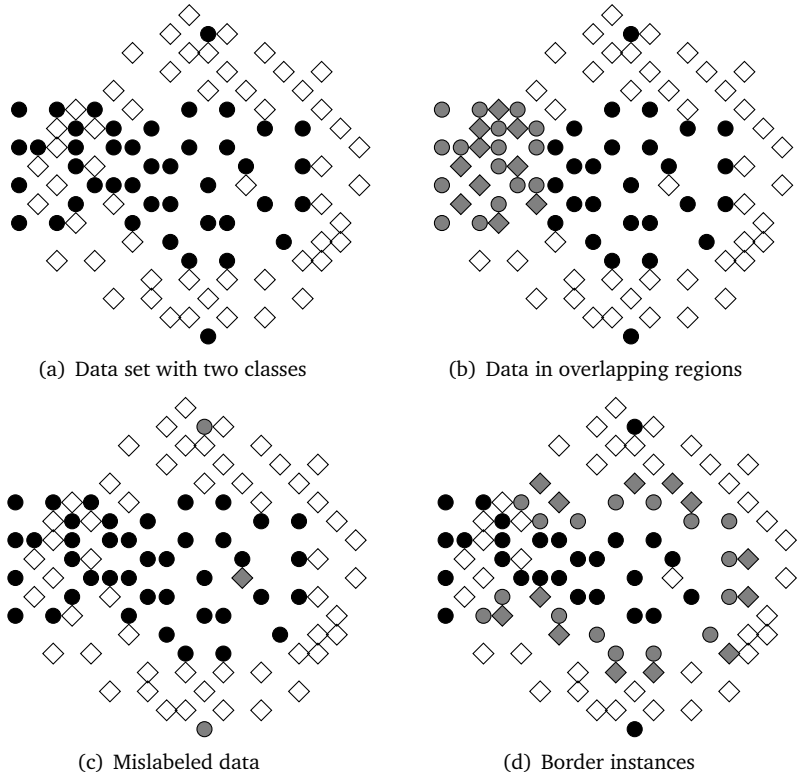
We introduce a new PS method based on fuzzy rough set theory, called Fuzzy Rough Prototype Selection (FRPS), that alleviates these problems. First of all, the

threshold is determined automatically based on the train data. Secondly, FRPS uses a different quality measure. FRIS uses the traditional fuzzy rough positive region to measure the quality of instances. We improve upon this by using OWA fuzzy rough set theory, and additionally we define a quality measure based on the fuzzy rough upper approximation. Note that, by using OWA fuzzy rough set theory, removing one instance does not drastically change the membership values of other instances to the fuzzy rough positive region or lower approximation.
In Section 4.1 we present the quality measures and in Section 4.2 we explain how the prototypes are selected using these quality measures. The algorithm presented in this chapter was proposed in [160], an earlier version of this algorithm was presented in [159]. In Section 4.3 we discuss the relationship between both approaches. We evaluate FRPS and compare it against the state-of-the-art in Section 4.4.

4.1 Quality measures based on fuzzy rough set theory

When observing a data set containing different classes, some of the instances are more typical for their class than others. Consider the example in Figure 4.1.1(a) where a data set with two classes, diamonds and circles, is depicted. We point out three types of instances that are less typical for their class or of lower quality. In Figure 4.1.1(b), instances in overlapping regions are indicated in gray. This situation can happen when the classes cannot be distinguished for a certain region in the feature space. Instances in overlapping regions are not typical for their class and as a result they are less useful for classifying new data. Another type of instances that are less valuable for classification are mislabeled data, these instances are indicated in gray in Figure 4.1.1(c). Mislabeling happens often in real-world situations where data is for instance manually annotated. Using this type of data for classification can be misleading. The last type of instances that are less typical for the class they belong to are border instances, indicated in gray in Figure 4.1.1(d). These instances are important for classification as they separate the classes, but they are less typical for their class.
Fuzzy rough set theory is an excellent tool to model the quality and typicality of instances. The positive region expresses for each instance to which extent instances similar to it belong to the same class:

$$\begin{aligned} \forall x \in U : POS(x) &= (R \downarrow [x]_d)(x) \\ &= \min_{y \in U} \mathcal{I}(R(x,y), [x]_d(y)), \end{aligned} \tag{4.1}$$



(a) Data set with two classes

(b) Data in overlapping regions

(c) Mislabeled data

(d) Border instances

Figure 4.1.1: Data set with two classes, diamonds and circles. Instances that are less typical for their class or of low quality are indicated in gray.

where $\mathcal{I}$ is an implicator, U is the entire training set and $[x]_d$ denotes the class of x. Knowing that $[x]_d(y)$ only takes values in $\{0,1\}$ and $\mathcal{I}(a,1) = 1$ for all $a \in [0,1]$, the positive region can also be rewritten as:

$$\forall x \in U : POS(x) = \min_{y \in U \setminus [x]_d} \mathcal{I}(R(x,y),0). \tag{4.2}$$

This means that the membership value of x to the positive region is low if the most similar instance from a different class than x is similar to x. Border instances, mislabeled instances and instances in border regions will have a low membership value to the positive region as there exist instances from other classes that are similar to it. Note that the FRIS algorithm [80] is based on this fuzzy rough positive region: instances $x \in U$ for which $POS(x)$ is below a certain threshold are removed.

The positive region is only based on the lower approximation. In [162], it was noted that it might also be interesting to use the upper approximation to assess the quality of instances; more specifically, analogously to the positive region, the fuzzy rough upper approximation of an instances' own class can be calculated. We denote this quality measure based on the fuzzy rough upper approximation by Q_u:

$$\forall x \in U : Q_u(x) = (R \uparrow [x]_d)(x) = \max_{y \in U \setminus \{x\}} \mathcal{T}(R(x,y), [x]_d(y)), \tag{4.3}$$

with $\mathcal{T}$ a t-norm. This value expresses the extent to which there exist instances similar to y and in the same class of y. Note that the constraint $y \neq x$ is needed because otherwise the value of $(R \uparrow [x]_d)(x)$ would always be equal to 1. The upper approximation is especially useful to detect mislabeled instances, as there are no instances of the same class close to mislabeled instances. The upper approximation of border instances and instances in overlapping regions of classes will be higher because there do exist instances from the same class in the surrounding regions. As Q_u is not able to identify border instances nor instances in overlapping regions, Q_u should be used in combination with the positive region. The $Q_{pos,u}$ value, defined as follows,

$$\forall x \in U : Q_{pos,u}(x) = POS(x) + Q_u(x), \tag{4.4}$$

balances the positive region and the Q_u value. Border instances and instances in overlapping regions will have a low membership value to POS but a normal membership value to Q_u, while mislabeled instances will have both a low membership value to POS and Q_u. The added value of using $Q_{pos,u}$ is that a

further gradation between low-quality instances can be made, for instance, mislabeled instances will have a lower $Q_{pos,u}$ value than border instances. Another advantage is that some inconsistencies can be alleviated: consider a mislabeled instance m and its nearest neighbor x, which is of a different class than m. The value of $POS(m)$ is low, as desired, but at the same time, also the value $POS(x)$ will be low. Using $Q_{POS,u}(x)$ will raise the quality value of x and solve this problem.
In the above approaches, the traditional fuzzy rough set model is used, and as a result the values of the quality measures of the instances only depend on one other instance (or two in the case of $Q_{pos,u}$). To alleviate this weakness, the OWA fuzzy rough set model can be used. In that case, the positive region is calculated as:

$$\begin{aligned} \forall x \in U : POS(x) &= (R \downarrow^{OWA} [x]_d)(x) \\ &= \underset{y \in U \setminus [x]_d}{OWA_W} \mathcal{I}(R(x,y), [x]_d(y)) \ , \end{aligned} \tag{4.5}$$

with W a min-like vector of weights. Note that instances y that are in the same class as x are not taken into account as for these instances the value $\mathcal{I}(R(x,y), [x]_d(y))$ is always equal to 1. The measure Q_u can be replaced by

$$\begin{aligned} \forall x \in U : Q_u^{OWA}(x) &= (R \uparrow^{OWA} [x]_d)(x) \\ &= \underset{y \in [x]_d, y \neq x}{OWA_W} \mathcal{T}(R(x,y), [x]_d(y)) \ , \end{aligned} \tag{4.6}$$

where W is a max-like vector of weights. Now the instances y that do not have the same class as x are not taken into account as $\mathcal{T}(R(x,y), [x]_d(y))$ is 0 for these instances. Finally, the $Q_{pos,u}^{OWA}$ measure is defined as:

$$\forall x \in U : Q_{pos,u}^{OWA}(x) = POS^{OWA}(x) + Q_u^{OWA}(x). \tag{4.7}$$

4.2 The FRPS algorithm

In the previous section we introduced three quality measures for instances in the training set U. In this section we assume that we have chosen one of those quality measures and we denote it by Q. Instances $x \in U$ for which $Q(x)$ is high are of good quality and should be retained in the prototype set, while instances for which $Q(x)$ is low are of lower quality and should be removed.
The question that automatically emerges is which threshold to use to decide if an instance should be removed or retained. Below we outline the main ideas of FRPS:

1. **Candidate thresholds** The main idea of FRPS is that it considers a wide range of candidate thresholds and selects the optimal one amongst them. The most complete range of thresholds is the set $T = \{Q(x)|x \in U\}$ that contains the values of the quality measure Q for all instances. Multiple instances can have the same value for Q, these duplicates are removed from T.

2. **Candidate prototype subsets** For each threshold $\tau \in T$ the corresponding prototype subset $S_\tau = \{x \in U|Q(x) \geq \tau\}$ is determined. It contains the best instances, as instances with a quality value lower than τ are not included in S_τ. Note that S_τ is never empty, and that $S_{min(T)}$ equals U.

3. **Leave-one-out accuracy of the candidate subsets** For each threshold $\tau \in T$, the entire training set U is classified with the 1NN classifier using S_τ as prototype subset in a leave-one-out setting. That is, for each $x \in U$ its nearest neighbor is looked up in $S_\tau \setminus \{x\}$ if $x \in S_\tau$ and in S_τ else. For each prototype subset S_τ, the corresponding training accuracy, which is the percentage of correctly classified instances in U, is denoted by acc_τ.

4. **Selecting the optimal prototype subset** The subset S_τ for which acc_τ is maximal is returned as final prototype subset. In case multiple subsets $S_{\tau_1}, \ldots, S_{\tau_t}$ correspond to the maximal training accuracy, there are three options. The first is to choose the subset $S_{max(\tau_1,\ldots,\tau_t)}$ with the least instances. The second option is to return $S_{min(\tau_1,\ldots,\tau_t)}$ with the most instances, and the last option is to return $S_{median(\tau_1,\ldots,\tau_t)}$.

Calculating the leave-one-out accuracy of the candidate subsets is computationally the most expensive part of the FRPS algorithm. For each instance and each candidate subset of prototypes, the nearest neighbor needs to be calculated. The FRPS algorithm avoids this by keeping track of the nearest neighbors of all instances and by considering the thresholds in T in an increasing manner in step 3. The pseudocode of FRPS is given in Algorithm 4.1. Assume that the thresholds in T are $\tau_1 < \tau_2 < \ldots < \tau_t$ and recall that $S_{\tau_1} = U$. In Line 5 to 8 we consider the lowest threshold τ_1 and calculate the nearest neighbors of all instances x in U where the pool of nearest neighbors is $U = S_{\tau_1}$ without x. From line 11 to 23 we iterate over the remaining thresholds. For the second threshold τ_2 the new candidate subset is S_{τ_2}. Only instances in U whose nearest neighbor in S_{τ_1} belongs to $S_{\tau_1} \setminus S_{\tau_2}$ have a different nearest neighbor in S_{τ_2}, which means that only for those instances the nearest neighbor should be recalculated. This strategy is repeated for the third threshold: the nearest neighbors only need to be recalculated for instances in U whose nearest neighbor in S_{τ_2} belongs to

$S_{\tau_2} \setminus S_{\tau_3}$. Proceeding like this ensures that during the FRPS process the number of nearest neighbors that needs to be recalculated is limited, and as a result the running time of FRPS is limited.

We illustrate the FRPS procedure with a toy example.

Algorithm 4.1 Outline of the FRPS algortihm

Input: Decision system $(U, \mathcal{A} \cup \{d\})$
$T \leftarrow \{Q(x) | x \in U\}$
Order thresholds: $T = \{\tau_1, \tau_2, \ldots, \tau_t\}$ such that $\tau_1 < \tau_2 < \ldots < \tau_t$
$acc \leftarrow 0$
for all $x \in X$ **do**
  $NN(x) =$ nearest neighbor of x in $U \setminus \{x\}$
  if $d(NN(x)) = d(x)$ **then**
    $acc \leftarrow acc + 1$
$bestacc \leftarrow acc$
$best_\tau = \{\tau_1\}$
for all $i = 2, \ldots, t$ **do**
  $S = \{x \in U | Q(x) \geq \tau_i\}$
  $acc \leftarrow 0$
  for all $x \in X$ **do**
    if $Q(NN(x)) = \tau_{i-1}$ **then**
      $NN(x) =$ nearest neighbor of x in $S \setminus \{x\}$
    if $d(NN(x)) = d(x)$ **then**
      $acc \leftarrow acc + 1$
  if $bestacc = acc$ **then**
    $best_\tau = best_\tau \cup \{\tau_i\}$
  else if $bestacc < acc$ **then**
    $best_\tau = \{\tau_i\}$
    $bestacc \leftarrow acc$
$\tau =$ median$(best_\tau)$ or $\tau =$ max$(best_\tau)$ or $\tau =$ min$(best_\tau)$
Return $S = \{x \in U | Q(x) \geq \tau\}$

Example 4.1. *Assume that there are five instances $x_1, \ldots, x_5$ in the decision system and assume that the distances between them are as in Table 4.1. The classes of the instances and their quality are listed in the last two columns of Table 4.1. There are three different thresholds to consider: $\tau_1 = 0.3$, $\tau_2 = 0.5$ and $\tau_3 = 0.7$. We calculate the nearest neighbors of all instances: $NN(x_1) = x_2$, $NN(x_2) = x_1$, $NN(x_3) = x_1$, $NN(x_4) = x_1$ and $NN(x_5) = x_1$. The current accuracy is 1, only*

Table 4.1: Toy example for FRPS with five instances

x	$d(x,x_1)$	$d(x,x_2)$	$d(x,x_3)$	$d(x,x_4)$	$d(x,x_5)$	$d(x)$	$Q(x)$
x_1	0	0.1	0.2	0.3	0.4	A	0.3
x_2	0.1	0	0.5	0.6	0.7	B	0.7
x_3	0.2	0.5	0	0.8	0.9	B	0.5
x_4	0.3	0.6	0.8	0	1	A	0.3
x_5	0.4	0.7	0.9	1	0	B	0.7

x_4 is classified correctly. The best accuracy is also 1, and the set of thresholds is $best_\tau = \{0.3\}$.
We now iterate over the remaining thresholds. The next threshold is 0.5, and the new subset is $S = \{x_2, x_3, x_5\}$. The nearest neighbors of all instances except x_1 are not in S so they need to be recalculated: $NN(x_2) = x_3$, $NN(x_3) = x_2$, $NN(x_4) = x_2$, $NN(x_5) = x_2$. Again, only one instance is classified correctly, namely x_2. The current accuracy is 1, the best accuracy remains 1 and the threshold 0.5 is added to the current set of best thresholds: $best_\tau = \{0.3, 0.5\}$. The last threshold is 0.7 and the corresponding subset is $S = \{x_2, x_5\}$. Only the nearest neighbor of x_2 needs to be recalculated: $NN(x_2) = x_5$. The accuracy is now 2, which is better than the best accuracy reached so far, so $bestacc = 2$ and the set of best thresholds is now $best_\tau = \{0.7\}$. The final subset of prototypes that is returned is $S = \{x_2, x_5\}$.

4.3 Relation between FRPS and an earlier proposal

The difference between the FRPS algorithm presented in this chapter, as proposed in [160], and the earlier version of FRPS presented in [159] (from now on referred to as FRPS') lies in the ordering of the instances. In Section 4.3.1 we explain how FRPS' orders the instances, and in Section 4.3.2 we explain how FRPS' is related to FRPS.

4.3.1 FRPS'

The FRPS' algorithm uses a particular indiscernibility relation R_α that depends on a parameter $\alpha \in [0, \infty)$, called the granularity:

$$\forall x, y \in U : R_\alpha(x,y) = \underset{a \in \mathcal{A}}{\mathcal{T}}(\max(0, 1 - \alpha\delta_a(x,y))), \tag{4.8}$$

where $\mathcal{T}$ is a t-norm and δ_a is the distance between x and y for attribute $a \in \mathcal{A}$, defined as

$$\forall x, y \in U : \delta_a(x, y) = |a(x) - a(y)| \tag{4.9}$$

for a continuous attribute and as

$$\forall x, y \in U : \delta_a(x, y) = \begin{cases} 0, & \text{if } a(x) = a(y) \\ 1, & \text{else} \end{cases} \tag{4.10}$$

for a discrete attribute. Note that, as we work in a normalized decision system where $a(x)$ takes values in $[0, 1]$ for all x in U for a continuous attribute, this definition coincides with the one in Equation (4.29) if we use $\alpha = 1$ and use a general aggregator instead of a t-norm.
The parameter α expresses how large the differences between attribute values of instances need to be in order to distinguish between them. When α is smaller, the attribute values of the instances need to differ more in order to discern between them. In the extreme case where $\alpha = 0$, all instances are indiscernible with respect to R_α. When α is larger, small differences between the attribute values of two instances are sufficient to discern between them.
The FRPS' algorithm introduces the granularity $\alpha(x)$ of an instance $x \in U$ as follows:

$$\alpha(x) = \sup\{\alpha \in [0, \infty) | POS_\alpha(x) < 1\}, \tag{4.11}$$

that is, $\alpha(x)$ is the minimum value α for which x fully belongs to the positive region POS_α. When for $x, y \in U$ it holds that $\alpha(x) > \alpha(y)$, it means that there are values α for which x does not fully belong to the positive region POS_α and y does, meaning that the quality of instance y is better than that of instance x. Hence, $\alpha(x)$ can be used as a measure to assess the lack of predictive ability of an instance $x \in U$.
The formula in Equation (4.11) is hard to use in an algorithm. Therefore, in [159] the so-called minimum granularity theorem that derives a formula for $\alpha(x)$ that is easy to use was proposed. First, we show that POS_α is increasing in α:

Lemma 4.3.1. $\forall x \in U : \forall \alpha_1, \alpha_2 \in [0, \infty) : \alpha_1 \leq \alpha_2 \Rightarrow POS_{\alpha_1}(x) \leq POS_{\alpha_2}(x)$

Proof. Assume $y \in U, a \in \mathcal{A}$ and $\alpha_1 \leq \alpha_2$. Then we have

$$\forall x \in U : \max(0, 1 - \alpha_1 \delta_a(x, y)) \geq \max(0, 1 - \alpha_2 \delta_a(x, y)). \tag{4.12}$$

As t-norms are increasing in both arguments, this means:

$$\begin{aligned} \forall x \in U : R_{\alpha_1}(x,y) &= \underset{a\in\mathcal{A}}{\mathcal{T}}(\max(0, 1-\alpha_1\delta_a(x,y))) \\ &\geq \underset{a\in\mathcal{A}}{\mathcal{T}}(\max(0, 1-\alpha_2\delta_a(x,y))) \\ &= R_{\alpha_2}(x,y). \end{aligned} \tag{4.13}$$

As implicators are decreasing in the first argument, this leads to:

$$\begin{aligned} POS_{\alpha_1}(y) &= \min_{x\in U} \mathcal{I}(R_{\alpha_1}(x,y), [y]_C(x)) \\ &\leq \min_{x\in U} \mathcal{I}(R_{\alpha_2}(x,y), [y]_d(x)) \\ &= POS_{\alpha_2}(y). \end{aligned} \tag{4.14}$$

Based on this lemma, we can find an expression for $\alpha(x)$, depending on the t-norm used.

Theorem 4.3.2. *(Minimum granularity theorem) Let $\mathcal{I}$ be an implicator such that $\forall t \in [0,1], \mathcal{I}(t,0) = 1-t$ holds (which is the case for e.g. the Łukasiewicz or Kleene-Dienes implicator), and let $x \in U$. Then if $\mathcal{T} = \mathcal{T}_M$ or $\mathcal{T} = \mathcal{T}_P$:*

$$\alpha(x) = \max_{y\in X\setminus[x]_d} \frac{1}{\max\limits_{a\in\mathcal{A}} \delta_a(x,y)}, \tag{4.15}$$

and if $\mathcal{T} = \mathcal{T}_L$:

$$\alpha(x) = \max_{y\in X\setminus[x]_d} \frac{1}{\sum\limits_{a\in\mathcal{A}} \delta_a(x,y)}. \tag{4.16}$$

Proof. First we rewrite the positive region, based on the fact that $\mathcal{I}(t,1) = 1$ for all $t \in [0,1]$, the assumption that $\mathcal{I}(t,0) = 1-t$ for all $t \in [0,1]$ and the definition of R_α. Assume $x \in U$, then:

$$\begin{aligned} POS_\alpha(x) &= \min_{y\in U} \mathcal{I}(R_\alpha(x,y), [x]_d(y)) \\ &= \min_{y\in U\setminus[x]_d} \mathcal{I}(R_\alpha(x,y), 0) \\ &= \min_{y\in U\setminus[x]_d} 1 - R_\alpha(x,y) \\ &= 1 - \max_{y\in U\setminus[x]_d} R_\alpha(x,y) \\ &= 1 - \max_{y\in U\setminus[x]_d} \underset{a\in\mathcal{A}}{\mathcal{T}}(\max(0, 1-\alpha\delta_a(x,y))). \end{aligned} \tag{4.17}$$

Now assume that $\mathcal{T} = \mathcal{T}_M$ or $\mathcal{T} = \mathcal{T}_P$. Then:

$$\begin{aligned}
&POS_\alpha(x) < 1 \\
&\Leftrightarrow \max_{y \in U \setminus [x]_d} \mathop{\mathcal{T}}_{a \in \mathcal{A}} (\max(0, 1 - \alpha\delta_a(x,y))) > 0 \\
&\Leftrightarrow \max_{y \in U \setminus [x]_d} \min_{a \in \mathcal{A}} (\max(0, 1 - \alpha\delta_a(x,y))) > 0 \\
&\Leftrightarrow \max_{y \in U \setminus [x]_d} \min_{a \in \mathcal{A}} (1 - \alpha\delta_a(x,y)) > 0 \\
&\Leftrightarrow \max_{y \in U \setminus [x]_d} (1 - \max_{a \in \mathcal{A}} (\alpha\delta_a(x,y))) > 0 \\
&\Leftrightarrow (\exists y \in U \setminus [x]_d)(\alpha < \tfrac{1}{\max_{a \in \mathcal{A}} \delta_a(x,y)}) \\
&\Leftrightarrow \alpha < \max_{y \in U \setminus [x]_d} \tfrac{1}{\max_{a \in \mathcal{A}} \delta_a(x,y)}.
\end{aligned} \tag{4.18}$$

From these equivalences it follows that

$$\alpha(x) = \max_{y \in U \setminus [x]_d} \frac{1}{\max_{a \in \mathcal{A}} \delta_a(x,y)}. \tag{4.19}$$

On the other hand, when $\mathcal{T} = \mathcal{T}_L$, it follows that:

$$\begin{aligned}
&POS_\alpha(x) < 1 \\
&\Leftrightarrow \max_{y \in U \setminus [x]_d} \mathop{\mathcal{T}}_{a \in \mathcal{A}} (\max(0, 1 - \alpha\delta_a(x,y))) > 0 \\
&\Leftrightarrow \max_{y \in U \setminus [x]_d} \mathop{\mathcal{T}}_{a \in \mathcal{A}} (1 - \alpha\delta_a(x,y)) > 0 \\
&\Leftrightarrow \max_{y \in U \setminus [x]_d} \max(0, 1 - \alpha\delta_{a_1}(x,y) + \ldots + 1 - \alpha\delta_{a_m}(x,y) - m + 1) > 0 \\
&\Leftrightarrow (\exists y \in U \setminus [x]_d)(\alpha\delta_{a_1}(x,y) + \ldots + \alpha\delta_{a_m}(x,y) < 1) \\
&\Leftrightarrow (\exists y \in U \setminus [x]_d)(\alpha < \tfrac{1}{\alpha \sum_{a \in \mathcal{A}} \delta_a(x,y)}),
\end{aligned} \tag{4.20}$$

from which it follows that

$$\alpha(x) = \max_{y \in U \setminus [x]_d} \frac{1}{\sum_{a \in \mathcal{A}} \delta_a(x,y)}. \tag{4.21}$$

These two expressions for $\alpha(x)$ are used in FRPS' to order the instances, and the rest of the algorithm is exactly the same as for FRPS.

4.3.2 Relation between FRPS and FRPS'

We show that the order imposed by the function α on the instances $x \in U$ is the reverse of the order that is imposed by POS on the instances $x \in U$, where

POS is the traditional fuzzy rough positive region with $\alpha = 1$, as used in the FRPS approach. More specifically, when $\mathcal{T} = \mathcal{T}_M$, the reverse order as defined by Equation (4.19) is recovered, and when $\mathcal{T} = \mathcal{T}_L$, the reverse order as defined by Equation (4.21) is recovered.
First note that when $\alpha = 1$, the expression in Equation (4.17) for the positive region becomes:

$$POS(x) = 1 - \max_{y \in U \setminus [x]_d} \mathop{\mathcal{T}}_{a \in \mathcal{A}} (1 - \delta_a(x, y)). \tag{4.22}$$

Assume that $\mathcal{T} = \mathcal{T}_M$, then this can be rewritten as follows:

$$\begin{aligned} POS(x) &= 1 - \max_{y \in U \setminus [x]_d} \min_{a \in \mathcal{A}} (1 - \delta_a(x, y)) \\ &= 1 - \max_{y \in U \setminus [x]_d} (1 - \max_{a \in \mathcal{A}} \delta_a(x, y)) \\ &= 1 - (1 - \min_{y \in U \setminus [x]_d} \max_{a \in \mathcal{A}} \delta_a(x, y)) \\ &= \min_{y \in U \setminus [x]_d} \max_{a \in \mathcal{A}} \delta_a(x, y). \end{aligned} \tag{4.23}$$

Now assume that $\alpha(x) \leq \alpha(z)$, with α as defined in Equation (4.19). This means that:

$$\max_{y \in U \setminus [x]_d} \frac{1}{\max_{a \in \mathcal{A}} \delta_a(x, y)} \leq \max_{y \in U \setminus [z]_d} \frac{1}{\max_{a \in \mathcal{A}} \delta_a(z, y)}, \tag{4.24}$$

which is equivalent to

$$\min_{y \in U \setminus [x]_d} \max_{a \in \mathcal{A}} \delta_a(x, y) \geq \min_{y \in U \setminus [z]_d} \max_{a \in \mathcal{A}} \delta_a(z, y), \tag{4.25}$$

which corresponds to $POS(x) \geq POS(z)$.
When $\mathcal{T} = \mathcal{T}_L$, the positive region can be rewritten as follows:

$$\begin{aligned} POS(x) &= 1 - \max_{y \in U \setminus [x]_d} \mathop{\mathcal{T}_L}_{a \in \mathcal{A}} (1 - \delta_a(x, y)) \\ &= 1 - \max_{y \in U \setminus [x]_d} (\sum_{a \in \mathcal{A}} (1 - \delta_a(x, y)) - |\mathcal{A}| + 1) \\ &= 1 + \min_{y \in U \setminus [x]_d} (\sum_{a \in \mathcal{A}} (\delta_a(x, y)) - 1). \end{aligned} \tag{4.26}$$

Now $POS(x) \leq POS(z)$ is equivalent to

$$\min_{y \in U \setminus [x]_d} \sum_{a \in \mathcal{A}} \delta_a(x, y) \leq \min_{y \in U \setminus [z]_d} \sum_{a \in \mathcal{A}} \delta_a(z, y), \tag{4.27}$$

which is also equivalent to

$$\max_{y \in U \setminus [x]_d} \frac{1}{\sum\limits_{a \in \mathcal{A}} \delta_a(x,y)} \geq \max_{y \in U \setminus [z]_d} \frac{1}{\sum\limits_{a \in \mathcal{A}} \delta_a(z,y)}, \tag{4.28}$$

which corresponds to $\alpha(x) \geq \alpha(z)$ with α as defined in Equation (4.21).
As shown above, the order imposed by the function α can also be expressed by means of the fuzzy positive region with $\alpha = 1$. This order is more intuitive and hence we presented this form of the FRPS algorithm.
We also note that in [159] the maximum operators were replaced by OWA operators that soften the maximum operators. This replacement is rather artificial and has nothing to do with OWA fuzzy rough set theory. The FRPS method introduces the OWA operators more meaningfully: instead of using the traditional fuzzy rough positive region to order the instances, the OWA fuzzy rough positive region is used, and the results obtained by FRPS are similar to the results in [159].

4.4 Experimental Evaluation

In this section we experimentally evaluate the performance of FRPS. We first discuss the experimental set-up, then compare the different parameter settings of FRPS and finally compare FRPS with the best parameter setting to the stat-of-the-art.

4.4.1 Experimental set-up

We first discuss the experimental set-up of the experimental evaluation. We list the parameters that FRPS depends on and then discuss the general setting of the experimental evaluation.

4.4.1.1 Parameters of the FRPS algorithm

The FRPS algorithm depends on four parameters, summarized in Table 4.2. The first one relates to the indiscernibility measure R used to model the quality of instances. We define the indiscernibility $R(x,y)$ between two instances $x, y \in U$ as

$$R(x,y) = \underset{a \in \mathcal{A}}{agg}(R_a(x,y)) \tag{4.29}$$

Table 4.2: Parameters of the FRPS algorithm considered in the experimental evaluation

Parameter	Options
Similarity aggregation	Average (Av.) Lukasiewicz t-norm (T_{luk}) Minimum (Min.)
Quality measure	POS Q_u $Q_{pos,u}$
OWA weights	strict (str.): W_{min}, W_{max} additive (add.): $W^{add}_{min}, W^{add}_{max}$ exponential (exp.): $W^{exp}_{min}, W^{exp}_{max}$ inverse additive (inv.)= $W^{inv}_{min}, W^{inv}_{max}$
FRPS threshold selection	minimum median maximum

where $\mathcal{A}$ is the set of conditional attributes and the aggregator agg is either the Łukasiewicz t-norm, the minimum t-norm or the average. The indiscernibility relation R_a for a single continuous attribute is the same as in Equation (3.14) and for a discrete attribute we use the crisp $0-1$ indiscernibility relation.
The second parameter is the type of quality measure that is used. There are three options: POS, Q_u and $Q_{pos,u}$.
The third parameter determines the set of OWA weights used in the quality measures. We consider four types of OWA weights for the lower and upper approximation used for the quality measures. The first weights are $W_{min} = \langle 0, \ldots, 0, 1\rangle$ and $W_{max} = \langle 1, 0, \ldots, 0\rangle$ that model the minimum and maximum for the lower and upper approximation respectively. This approach corresponds to the traditional fuzzy rough model. The other sets of weights are the additive weights, exponential weights and inverse additive weights as defined in Chapter 3. Note that when Q_{pos} is used, only the first weight vector of the OWA set of weights is used, and when Q_u is used, only the second weight vector is needed.
The last parameter relates to the FRPS algorithm itself. When multiple optimal thresholds are found during the FRPS algorithm, either the minimum, the median or the maximum among these thresholds can be selected.

4.4.1.2 Set-up of the experimental evaluation

We apply FRPS and the state-of-the-art algorithms to the data sets listed in Table 3.1. We work with a 10 fold cross-validation setting. Each PS algorithm is applied to the train data U and a subset $S \subseteq U$ is returned. The test data T is classified with the 1NN classifier using S as train data. We report the average accuracy, the average value of Cohen's kappa, the average percentage of removed instances (reduction) and average running time of the PS algorithms over the 10 folds. This running time does not include the running time of the 1NN algorithm that follows the PS. The parameter settings for the state-of-the-art PS algorithms are the same as suggested in [55].

4.4.2 Selecting the optimal parameter setting for FRPS

In the following we analyze the influence of the different parameters of FRPS. To this goal we compare the different parameter settings with respect to the average accuracy over the 40 data sets.
In Figure 4.4.1 we compare the three aggregation operators that were proposed to model the indiscernibility of instances with respect to multiple attributes: the Łukasiewicz t-norm, the minimum and the average. On the vertical axis the accuracy of the FRPS algorithm is shown for different quality measures and weight vectors. In the first graph the selected threshold is the median among the optimal thresholds, in the second graph it is the minimum and in the last graph the maximum. For each setting, using the average operator results in the best accuracy. Therefore, from now on we only consider this option for the FRPS algorithm.
The next parameter that we discuss is the weights used. In Figure 4.4.2 we compare the weights for the different quality measures and for the different threshold selectors. The weights corresponding to the traditional fuzzy rough set model perform worst. When the OWA model is used, the accuracy is improved most when inverse additive weights are chosen. Note that we showed in the previous chapter that this weight vector is most robust against class and attribute noise, which might be the reason for its good performance here. In the following we only consider these inverse additive weights.
Another choice that needs to be made is which quality measure to consider. Recall that the positive region can be used, the upper approximation or a combination of both. In Figure 4.4.3 we compare these quality measures against each other. The Q_u measure clearly performs worse than the others. The quality measure

based on the fuzzy rough upper approximation expresses for each instance to what extent there exist instances indiscernible from it in the same class. If we only use this measure to assess the quality of instances, we do not have sufficient information: we have no information about instances from other classes, and instances with high Q_u values can lie very closely to instances of other classes. The results for POS and $Q_{POS,u}$ are similar, which shows that using the fuzzy rough positive region membership values results in the best performance, and that using the fuzzy rough upper approximation has no added value. Apparently, knowledge about how far instances from other classes are is much more valuable than knowledge about how close instances from the same class are. We will use POS in the remainder of the analysis.

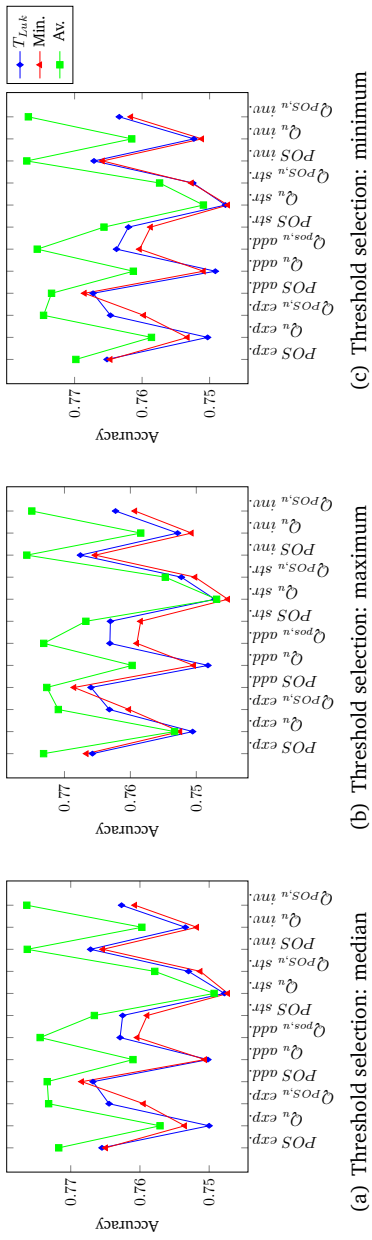
From Figure 4.4.3 we also see that the way in which the final threshold is selected among the optimal ones does not influence the accuracy much. As the best result is obtained using the minimum in combination with the other parameters, we will use the minimum in the following. This means that among the optimal subsets, the largest one is returned.

Summarized, the optimal parameter setting for the FRPS algorithm is to use the average operator to aggregate the indiscernibility degrees corresponding to the different attributes, to use the OWA fuzzy rough positive region with inverse additive weights and to select the minimum among the optimal thresholds. In the next section we compare this version of FRPS with the state-of-the-art.

4.4.3 Comparison of FRPS with the state-of-the-art

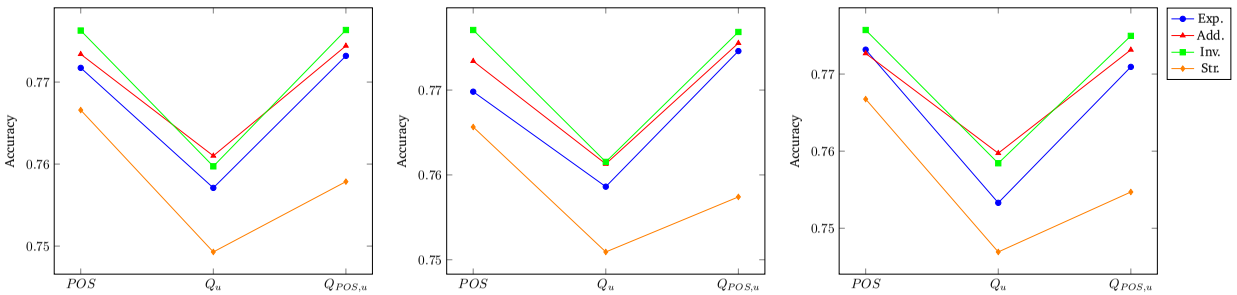
In Table 4.3 we show the averages of the evaluation measures over the 40 data sets, ordered according to performance. FRPS has the highest accuracy, and has the best value for Cohen's kappa. The closest competitors of FRPS with respect to accuracy or Cohen's kappa are the genetic approaches and RNG. When we look at running time, FRPS is slower than the filter approaches, but is clearly faster than the wrapper methods. That is, FRPS is more accurate than wrapper methods but without the computational cost that wrapper methods usually require. FRPS removes about one third of the instances. We did not expect a high reduction rate as FRPS is designed as an editing method that only removes instances that deteriorate the classification, FRPS does not aim to remove a large amount of instances.

To test if FRPS significantly outperforms the state of the art in PS methods, we first compare it pairwise with all state-of-the-art methods using the statistical Wilcoxon test. In Table 4.4 we show the values of the statistics of this test: the sum of ranks in favor of FRPS (R+), the sum of ranks in favor of the other PS



(a) Threshold selection: median (b) Threshold selection: maximum (c) Threshold selection: minimum

Figure 4.4.1: Comparison of three aggregation operators for the similarity measure in the FRPS algorithm.



(a) Threshold selection: median (b) Threshold selection: maximum (c) Threshold selection: minimum

Figure 4.4.2: Comparison of weights used for the quality measures in the FRPS algorithm.

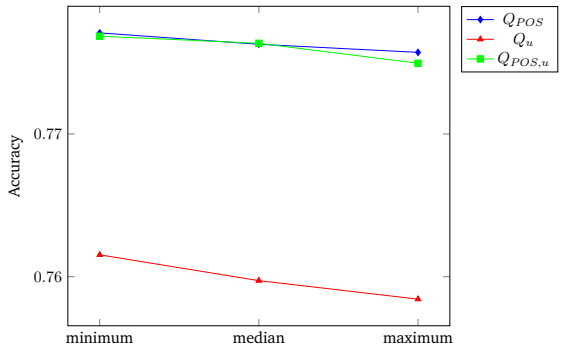


Figure 4.4.3: Comparison of the quality measures used in the FRPS algorithm.

algorithm (R-) and the asymptotic p-value (ass. p-value). FRPS significantly outperforms most of the state-of-the-art PS methods, except the genetic approaches SSGA, CHC and GGA. However, the low p-values suggest that FRPS does improve them. Note that all of these methods are slower than FRPS.
As we lose control of the family-wise error rate using the Wilcoxon test for multiple pairwise comparisons, we also use the Friedman test and Holm post-hoc procedure. We only compare the 7 best algorithms, as too many algorithms may reduce the power of the Friedman comparison. These algorithms are FRPS and additionally SSGA, SSMA, CHC, GGA, RNG and RMHC. When we compare these algorithms with respect to accuracy, the value of the Friedman statistic is 18.632143 and the p-value is 0.004832, which means that the Friedman test detects significant differences. When comparing the algorithms with respect to Cohen's kappa, the Friedman statistic is 12.950893 and the p-value is 0.043822, so again significant differences are detected. The Friedman rankings are listed in Table 4.5. FRPS get the best ranking, both for accuracy and Cohen's kappa. In the same table, we list the adjusted p-values of Holm's post-hoc procedure comparing FRPS with the other algorithms with respect to accuracy and Cohen's kappa. FRPS significantly outperforms RMHC and SSMA with respect to accuracy. The other comparisons return low values, suggesting that FRPS improves these algorithms. Holm's post-hoc procedure that compares FRPS with the other algorithms with respect to Cohen's kappa only shows that FRPS significantly outperforms RMHC, again the low p-values corresponding to the other comparisons suggest that FPRS improves the other algorithms.

In summary, we can state that FRPS significantly outperforms all filter PS methods with respect to accuracy. FRPS performs better or at least as well as the wrapper PS methods. The average experimental running time of FRPS is lower than the running time of the wrapper PS methods. In the next subsection, we analyze the theoretical time complexity of FRPS and compare it against the time complexity of the wrapper PS methods.

4.4.4 Theoretical time complexity of FRPS

Denote by n the number of instances and by m the number of features. In order to calculate the quality measure at hand for each instance, $\mathcal{O}(n^2 log(n) m)$ calculations are needed. Afterward, the nearest neighbors of all instances are calculated, requiring $\mathcal{O}(n^2 m)$ calculations. Next, FRPS iterates through all candidate thresholds and recalculates the nearest neighbors corresponding to the current threshold. In the worst case, the entire run through all thresholds requires $\mathcal{O}(n^3 m)$ calculations, as a result the worst time complexity is $\mathcal{O}(n^3 m)$. The average time complexity is lower: in each iteration, there is only $1/n$ chance for needing to recalculate the nearest neighbor of an instance. Hence, the average time complexity of FRPS is $\mathcal{O}(n^2 log(n) m)$.
The time complexity of most wrapper PS algorithms is $\mathcal{O}(\text{nev} n^2 m)$ where nev denotes the number of evaluations carried out. For each out of the nev evaluations, the training accuracy needs to be calculated, which requires $\mathcal{O}(n^2 m)$ calculations. As a typical value for nev is 10 000, which is in the range of n, the worst time complexity of FRPS is similar to the time complexity of wrapper PS algorithms, but the average time complexity of FRPS is more favorable.

4.5 Conclusion

In this chapter we have introduced a new approach to PS based on fuzzy rough set theory, called FRPS. As the fuzzy rough set model is designed to model inconsistent and vague information in data, it seems to be an excellent tool to detect border and noisy instances. We assess the quality of instances by means of the fuzzy rough lower and upper approximation, and experimentally determined that especially the fuzzy rough lower approximation is suited to model the usefulness of instances for NN classification. FRPS automatically determines a threshold to decide which instances to retain. Using the OWA fuzzy rough set model introduced in the previous chapter further improves our technique.
We experimentally showed that FRPS significantly improves the state-of-the-art in

Table 4.3: Comparison of FRPS with the state-of-the-art in PS

	Accuracy		κ		Reduction		Time (s)
FRPS	**0.7771**	**FRPS**	**0.5696**	CHC	0.9706	1NN	0
SSGA	0.7676	SSGA	0.5602	SSMA	0.9386	POP	0.0113
CHC	0.7660	CHC	0.5403	GGA	0.9363	CNN	0.0124
GGA	0.7614	GGA	0.5361	RNN	0.9151	IB3	0.0383
RNG	0.7586	SSMA	0.5333	CPruner	0.9150	MSS	0.0463
SSMA	0.7549	HMNEI	0.5332	MCNN	0.9034	FCNN	0.0506
RMHC	0.7542	RNG	0.5298	RMHC	0.9015	ModelCS	0.0515
ModelCS	0.7538	RMHC	0.5290	SSGA	0.8887	MENN	0.0706
AllKNN	0.7470	ModelCS	0.5271	DROP3	0.8183	MCNN	0.0738
HMNEI	0.7467	FRIS	0.5228	ICF	0.7155	AllKNN	0.0909
FRIS	0.7435	AllKNN	0.5079	IB3	0.6912	HMNEI	0.1280
MENN	0.7419	POP	0.4998	FCNN	0.6151	CPruner	0.1633
POP	0.7353	1NN	0.4997	CNN	0.5576	ICF	0.1724
1NN	0.7293	MENN	0.4858	HMNEI	0.5522	Reconsistent	0.4454
RNN	0.7272	MSS	0.4806	Reconsistent	0.5398	DROP3	0.5554
MSS	0.7270	RNN	0.4741	MENN	0.5222	FRIS	0.7495
FCNN	0.7095	IB3	0.4711	MSS	0.4226	**FRPS**	**3.9383**
IB3	0.7088	CNN	0.4702	**FRPS**	**0.3789**	RNG	7.0373
CNN	0.7016	FCNN	0.4699	AllKNN	0.3697	RNN	15.8304
DROP3	0.7015	DROP3	0.4523	RNG	0.2525	CHC	23.4272
Reconsistent	0.6922	MCNN	0.4476	ModelCS	0.1307	SSMA	26.9842
ICF	0.6870	Reconsistent	0.4415	FRIS	0.0865	RMHC	31.7803
MCNN	0.6830	ICF	0.4333	POP	0.0638	SSGA	56.1130
CPruner	0.6718	CPruner	0.3298	1NN	0	GGA	83.4444

Table 4.4: Statistics of the Wilcoxon test comparing FRPS to the state-of-the-art in PS with respect to accuracy and Cohen's Kappa κ.

	Accuracy			κ		
	R+	R-	p-value	R+	R-	p-value
AllKNN	693.0	127.0	0.000139	694.0	126.0	0.000131
CHC	454.0	326.0	0.368068	471.0	309.0	0.255399
CNN	788.0	32.0	0	724.0	96.0	0.000024
CPruner	798.0	22.0	0	820.0	0.0	0
DROP3	811.0	9.0	0	780.0	40.0	≤ 0.000001
FCNN	772.0	48.0	≤ 0.000001	755.0	65.0	0.000003
GGA	554.0	266.0	0.052105	566.0	254.0	0.035417
HMNEI	677.0	143.0	0.000324	588.0	232.0	0.016428
IB3	816.0	4.0	0	759.0	61.0	0.000003
ICF	810.0	10.0	0	785.0	35.0	0
MCNN	799.0	21.0	0	756.0	64.0	0.000003
MENN	583.0	197.0	0.006928	631.0	149.0	0.000751
ModelCS	702.0	118.0	0.000084	589.0	231.0	0.015834
MSS	762.0	58.0	0.000002	717.0	103.0	0.000036
POP	749.0	71.0	0.000005	638.0	182.0	0.002131
Reconsistent	820.0	0	0	798.0	22.0	0
RMHC	611.0	209.0	0.00676	589.0	231.0	0.015834
RNG	558.0	262.0	0.04593	599.0	221.0	0.010861
RNN	769.0	51.0	≤ 0.000001	769.0	51.0	≤ 0.000001
SSMA	596.0	184.0	0.003955	564.0	216.0	0.014886
FRIS	708.0	112.0	0.00006	612.0	208.0	0.006492
SSGA	546.0	274.0	0.066545	518.0	302.0	0.144736
1NN	770.0	50.0	≤ 0.000001	670.0	150.0	0.000463

Table 4.5: Friedman rankings of FRPS and the best state-of-the-art PS methods with respect to accuracy and Cohen's kappa κ, and adjusted p-values returned by Holm's post-hoc procedure.

	Accuracy		Kappa	
	Friedman Ranking	Adj. p-value	Friedman Ranking	Adj. p-value
CHC	3.738	0.2804	3.950	0.1486
GGA	4.013	0.1228	4.125	0.1486
RMHC	4.925	0.0005	4.788	0.0024
RNG	3.688	0.2804	4.038	0.1486
SSMA	4.388	0.0240	4.100	0.1486
SSGA	4.225	0.0519	3.925	0.1486
FRPS	3.025	-	3.075	-

PS methods, with the additional asset that FRPS is faster than the most accurate PS techniques. In the next two chapters we further elaborate on FRPS and use it in combination with feature selection and verify if FRPS also benefits SVM classification.

5. Combining Fuzzy Rough and Evolutionary Prototype and Feature Selection

As shown in the previous chapter, FRPS is a very accurate PS method that improves the state-of-the-art. In this chapter we want to study if combining Feature Selection (FS, [68, 92, 138, 104]) with FRPS can further enhance the performance of 1NN classification. FS is a well-studied topic in data mining that removes features from the data before using it for classification. Many researchers have studied the use of fuzzy rough set theory for FS. Most of the work focuses on removing as many features as possible preserving the predictive ability of the decision system [30, 14, 85, 84]. In [120], the authors introduce a method that simultaneously carries out PS and FS, based on the concept of bireducts [147]. Again, the authors aim to find a prototype and feature subset that preserves the predictive ability of the decision system.

As we are interested in raising the accuracy, even at the cost of removing less features, we do not use fuzzy rough set theory for the FS component. Motivated by the fact that a Steady-State Genetic Algorithm for FS (further referred to as FS-SSGA) is able to improve the accuracy of 1NN [36], we will work with this algorithm in this chapter. A straightforward way to combine FRPS and FS-SSGA would be to apply these algorithms in sequence. However, the question raises which order to use: first apply FS and then PS or the other way around. In order to decide which features to remove, one should not rely on noisy instances, but on the other hand, noisy instances should not be determined using misleading features. Intuitively, one feels that the prototype and feature subsets should be adapted to each other and therefore a simultaneous approach is in place.

This strategy has been applied successfully in [36], where the authors simultaneously apply SSGA for PS (denoted by PS-SSGA in the following) and QuickReduct (QR, [30]), a FS algorithm based on fuzzy rough set theory. Their proposal, to

which we refer as SIM-SSGA-QR, carries out the PS-SSGA algorithm and applies QR at certain stages in the algorithm. As such, the feature and prototype subset are accustomed to each other. We propose SIM-FRPS-SSGA, which carries out FS-SSGA and applies FRPS at certain stages in the algorithm. Based on the observation from the previous chapter that FRPS outperforms PS-SSGA and the fact that FS-SSGA improves QR, we expect SIM-FRPS-SSGA to outperform SIM-SSGA-QR.
We present the algorithm in Section 5.1, discuss the baselines against which we compare SIM-FRPS-SSGA in Section 5.2 and we experimentally evaluate the performance of SIM-FRPS-SSGA in Section 5.3.

5.1 Simultaneous FRPS and SSGA Feature Selection

A straightforward way to combine FS and PS would be to first apply FRPS to the dataset, followed by FS, or the other way around. However, the FRPS process highly depends on the features included in the dataset, and also the FS process depends on the instances included in the data. This problem demands for approaches that simultaneously carry out PS and FS.
The SIM-FRPS-SSGA algorithm [38] uses the FRPS algorithm for the PS part and relies on an evolutionary search strategy for the FS process. The main idea is that an evolutionary algorithm for FS is carried out, and that at certain points in the algorithm the instances in the decision system are updated using FRPS.
We first recall the FS-SSGA algorithm, an evolutionary strategy for FS that is at the basis of SIM-FRPS-SSGA. The outline of FS-SSGA is given in Algorithm 5.1. Feature subsets are represented by binary strings where a gene is 1 if the feature is included and 0 else. The fitness of an individual i depends on two components. The first component is the accuracy acc_i obtained when classifying each instance $x \in U$ with the 1NN rule using $U \setminus \{x\}$ as pool of candidate nearest neighbors, where distances are only based on the feature subset represented by the individual i. The second component is the percentage of removed features red_i, that is, the smaller the feature subset, the higher the fitness. The two components are balanced using a parameter α:

$$\text{fitness}_i = \alpha acc_i + (1 - \alpha) red_i. \tag{5.1}$$

The higher α, the less important the reduction.
In each generation, two parents are selected in the population using a binary

tournament: the first parent is the fittest one among two randomly picked different individuals, and the second parent is selected in the same way. Next, two-point crossover is applied to these parents and two children are produced. Mutation is applied to these two children, that is, for each gene in the child, the bit is flipped with a certain low probability. Next, the fitness of the offspring is calculated, and if the fitness of the two worst individuals in the current population is lower than the fitness of this offspring, these worst individuals are replaced by the offspring. This process is repeated until a certain maximum number of evaluations is reached.

Our proposal, SIM-FRPS-SSGA is based on this FS-SSGA algorithm. Before

Algorithm 5.1 Outline of the FS-SSGA algorithm

1: **Input:** A decision system $(U, \mathcal{A} \cup \{d\})$, parameters n_{ev}^{max}, α
2: Initialize the chromosomes in the population P: random binary strings of length $|\mathcal{A}|$.
3: Evaluate the individuals in P, $n_{ev} \leftarrow |P|$
4: **while** $n_{ev} < n_{ev}^{max}$ **do**
5: Select two parents p_1 and p_2 for offspring in P (binary tournament)
6: Apply two-point crossover to p_1 and p_2, offspring is c_1 and c_2
7: Apply bit-flip mutation to c_1 and c_2
8: Evaluate c_1 and c_2
9: $n_{ev} \leftarrow n_{ev} + 2$
10: Replace two worst individuals in P if fitness of c_1 and c_2 is better
11: **Output:** The feature subset corresponding to the fittest individual in P.

introducing our algorithm we fix an important notation: given a decision system $(U, B \cup \{d\})$ with $B \subseteq \mathcal{A}$, we denote by FRPS$(U, B \cup \{d\})$ the prototype subset returned by FRPS when B is used as feature subset. That is, instead of using the entire feature set $\mathcal{A}$ to calculate the quality measure in FRPS and to evaluate candidate prototype subsets, only the features in B are used.

The outline of SIM-FRPS-SSGA is given in Algorithm 5.2. During the course of the algorithm, the subset of features B and the subset of instances S are simultaneously updated. The feature subset B is initialized as the entire feature set in line 3, and S is initialized by applying the FRPS algorithm to the original decision system with all features included in line 2.

Next, the first steps of the SSGA algorithm for FS are applied in line 13 to 18. First, two parents are selected in the population using a binary tournament. These parents produce two children using two-point crossover, and mutation

is applied to this offspring. Next, the fitness of the offspring is calculated. The reduction component is derived as usual, but the accuracy component is more involved. In order to calculate the leave-one-out-accuracy, all instances in U are classified using the current subset S of instances as pool of candidate nearest neighbors, where distances are based on the feature subset that is represented by the current fittest individual. The two least fit individuals in the population are replaced by the offspring if their fitness is worse. The best individual is stored in B in line 22.
After every 50 generations (note that in each generation two evaluations are completed), a PS step is carried out in line 11 and 12. The FRPS algorithm is applied using the feature subset B, that is, the quality measures are based on the features B and also the training accuracy is obtained using the features in B only. Only if the leave-one-out accuracy is improved with this FRPS step, the new prototype subset is used. After a certain number of evaluations is reached, the SIM-FRPS-SSGA algorithm turns into a stabilizing phase, where no PS is carried out anymore, but where only the feature subset is optimized. This stabilization phase ensures that the final feature subset is sufficiently optimized for the final prototype subset.

5.1.1 Theoretical time complexity of SIM-FRPS-SSGA

We denote by n the number of instances and by m the number of features in the decision system. Recall from the last chapter that the worst theoretical time complexity of FRPS is $\mathcal{O}(n^3m)$, and the average time complexity $\mathcal{O}(n^2log(n)m)$. The FS-SSGA algorithm requires $\mathcal{O}(\text{nev}n^2m)$ operations: for each evaluation, all nearest neighbors need to be recalculated, which requires $\mathcal{O}(n^2m)$ operations. This means that the SIM-FRPS-SSGA algorithm requires at least $\mathcal{O}(\text{nev}n^2m)$ operations. Additionally, the FRPS algorithm needs to be carried out $\mathcal{O}(nev/100)$ times. This means that the worst time complexity of SIM-FRPS-SSGA is $\mathcal{O}(\text{nev}n^3m)$ and the average time complexity $\mathcal{O}(\text{nev}n^2log(n)m)$. Note that in practice, the algorithm will be faster as the application of FS-SSGA and FRPS is carried out on a part of the data in each step. FRPS removes on average about 40 percent of the instances, which means that FS-SSGA is mostly carried out on 60 percent of the instances. On the other hand, we will see in the experimental evaluation that FS-SSGA removes about 50 percent of the features, meaning that FRPS is only carried out on half of the data.

Algorithm 5.2 Outline of the SIM-FRPS-SSGA algorithm

Input: A decision system $(U, \mathcal{A} \cup \{d\})$, parameters n_{ev}^{max}, β
$S \leftarrow$ FRPS$(U, \mathcal{A} \cup \{d\})$
$B \leftarrow \mathcal{A}$
Initialize the chromosomes in the population P: random binary strings of length $|\mathcal{A}|$.
Evaluate the individuals in P, $n_{ev} \leftarrow |P|$
stabilize $\leftarrow$ false
count $\leftarrow 0$
while $n_{ev} < n_{ev}^{max}$ **do**
  if $count = 100$ AND stabilize = false **then**
    count $\leftarrow 0$
    if acc$(U, B \cup \{d\}) <$ acc(FRPS$(U, B \cup \{d\}), B \cup \{d\})$ **then**
      $S \leftarrow$ FRPS$(U, B \cup \{d\})$
  Select two parents p_1 and p_2 for offspring in P (binary tournament)
  Apply two-point crossover to p_1 and p_2, offspring is c_1 and c_2
  Apply bit-flip mutation to c_1 and c_2
  Evaluate c_1 and c_2 using S as set of instances
  $n_{ev} \leftarrow n_{ev} + 2$
  count $\leftarrow$ count $+ 2$
  if $n_{ev} > \beta n_{ev}^{max}$ **then**
    stabilize $\leftarrow$ true
  Replace two worst individuals in P if fitness of c_1 and c_2 is better
  $B \leftarrow$ best individual in P
Output: Prototype subset S and feature subset B

Table 5.1: Overview of the baselines against which we compare SIM-FRPS-SSGA

Algorithm	Summary
FRPS [160]	Fuzzy rough algorithm for PS
QR [30]	Fuzzy rough algorithm for FS
FS-SSGA	Evolutionary approach for FS
PS-SSGA	Evolutionary approach for PS
SIM-SSGA [36]	Evolutionary approach for simultaneous PS and FS
SIM-SSGA-QR [36]	Simultaneous PS and FS, where QR is used for the FS part and SSGA for the PS part
FRPS→QR	FRPS followed by QR
QR→FRPS	QR followed by FRPS
FRPS→FS-SSGA	FRPS followed by FS-SSGA
FS-SSGA→FRPS	FS-SSGA followed by FRPS
QR→PS-SSGA	QR followed by PS-SSGA
PS-SSGA→QR	PS-SSGA followed by QR
FS-SSGA→PS-SSGA	FS-SSGA followed by PS-SSGA
PS-SSGA→FS-SSGA	PS-SSGA followed by FS-SSGA
1NN	No preprocessing

5.2 Baselines

In this section we give an overview of the baselines against which we compare the SIM-FRPS-SSGA algorithm. All baselines are listed in Table 5.1. The first four algorithms are either separate FS or PS components. FRPS is the algorithm presented in the previous chapter, QR is a fuzzy rough approach to FS that will be explained in Section 5.2.1. Note that we denote the SSGA approach to PS by PS-SSGA to make the distinction between PS-SSGA and FS-SSGA, whereas we used the notation SSGA for PS-SSGA in the previous chapter. This PS-SSGA algorithm is recalled in Section 5.2.2. The next two approaches carry out simultaneous FS and PS, these algorithms are explained in Section 5.2.3 and 5.2.4. The last eight approaches are processes where FS and PS are carried out sequentially. Note that, for a better understanding and overview of the algorithms, the notations for the algorithms used in this chapter deviate from the notations in their respective papers.

5.2.1 QuickReduct

The goal of QR [30] is to find a minimal subset of features $B \subseteq \mathcal{A}$ such that the cardinality of the fuzzy rough positive region corresponding to B is equal to the cardinality of the fuzzy rough positive region corresponding to the entire set of features $\mathcal{A}$. More specifically, QR finds a subset $B \subseteq \mathcal{A}$ such that $|POS_B| = |POS_{\mathcal{A}}|$. The outline of QR is given in Algorithm 5.3. In each iteration, the feature that improves the cardinality of the fuzzy rough positive region most is included in the current subset of features, and this process is repeated until $|POS_B| = |POS_{\mathcal{A}}|$. Note that this algorithm terminates, as the fuzzy rough positive region is increasing in B, and as the entire subset $B = \mathcal{A}$ trivially fulfills the termination criterion.

Algorithm 5.3 The QR algorithm

Input: A decision system $(U, \mathcal{A} \cup \{d\})$
Calculate $|POS_{\mathcal{A}}|$
$B \leftarrow \{\}$
$current_{POS} \leftarrow 0$
while $current_{POS} < |POS_{\mathcal{A}}|$ **do**
  $T \leftarrow B$
  $best_{POS} \leftarrow 0$
  for $b \in \mathcal{A} \setminus B$ **do**
    if $|POS_{B \cup \{b\}}| > best_{POS}$ **then**
      $T \leftarrow B \cup \{b\}$
      $best_{POS} \leftarrow |POS_{B \cup \{b\}}|$
  $B \leftarrow T$
Output: Feature subset B

5.2.2 PS-SSGA

The outline of PS-SSGA is given in Algorithm 5.4. The candidate prototype subsets are represented by binary strings of length $|U|$ where a gene is 1 if the prototype is included and 0 else. The fitness of an individual i consists of an accuracy component acc_i that is obtained when classifying each instance x in U using the prototype subset S represented by i as pool of candidate neighbors with a leave-one-out strategy. The second component of the fitness is the reduction rate red_i in terms of instances. These components are balanced using a parameter

α:

$$\text{fitness}_i = \alpha acc_i + (1 - \alpha) red_i. \tag{5.2}$$

After initialization, subsequent generations are produced until a certain maximum number of evaluations is carried out. In each step, two parents are selected using a binary tournament. Two-point crossover is applied to these parents to produce offspring. After applying mutation to the children, the worst individuals in the population are replaced if their fitness is improved by the new offspring.

Algorithm 5.4 Outline of the PS-SSGA algorithm

1: **Input:** A decision system $(U, \mathcal{A} \cup \{d\})$, parameters n_{ev}^{max}, α
2: Initialize the chromosomes in the population P: random binary strings of length $|U|$.
3: Evaluate the individuals in P, $n_{ev} \leftarrow |P|$
4: **while** $n_{ev} < n_{ev}^{max}$ **do**
5: Select two parents p_1 and p_2 for offspring in P
6: Apply two-point crossover to p_1 and p_2, offspring is c_1 and c_2
7: Apply bit-flip mutation to c_1 and c_2
8: Evaluate c_1 and c_2 using S as set of instances
9: $n_{ev} \leftarrow n_{ev} + 2$
10: Replace two worst individuals in P if fitness of c_1 and c_2 is better
11: **Output:** The prototype subset corresponding to the fittest individual in P.

5.2.3 SIM-SSGA-QR

In [36], a method for simultaneous PS and FS based on the QR algorithm was proposed. We refer to this method as SIM-SSGA-QR. This method is symmetrical to the SIM-FRPS-SSGA approach, that is, the PS process follows an SSGA approach, while the FS part is achieved using fuzzy rough set theory.
The outline of SIM-SSGA-QR is given in Algorithm 5.5. We denote by QR$(S, \mathcal{A} \cup \{d\})$ the subset of features that is returned when applying QR using only the instances in $S \subseteq U$. First, the QR algorithm is applied to the decision system and returns a feature subset B. If the leave-one-out accuracy using B as feature subset is higher than when the entire feature subset is used, the initial feature subset is B, and else the entire feature set $\mathcal{A}$ is used. Next, the population is initialized randomly; the prototype subsets are represented by binary strings of length $|U|$.
After this initialization, PS-SSGA is carried out: two parents are chosen using

a binary tournament, then two-point crossover is applied and produces two children. Mutation is applied to this offspring, that is, each bit in the corresponding gene is flipped with a certain (low) probability. Next, these children are evaluated, where the accuracy component in the fitness function is obtained by applying the 1NN rule with the current set of features B. If the fitness of the worst two individuals in the population is worse than the fitness of the offspring, these two individuals are replaced by the offspring. This process is repeated, and after every 100 evaluations (i.e. 50 generations), the QR algorithm is carried out. This QR algorithm uses the instances represented by the best individual in the population. If the leave-one-out accuracy using the new feature subset is higher than when the previous feature subset is used, the optimal feature subset is replaced.
When a certain number of evaluations is reached, the FS process is stopped; the algorithm arrives at a stabilizing phase where only the evolutionary PS part is carried out.

5.2.4 SIM-SSGA

The SIM-SSGA algorithm uses a SSGA approach to simultaneously improve the feature subset B and the prototype subset S. The features and instances are encoded in one chromosome, that is, the individuals are represented by binary strings of length $|U| + |\mathcal{A}|$, where the first part represents the prototype subset and the second part the feature subset. The SSGA procedure follows the usual SSGA procedure, except for the crossover part: instead of using a single two-point crossover procedure, a double two-point crossover is needed, that is, the chromosome is split in the prototype part and the feature part, and the two-point crossover is applied to each part separately.

5.2.5 Theoretical time complexity of the baselines

Denote by n the number of instances and m the number of features in the decision system. The time complexity of the FS-SSGA, PS-SSGA and SIM-SSGA is $\mathcal{O}(\text{nev}n^2m)$, as for each out of the nev evaluations the nearest neighbors of all instances need to be calculated, requiring $\mathcal{O}(n^2m)$ operations. The theoretical time complexity of QR is $\mathcal{O}(m^3n^2)$: in each iteration all features are considered, and for each feature the positive region of all instances needs to be calculated, requiring $\mathcal{O}(n^2m)$ operations. As there are maximally m iterations, the positive region of all instances needs to be calculated $O(m^2)$ times. The time complexity

Algorithm 5.5 Outline of the SIM-SSGA-QR algorithm

Input: A decision system $(U, \mathcal{A} \cup \{d\})$, parameters n_{ev}^{max}, β
if acc$(U, \mathcal{A} \cup \{d\}) <$ acc$(U,$ QR$(U, \mathcal{A} \cup \{d\}) \cup \{d\})$ **then**
  $B \leftarrow$ QR$(U, \mathcal{A} \cup \{d\})$
else
  $B \leftarrow \mathcal{A}$
Initialize the chromosomes in the population P: random binary strings of length $|U|$.
Evaluate the individuals in P, $n_{ev} \leftarrow |P|$
$S \leftarrow U$
stabilize $\leftarrow$ false
count $\leftarrow 0$
while $n_{ev} < n_{ev}^{max}$ **do**
  if count $= 100$ AND stabilize $=$ false **then**
    count $\leftarrow 0$
    if acc$(S, \mathcal{A} \cup \{d\}) <$ acc$(S,$ QR$(S, \mathcal{A} \cup \{d\}) \cup \{d\})$ **then**
      $B =$ QR$(S, \mathcal{A} \cup \{d\})$
  Select two parents p_1 and p_2 for offspring in P (binary tournament)
  Apply two-point crossover to p_1 and p_2, offspring is c_1 and c_2
  Apply bit-flip mutation to c_1 and c_2
  Evaluate c_1 and c_2 using B as set of features
  $n_{ev} \leftarrow n_{ev} + 2$
  count $\leftarrow$ count $+ 2$
  if $n_{ev} > \beta n_{ev}^{max}$ **then**
    stabilize $\leftarrow$ true
  Replace two worst individuals in P if fitness of c_1 and c_2 is better
  $S \leftarrow$ best individual in P

of SIM-SSGA-QR is at least $\mathcal{O}(\text{nev}n^2m)$, and additionally the QR algorithm needs to be carried out nev$/100$ times. As a result, the theoretical time complexity of SIM-SSGA-QR is $\mathcal{O}(\text{nev}n^2m^3)$.
We note that when the FS and PS algorithms are carried out simultaneously or in sequence, the running time is lower as the algorithms are applied to only a part of the data.

5.3 Experimental Evaluation

In this section we evaluate SIM-FRPS and compare it to the baselines. In Section 5.3.1 we explain the setup of the experimental evaluation, and in Section 5.3.2 we present the results.

5.3.1 Experimental setup

We first discuss the parameters of the algorithms. The QR algorithm depends on an indiscernibility relation R, which we define as follows for a subset $B \subseteq \mathcal{A}$:

$$\forall x, y \in U : R_B(x, y) = \min_{b \in B} R_b(x, y), \tag{5.3}$$

where R_b is defined as in Equation (3.14) for separate features. The reason why we use the minimum operator here instead of the average is that the termination criterion in the QR algorithm is reached faster when using $\min$, resulting in smaller feature subsets.
The settings for the FRPS algorithm are the ones with which we obtained the best results in the previous chapter.
All SSGA approaches are carried out with 10000 evaluations, population size 50 and mutation probability 0.005 per bit. The parameter α that balances the reduction and accuracy in the fitness function is 0.5 for PS and 0.99 for FS. These parameters were suggested in [21] and [36]. We could tune the parameters for the SSGA part, but as this can hide the net contribution of our proposal, we choose to fix these parameters. All SSGA algorithms are repeated five times and the average results over these runs are reported. The classifier used is 1NN.
We work with a 10 fold cross validation scheme and report the accuracy, Cohen's kappa, the reduction in terms of instances, the reduction in terms of features and the running time that only covers the execution time of the preprocessing procedure, not the classification afterwards. We apply SIM-FRPS-SSGA and the baselines to the 40 datasets listed in Table 3.1.

5.3.2 Results

In Table 5.2 we show the average results over the 40 datasets of our proposal SIM-FRPS-SSGA and the baselines, ordered according to accuracy.

FS→PS vs. PS→FS The first question we pose is whether PS should be carried out before FS or the other way around if we use a sequential strategy. In all cases, first applying FS and then PS results in the best performance, both with respect to accuracy and Cohen's kappa. A possible explanation for this could be that the FS methods perform better when they have more information, i.e. more instances, and that the PS methods perform better if only the relevant features are selected. The bad results for PS-SSGA→QR and PS-SSGA→FS-SSGA can be explained by the fact that PS-SSGA removes about 90 percent of the instances, and that the FS part happens on a small part of the data.

Simultaneous vs. (sequential) FS and PS Another interesting conclusion is that simultaneous application of FS and PS performs better than sequential FS and PS. This confirms our hypothesis that the feature and prototype subset should be matched carefully. We also note that the simultaneous application of FS and PS outperforms the separate components. For instance, our proposal SIM-FRPS-SSGA improves both FS-SSGA and FRPS. This shows that the FS and PS components are able to enhance each other through simultaneous application.

Reduction The instance reduction rates of the simultaneous and sequential approaches are more or less in line with the reduction rates of their separate components. Methods incorporating PS-SSGA remove about 89 percent of the instances on average, whereas methods based on FRPS remove about 38 percent of the instances. This does not hold for the feature reduction rates. For instance, SIM-FRPS-SSGA removes about 39 percent of the features whereas FS-SSGA removes 52 percent of the features. Methods incorporating QR remove in general few instances. When QR is applied after a PS method, it removes more features. A possible explanation could be that when there are fewer instances, the condition that the cardinality of the positive region w.r.t. the selected features is as large as the positive region w.r.t. the original feature subset can be fulfilled more easily. The running times required by the QR and FRPS components are clearly lower than the components involving an SSGA component.

Comparing SIM-FRPS-SSGA against the baselines Our proposal SIM-FRPS-SSGA outperforms the baselines on average, both with respect to accuracy and Cohen's kappa. It improves SIM-SSGA-QR, which is a consistent result as the components of SIM-FRPS-SSGA outperform the components of SIM-SSGA-QR. The closest competitor of SIM-FRPS-SSGA is SIM-SSGA. Unfortunately, SIM-FRPS-SSGA removes fewer instances and features and is slower than SIM-SSGA. To test if the benefits of SIM-FRPS-SSGA compensate for this lower reduction rate and computational time we test if it significantly outperforms the baselines with respect to accuracy and Cohen's kappa.
We first focus on accuracy. The value of the Friedman statistic is 158.17, and the p-value is smaller than 0.000001, meaning that significant differences are detected. The Friedman rankings are listed in the second column of Table 5.3, SIM-FRPS-SSGA gets the best ranking. The value of the Friedman statistic comparing the algorithms with respect to Cohen's kappa is 131.42, and the p-value is smaller than 0.000001, so again significant differences are detected among the methods. In the third column of Table 5.3 we show the Friedman rankings with respect to Cohen's kappa, again SIM-FRPS-SSGA gets the best ranking.
In the last two columns of the same table, we list the adjusted p-values produced by the Holm post-hoc procedure for accuracy and Cohen's kappa. All methods are significantly outperformed by SIM-FRPS-SSGA, except SIM-SSGA, QR→FRPS and FRPS, but the low p-values suggest that SIM-FRPS-SSGA does improve these methods.

5.4 Conclusion

The good performance of FRPS in the previous chapter motivated us to use it in combination with FS. Our proposal, called SIM-FRPS-SSGA, carries out a genetic FS algorithm, and performs FRPS at certain stages during the course of the algorithm. As such, we hope that the resulting feature and prototype subsets cohere better than when the FS and PS algorithms are carried out in sequence.
Our experimental evaluation shows that our approach indeed outperforms its components, and that the simultaneous approach performs better than the sequential approach. Moreover, we improve other simultaneous approaches that solely use genetic approaches, or that use a fuzzy rough based strategy for the feature selection component. Once again, this demonstrates the benefits of fuzzy rough set theory in PS over genetic approaches.

Table 5.2: Average results of SIM-FRPS-SSGA and the baselines over the 40 datasets.

	acc	κ	red_i	red_f	Time (s)
SIM-FRPS-SSGA	0.7983	0.6050	0.3604	0.3898	67.3458
SIM-SSGA	0.7820	0.5800	0.9006	0.4959	23.6398
FS-SSGA	0.7790	0.5723	0	0.5248	69.3546
FRPS	0.7771	0.5696	0.3789	0	3.9378
SIM-SSGA-QR	0.7723	0.5615	0.8990	0.1973	32.1605
PS-SSGA	0.7698	0.5602	0.8887	0	56.1133
QR→FRPS	0.7639	0.5515	0.3725	0.0887	5.3630
FS-SSGA→PS-SSGA	0.7584	0.5389	0.8910	0.5223	85.3515
QR→PS-SSGA	0.7556	0.5439	0.8807	0.0887	48.4763
FS-SSGA→FRPS	0.7478	0.5153	0.3473	0.5246	73.5386
FRPS→QR	0.7378	0.5132	0.3789	0.1927	4.6109
QR	0.7300	0.5048	0	0.0887	4.6968
1NN	0.7293	0.4997	0	0	0
FRPS→FS-SSGA	0.7060	0.4592	0.5557	0.5557	38.6447
PS-SSGA→QR	0.6823	0.4340	0.8782	0.3202	45.1226
PS-SSGA→FS-SSGA	0.6478	0.3757	0.8778	0.5794	320.7321

Table 5.3: Statistics of the Friedman test and Holm post-hoc procedure comparing the baselines with our proposal SIM-FRPS-SSGA

	Ranking acc.	Ranking κ	Adj. p-value acc.	Adj. p-value κ
SIM-FRPS-SSGA	4.10	4.33	-	-
FRPS	5.69	6.39	0.13591	0.070327
QR→FRPS	6.23	6.74	0.091848	0.070327
SIM-SSGA	6.58	6.65	0.060239	0.070327
PS-SSGA	7.31	7.43	0.010191	0.017959
SIM-SSGA-QR	7.39	7.38	0.010073	0.017959
FS-SSGA	7.88	7.85	0.002347	0.005575
FS-SSGA→FRPS	8.13	8.39	0.001094	0.000949
FS-SSGA→PS-SSGA	8.40	8.70	0.000429	0.000396
FRPS→QR	8.53	8.58	0.000291	0.000589
QR→PS-SSGA	8.65	8.50	0.000192	0.000703
QR	10.25	9.31	≤ 0.000001	0.000031
FRPS→FS-SSGA	10.84	11.09	≤ 0.000001	≤ 0.000001
1NN	11.13	9.81	≤ 0.000001	0.000003
PS-SSGA→QR	11.15	11.20	≤ 0.000001	≤ 0.000001
PS-SSGA→FS-SSGA	13.78	13.68	≤ 0.000001	≤ 0.000001

6. Feature and Training Set Selection for Support Vector Machines

In the previous chapters we have studied how fuzzy rough and evolutionary PS and FS can improve 1NN classification. We want to take this research one step further and verify if these techniques can also improve other classifiers. We focus on the SVM classifier as this is a widely used and very accurate state-of-the-art classifier. Recall that we refer to instance selection for SVMs as Training Set Selection (TSS).

The concept of using TSS and FS for SVM is notably different from PS and FS for 1NN classification. The 1NN classifier is model-free, which means that in order to classify new instances the train data is used in its original form. Consequently, if the train data is modified, the classification of new instances is independently influenced. This is not true for SVM classification, where a classification model is built on the train data before using it for classification. When changing the train data, the classification model changes, which in turn can impact the classification of new data. Remind that the SVM model defines classification borders between the instances. When TSS is applied to the data, these borders can alter, but it does not necessarily mean that the classification of new instances drastically changes. In other words, the classification model acts as a buffer between the application of TSS and the classification of new instances. As a result, we expect that the effect of TSS on SVM will be smaller than the effect of PS on KNN. Another difference is that by using smooth decision boundaries, SVMs are constructed to cope with noisy data, while 1NN classification is highly susceptible to noise. For these and other reasons, TSS and FS for SVMs is more challenging, and the effect on SVM classification will be less clear than for 1NN classification.

In this chapter we try to improve SVM classification in three ways. We first consider filter TSS techniques, these methods can be used as preprocessing

techniques for SVM without any adjustments. However, we note that some of these techniques are based on NN ideas and might be less suited for SVM classification. We also consider two filter TSS techniques that were specifically designed for SVM classification.
Secondly, we consider five wrapper PS techniques: three genetic approaches, RMHC and FRPS. These methods have in common that they use the training accuracy after 1NN classification to evaluate entire candidate subsets of instances. We adjust these techniques for SVMs by basing the training accuracy on SVM classification.
Finally, we study the effect of fuzzy rough and genetic approaches to FS on SVM. We verify if QR and FS-SSGA can improve SVM classification, where the fitness function in FS-SSGA is adjusted for SVMs. Additionally, we study if the simultaneous approaches from the previous chapter also perform well for SVMs after adapting the training accuracy function for SVMs.
In Section 6.1 we present the different techniques, and in Section 6.2 we evaluate the approaches.

6.1 Fuzzy Rough and Evolutionary Feature and Training Set Selection approaches

There are three groups of techniques covered in this chapter. We discuss the methods below, an overview of all methods is given in Table 6.1.

6.1.1 Filter TSS and FS techniques

The first group are filter TSS and FS techniques. These include 17 filter TSS techniques that were specifically developed to improve 1NN classification. Although these techniques were designed for 1NN classification we do want to study if these methods work well for SVMs. We also consider two filter TSS techniques, SNG and MCIS, that were developed to improve SVM classification. All these filter TSS techniques are described in Section 2.4 and can be used without any adjustment for SVM classification. The last technique that we consider is the FS QuickReduct algorithm, this filter technique is discussed in Section 5.2.1.

6.1.2 Wrapper TSS techniques

We consider five wrapper TSS techniques: CHC, GGA, SSGA, RMHC as described in Section 2.4 and the FRPS method proposed in Chapter 4. All these techniques have in common that they use the training accuracy to assess the quality of candidate subsets of instances. Consider a decision system $(U, \mathcal{A} \cup \{d\})$. When the wrapper TSS techniques are used for 1NN classification, the training accuracy corresponding to a subset of instances $S \subseteq U$ is obtained by classifying each train instance $x \in U$ using the 1NN classifier where S is the pool of candidate nearest neighbors if $x \notin S$ and $S \setminus \{x\}$ else. We adjust the wrapper TSS techniques by changing this process. In order to obtain the training accuracy corresponding to $S \subseteq U$, we build the SVM model based on the decision system $(S, \mathcal{A} \cup \{d\})$, classify each instance $x \in U$ using this model and return the corresponding training accuracy. Note that we do not follow a leave-one-out strategy as for 1NN classification. The reason is that this would require building $|S| + 1$ SVM models, which is computationally too expensive.

6.1.3 Wrapper FS techniques

We only consider one FS wrapper technique, namely FS-SSGA, described in Section 5.1. This algorithm calculates the training accuracy corresponding to candidate feature subsets B. As for TSS techniques, FS-SSGA can easily be adjusted for SVM classification by changing the way in which the training accuracy is calculated. Consider a decision system $(U, \mathcal{A} \cup \{d\})$. In order to evaluate a candidate feature subset $B \subseteq \mathcal{A}$, the SVM model is built on the decision system $(U, B \cup \{d\})$, each instance in U is classified using this model and the training accuracy is returned.

6.1.4 Simultaneous TSS and FS

Recall that we have proposed a technique called SIM-FRPS-SSGA based on fuzzy rough set theory and evolutionary algorithms for simultaneous FS and PS in Section 5.1. This technique uses the FRPS algorithm for the PS part and FS-SSGA for the FS part. Additionally, we discussed two other techniques for simultaneous FS and PS in Section 5.2. SIM-SSGA carries out simultaneous FS and PS using evolutionary algorithms, and SIM-SSGA-QR uses fuzzy rough set theory for the FS part and SSGA for the PS part.
These three methods have in common that they use the training accuracy to

assess the quality of a candidate subset of instances and features. Given a decision system $(U, \mathcal{A} \cup \{d\})$, a subset of instances $S \subseteq U$ and a subset of features $B \subseteq \mathcal{A}$ we want to assess the quality of S and B. We build the SVM model based on these instances S and features B only, and classify each instance x in U using this model. The accuracy obtained using this model is then returned as training accuracy.

6.1.5 Theoretical time complexity of the proposals

We denote the number of instances by n and the number of features by m. Assuming that training the SVM model requires $\mathcal{O}(n^3m)$ operations, the CHC, GGA, SSGA, RMHC, FS-SSGA and SIM-SSGA algorithms require $\mathcal{O}(\text{nev}n^3m)$ computations, as for each evaluation the SVM model needs to be built on the train data. The FRPS algorithm requires $\mathcal{O}(n^4m)$ computations. Note that the average time is now equal to the worst time. The QR component in SIM-SSGA-QR requires $\mathcal{O}(n^2m^3)$ operations and is carried out nev$/100$ times, which means that the time complexity of SIM-SSGA-QR equals $\mathcal{O}(\text{nev}(n^2m^3 + n^3m))$. The SIM-FRPS-SSGA algorithm trains the SVM model $\mathcal{O}(\text{nev})$ times for the FS-SSGA part and $\mathcal{O}(n)$ times for the FRPS part, resulting in a total time complexity of $\mathcal{O}((\text{nev} + n)n^3m)$.

6.2 Experimental Evaluation

We apply the algorithms to the datasets listed in Table 3.1. We follow a 10 fold cross validation scheme and report the average accuracy, Cohen's kappa, reduction in terms of instances and features, and running time of the preprocessing component over the 10 folds. For the filter TSS techniques developed for 1NN classification we use the parameter settings as proposed in [55]. The parameters of the MCIS and SNG technique are automatically determined within the algorithm and the settings for the genetic and fuzzy approaches are the same as in the previous chapter.
We use the SMO approach to train the SVM, and use Platt's scaling [126] as described in Section 2.1.2.5 after building the SVM. We use the RBF kernel with parameter $\delta = 0.01$ and set the cost parameter $C = 1$. These parameters could be tuned, but as we want to study the net effect of TSS, we fix the parameters in our work. We use the pairwise coupling setting to handle multi-class problems, except for the MCIS algorithm that can only be used in combination with the

Table 6.1: Overview of the preprocessing methods for SVMs

Method	Description
SNG	Filter TSS [27]
MCIS	Filter TSS [183]
CPruner	Filter PS [187]
IB3	Filter PS [2]
AllKNN	Filter PS [154]
MCNN	Filter PS [39]
FCNN	Filter PS [5]
CNN	Filter PS [72]
HMNEI	Filter PS [112]
FRIS	Filter PS [80]
MSS	Filter PS [8]
POP	Filter PS [134]
MOCS	Filter PS [18]
MENN	Filter PS [74]
ICF	Filter PS [17]
Reconsistent	Filter PS [107]
RNG	Filter PS [141]
RNN	Filter PS [62]
DROP3	Filter PS [169]
QR	Filter FS [30]
FRPS	Wrapper TSS technique introduced in Chapter 4 and adjusted for SVMs
GGA	Wrapper TSS technique [98, 97] adjusted for SVMs
CHC	Wrapper TSS technique [21] adjusted for SVMs
SSGA	Wrapper TSS technique [21] adjusted for SVMs
RMHC	Wrapper TSS technique [146] adjusted for SVMs
FS-SSGA	Wrapper FS technique [36]
SIM-SSGA	Wrapper technique [36] for simultaneous FS and TSS adjusted for SVMs
SIM-SSGA-QR	Wrapper technique [36] for simultaneous FS and TSS adjusted for SVMs
SIM-FRPS-SSGA	Wrapper technique for simultaneous FS and TSS introduced in Section 5.1 ajdusted for SVMs

one-vs-all strategy.

6.2.1 Filter TSS techniques

In Table 6.3 we show the average results over all datasets. No filter TSS technique can improve the classification performance of the SVM. In Figure 6.2.1 we show the accuracy of the TSS methods in function of their instance reduction, and in Figure 6.2.2 we show Cohen's kappa in function of the instance reduction.
MSS is the only filter TSS technique that maintains the accuracy and Cohen's kappa of the SVM and removes a substantial (about 40 percent on average) amount of instances. The idea of MSS is to maintain the decision borders and to remove inner points, which could explain its good performance for SVMs: as the decision boundaries do not change, the SVM model remains the same.
This result implies that MSS is more suitable for SVMs than the MCIS and SNG techniques that were specifically designed for SVMs. Recall that the goal of MCIS and SNG is to remove as many instances as possible to speed up the training of the SVM. SNG removes fewer instances than MSS and is less accurate than MSS. MCIS removes 20 percent more instances than MSS but the accuracy of the SVM is 10 percent worse than without preprocessing. Both SNG and MCIS are faster than MSS on average but the difference is small. Hence, MSS is a good and fast technique to reduce the training data and maintain the accuracy of SVMs.
The other filter TSS techniques that preserve the SVM's accuracy have low reduction rates, for instance, MOCS has more or less the same accuracy and Cohen's kappa value as the SVM without preprocessing, but removes only 11 percent of the instances. This confirms our hypothesis that removing a limited amount of instances does not change the classification by SVMs drastically: in Figure 6.2.1 and 6.2.2 we see that when less than 30 percent of the instances is removed, the classification by the SVM remains the same.

6.2.2 Wrapper TSS techniques

As can be seen from Table 6.3 and in Figure 6.2.1 and 6.2.2, the only techniques that can substantially improve both the SVM classification accuracy and Cohen's kappa are the evolutionary TSS approaches GGA and CHC. These techniques are in the upper right corner of Figure 6.2.1 and 6.2.2, they remove about 90 percent of the instances and have high accuracy and Cohen's kappa values. CHC performs better than GGA with respect to Cohen's kappa but the opposite holds

for the accuracy rate. The SSGA technique improves the accuracy rate of the SVM but has about the same value for Cohen's kappa. The SSGA algorithm removes less instances than the other wrapper TSS algorithms, which can be a reason for its lower performance. The RMHC procedure performs poorly for SVMs. This can be explained by the fact that adding or removing one instance does not have an important impact on the SVM model. Hence, the instance subset returned by RMHC is likely to be very similar to the instance subset that was randomly generated at the beginning of the execution of the RMHC algorithm: removing and adding one instance does not change the SVM model much and hence it is unlikely that the accuracy will change. There is no clear reason why GGA performs better than CHC. Recall that CHC extends GGA using incest prevention and re-initialization. Perhaps, the CHC is more limited in optimizing the best solution than GGA.
The FRPS technique is not able to improve SVM classification. A possible explanation could be that FRPS removes border instances, which are important for defining the decision borders modeled by the SVM classifier. Moreover, FRPS only removes about 25 percent of the instances on average, and we remarked earlier on that removing a small amount of instances does not influence the accuracy of the SVM to a large extent.
We use the Friedman test and the Holm post-hoc procedure to contrast the wrapper TSS techniques among each other. The value of the Friedman statistic is 85.125 for accuracy and 47.36 for Cohen's kappa. In both cases the p-value is smaller than 0.000001, meaning that significant differences are detected. In Table 6.2 we show the Friedman rankings with respect to accuracy and Cohen's kappa. In both cases, GGA gets the best ranking. GGA significantly outperforms the other wrappers with respect to accuracy. It outperforms RMHC significantly with respect to Cohen's kappa and the low adjusted p-values for the other comparisons suggest that GGA does perform better than the other wrapper TSS algorithms. The fact that GGA outperforms CHC with respect to Cohen's kappa is remarkable given the fact that CHC performs better on average. This is due to the fact that GGA has a higher value for Cohen's kappa than CHC for most datasets, but for a few datasets CHC is much better than GGA.
As the best results are obtained with GGA, we recommend to use GGA as a preprocessing mechanism for improving SVMs.

6.2.3 Comparison of GGA with filter TSS techniques

The running time required by GGA is much higher than the running time of the filter TSS techniques. In order to show that this additional running time

Table 6.2: Values of the statistics of the Friedman test and Holm post-hoc procedure contrasting the results of the TSS wrappers

	Friedman rank (acc.)	Friedman rank (κ)	Adj. p-value (acc.)	Adj. p-value (κ)
GGA	1.65	2	-	-
SSGA	2	3	0.043878	0.269058
CHC	3.1	2.65	0.000082	0.269058
FRPS	3	3	0.000078	0.269058
RMHC	4.75	4	≤ 0.000001	≤ 0.000001

is justified, we test if GGA outperforms the wrapper TSS techniques and if GGA significantly improves the SVM. The Friedman statistic that compares all algorithms among each other is 276.05 for the accuracy and 229.56 for Cohen's kappa. In both cases the p-value is smaller than 0.000001, meaning that there are significant differences between the methods in the comparison. The Friedman rankings are listed in Table 6.4, GGA gets the best rankings both for accuracy and Cohen's kappa. The adjusted p-values calculated by Holm's post-hoc procedure can be found in the last two columns of Table 6.4. GGA significantly outperforms all algorithms with respect to accuracy, but this does not hold for Cohen's kappa: there are many methods that are not significantly outperformed by GGA, like MSS and the SVM without preprocessing.

6.2.4 Combining TSS and FS

The methods that use FS deteriorate the accuracy of the SVM as can be seen in Table 6.3. As we saw in the previous chapter, QR does not improve the performance of NN classification, and hence it comes at no surprise that the SVM's performance does not improve either after QR. The FS-SSGA method does not improve the accuracy of the SVM. Note that FS-SSGA for SVMs removes about 60 percent of the features, while FS-SSGA for NN removes only 50 percent of the features. This might be due to the fact that removing features cannot improve the train accuracy much. As the fitness function also contains a factor related to reduction, small feature subsets are preferred. QR performs slightly better than FS-SSGA, but removes less than 10 percent of the features. If QR is combined with SSGA, the accuracy is improved: the accuracy rate and Cohen's kappa for SIM-SSGA-QR are comparable to the values of the SVM, but unfortunately SIM-SSGA-QR comes with a high computational cost and removes only 15 percent of the features. As FS-SSGA performs worse than QR it is to be expected that

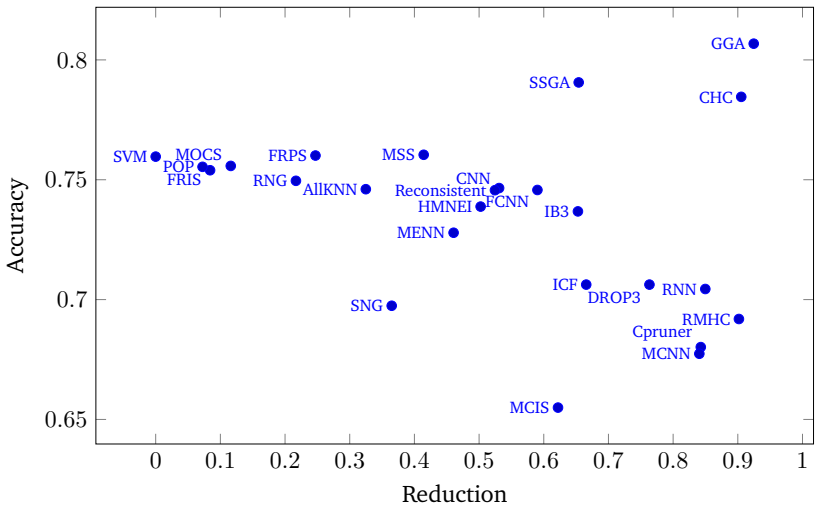


Figure 6.2.1: Accuracy of the TSS methods in function of the instance reduction

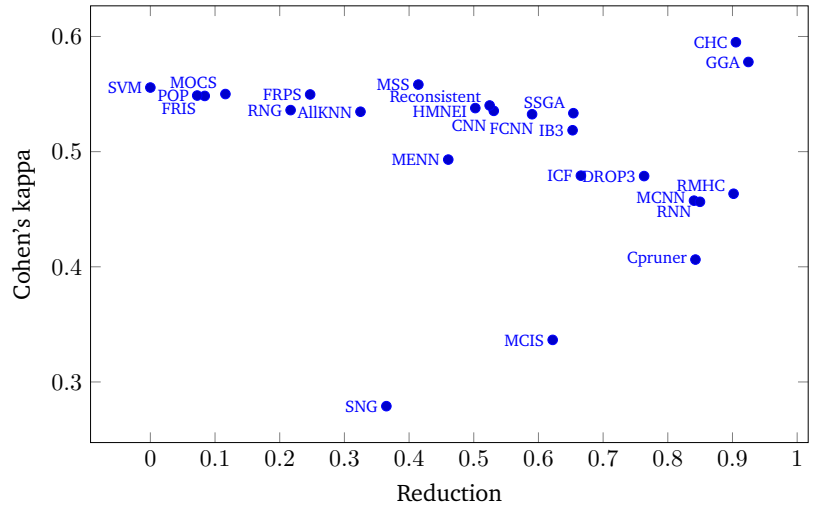


Figure 6.2.2: Cohen's kappa of the TSS methods in function of the instance reduction

combining it with FRPS does not work well either, which is confirmed by the results in Table 6.3. SIM-SSGA is one of the worst algorithms, possibly due to the fact that it removes about 67 percent of the features.

6.3 Conclusion

In this chapter we verified if the success of the techniques proposed in the previous chapters for NN translates to SVMs. We learned that only TSS techniques that remove a significant amount of instances can influence the performance of SVMs, due to the fact that the SVM model acts as a buffer between removing instances and the final classification. Methods that retain the decision borders perform well, reflected by the good performance of MSS. The FRPS approach does not improve SVM classification, probably due to the fact that it removes only a small part of the data and that it focuses on removing border instances. The RMHC technique performs poorly as it only interchanges one instance in each step, which does not affect the accuracy of the SVM to a great extent. The genetic approaches, SSGA, GGA and CHC improve the SVM's performance and the best results are obtained with GGA. Combining TSS with FS does not improve the results further, as the FS components deteriorate the SVM classification.

Table 6.3: Average results of the preprocessing techniques for SVMs over the 40 datasets.

	Acc.		κ		red_i		red_f		Time (s)
GGA	0.8068	CHC	0.5950	GGA	0.9246	SIM-SSGA	0.6756	SVM	0
SSGA	0.7906	GGA	0.5880	SIM-SSGA	0.9246	FS-SSGA	0.6716	SNG	0.12
CHC	0.7846	MSS	0.5582	SIM-SSGA-QR	0.9246	SIM-FRPS-SSGA	0.4159	MCIS	0.54
MSS	0.7604	SVM	0.5557	CHC	0.9053	SIM-SSGA-QR	0.1483	Cpruner	0.72
FRPS	0.7601	MOCS	0.5500	RMHC	0.9017	QR	0.0887	IB3	0.94
SVM	0.7596	FRPS	0.5497	RNN	0.8497	AllKNN	0	AllKNN	0.98
MOCS	0.7558	POP	0.5487	Cpruner	0.8427	CHC	0	MCNN	1.05
POP	0.7554	FRIS	0.5484	MCNN	0.8405	CNN	0	QR	1.07
FRIS	0.7540	SIM-SSGA-QR	0.5432	DROP3	0.7634	Cpruner	0	FCNN	1.18
SIM-SSGA-QR	0.7513	Reconsistent	0.5401	ICF	0.6656	DROP3	0	CNN	1.24
RNG	0.7496	HMNEI	0.5378	SSGA	0.6540	FCNN	0	HMNEI	1.43
CNN	0.7465	RNG	0.5361	IB3	0.6528	FRIS	0	FRIS	1.45
AllKNN	0.7461	CNN	0.5354	MCIS	0.6220	FRPS	0	MSS	1.48
FCNN	0.7457	AllKNN	0.5347	FCNN	0.5902	FS-SSGA	0	POP	1.51
Reconsistent	0.7456	SSGA	0.5334	CNN	0.5309	GGA	0	MOCS	1.89
HMNEI	0.7388	FCNN	0.5325	Reconsistent	0.5246	HMNEI	0	MENN	1.94
IB3	0.7368	IB3	0.5186	HMNEI	0.5024	IB3	0	ICF	2.05
SIM-FRPS-SSGA	0.7359	MENN	0.4931	MENN	0.4606	ICF	0	Reconsistent	2.65
MENN	0.7279	SIM-FRPS-SSGA	0.4887	MSS	0.4144	MCIS	0	RNG	4.00
QR	0.7245	ICF	0.4791	SNG	0.3649	MCNN	0	RNN	5.29
ICF	0.7063	DROP3	0.4788	SIM-FRPS-SSGA	0.3554	MENN	0	DROP3	10.80
DROP3	0.7063	QR	0.4688	AllKNN	0.3249	MOCS	0	FRPS	250.70
RNN	0.7044	RMHC	0.4634	FRPS	0.2470	MSS	0	FS-SSGA	1909.83
FS-SSGA	0.6983	MCNN	0.4574	RNG	0.2168	POP	0	SIM-SSGA	2047.32
SNG	0.6974	RNN	0.4564	MOCS	0.1161	Reconsistent	0	RMHC	2375.67
RMHC	0.6919	FS-SSGA	0.4226	FRIS	0.0841	RMHC	0	GGA	3384.74
Cpruner	0.6802	Cpruner	0.4063	POP	0.0725	RNG	0	SIM-SSGA-QR	3724.68
MCNN	0.6774	MCIS	0.3365	FS-SSGA	0	RNN	0	SSGA	4074.09
SIM-SSGA	0.6597	SIM-SSGA	0.3590	QR	0	SVM	0	CHC	7814.33
MCIS	0.6549	SNG	0.2790	SVM	0	SSGA	0	SIM-FRPS-SSGA	26475.11

Table 6.4: Statistics of the Friedman test and Holm post-hoc procedure comparing GGA to the other algorithms.

	Rank acc.	Rank κ	Adj. p-value acc.	Adj. p-value κ
GGA	3	6.7	-	-
MSS	7	7.075	0.008072	1.322632
SVM	7.25	7.4125	0.008072	1.322632
MOCS	8	7.825	0.002021	1.322632
FRIS	8	8.1125	0.002021	1.322632
POP	8	8.475	0.000967	1.1077
RNG	9	9.6	0.00006	0.2748
AllKNN	10	10.3	0.000004	0.118469
FCNN	11	10.025	0.000001	0.154178
CNN	11	10.5375	≤ 0.000001	0.090422
Reconsistent	11.3	10.2625	≤ 0.000001	0.118469
SNG	11.35	15.9875	≤ 0.000001	≤ 0.000001
MENN	12	12.2625	≤ 0.000001	0.00166
HMNEI	12	10.4125	≤ 0.000001	0.10564
IB3	12.8	11.6375	≤ 0.000001	0.008073
QR	13	13.5625	≤ 0.000001	0.000034
ICF	14	12.8125	≤ 0.000001	0.000358
DROP3	15	14.2875	≤ 0.000001	0.000003
RNN	16	15.5125	≤ 0.000001	≤ 0.000001
MCIS	16.5	16.7375	≤ 0.000001	≤ 0.000001
MCNN	17	15.1625	≤ 0.000001	≤ 0.000001
Cpruner	18	18.3	≤ 0.000001	≤ 0.000001

7. Speeding Up Evolutionary Prototype Selection using Ensembles

In Chapter 4 we introduced FRPS, a PS algorithm based on fuzzy rough set theory. We experimentally demonstrated that it improves evolutionary approaches to PS both with respect to classification performance as with respect to running time. This result is rather surprising, as evolutionary techniques are complex optimization techniques with a high potential to find good solutions. In this chapter we further elaborate on evolutionary approaches to PS, aiming to fully exploit their capacities.

A first observation is that in the classical PS and classification setting, only the fittest solution produced by the evolutionary PS technique is used, while many interesting suboptimal solutions are encountered during the course of the algorithm. Secondly, in order to classify instances in a certain region, the optimal solution might not be the best, and a suboptimal solution can result in better performances.

These observations and the slow running time of evolutionary PS techniques motivate us to design a strategy that uses multiple solutions generated by the evolutionary algorithm. Additionally, in order to classify a new instance, not all solutions are used but only solutions that perform well in the region of that instance.

We refer to this technique as an ensemble strategy, as multiple prototype subsets (which correspond to multiple classifiers) are used to classify new instances. Ensemble strategies have been used successfully for many aspects of classification, for instance with decision trees [151], feature selection [137] and bireducts [147]. Even though an ensemble approach seems to be highly suited for PS, the use of ensemble strategies for PS is yet unexplored.

The proposed framework can be applied for any evolutionary PS algorithm, but

we decide to use GGA as it does not use any optimization or re-initialization, and as we expect it to generate more diverse solutions than SSGA. We refer to this technique as GGA improved by ensembles (GGA-ENS). We note that this technique is not a PS technique but a classifier: the model of GGA-ENS consists of prototype subsets, which are all candidate subsets to classify new instances. The main goal of GGA-ENS is to achieve the same or better performance rates than GGA, but at a higher speed.
In Section 7.1 we introduce the framework for GGA-ENS, and in Section 7.2 we demonstrate that GGA-ENS obtains results at least as good as GGA after a much smaller number of evaluations.

7.1 GGA improved by ensembles

During the application of GGA many good and diverse prototype subsets are encountered, but only the fittest one is used for the classification of test instances. The first idea of GGA-ENS is to use multiple good prototype subsets encountered during GGA to classify test instances. Even though these prototype subsets are globally not optimal, they might have good properties and can be useful for classification. The second idea of GGA-ENS is that prototype subsets can be good to classify test instances in a particular region of the feature space, but that they are less suited to classify test instances in other regions. Using a single prototype subset neglects this idea. When using multiple prototype subsets one can choose which prototype subsets to use to classify a test instance, depending on which prototype subsets perform well in the region of that test instance.
The GGA-ENS framework encapsulates both ideas. The main steps of GGA-ENS are summarized below. Before the classification of test instances takes place, the following steps are carried out:

1. **Determine *best*** Execute GGA and store the *nbest* prototype subsets with the highest fitness values in *best*.

2. **Determine *div*** Select the *ndiv* most diverse prototype subsets among *best* and store them in *div*,

where *nbest* and *ndiv* are two user-defined parameters. Once these two steps are carried out, the classification of test instances can start. In order to classify the test instance t, the following steps are performed:

3. **Determine weights** Assign a weight $W(S)$ to each prototype subset S in *div* that expresses to what extent S is suited for classification of instances in the neighborhood of size *nk* of t.

4. **Weighted voting** Classify t using all the prototype subsets in *div* and use a weighted voting strategy to determine the final classification label of t,

where *nk* is a user-defined parameter.
Below, we discuss the separate steps in more detail.

Determine *best* The evolutionary algorithm GGA is carried out as usual, except that all prototype subsets encountered are stored in an array T during the execution. After each generation, the prototype subsets in the current population are added to T, making sure that T does not contain duplicates. When GGA completes, the *nbest* prototype subsets with the highest fitness value are stored in *best*. The parameter *nbest* should be an integer between 1 and between the number of prototype subsets encountered during the execution of GGA.

Determine *div* The prototype subsets in *best* can contain many similar prototype subsets. Therefore, it is important to select a diverse subset of prototype subsets among *best*. As the prototype subsets are used for classification, we want to ensure that the classification using the different prototype subsets is diverse. In [96], an experimental study was carried out to compare measures of diversity in classifier ensembles. This study showed that the Q_{av} function, which measures the similarity between two classifiers S_1 and S_2, achieves good results, and therefore we will use $1 - Q_{av}(S_1, S_2)$ in this work to measure the diversity between two classifiers S_1 and S_2. The similarity measure Q_{av} is given by

$$\frac{n_{00}n_{11} - n_{01}n_{10}}{n_{00}n_{11} + n_{01}n_{10}}, \tag{7.1}$$

where n_{00} is the number of instances incorrectly classified by both classifiers, n_{11} the number of instances correctly classified by both classifiers, n_{01} the number of instances incorrectly classified by S_1 but correctly classified by S_2 and n_{10} the number of instances correctly classified by S_1 and incorrectly classified by S_2. Note that if S_1 and S_2 return exactly the same classification, the value of $Q_{av}(S_1, S_2)$ equals one.
The procedure to measure the diversity between two prototype subsets is given in Algorithm 7.1. In line 4 to 9, the train instances are classified using a leave-one-out procedure with S_1 as pool of candidate nearest neighbors, and in line 10 to 15 the test instances are classified using S_2 as pool of candidate nearest neighbors. After the classification takes place, the Q_{av} measure is calculated in line 16 to 30.
In order to obtain the most diverse prototype subsets among a set of prototype

subsets, the procedure in Algorithm 7.2 is used. The set of most diverse prototype subsets *div* is initialized with the subset for which the fitness function is highest in line 2. This is the prototype subset that would be used in the traditional PS setting. From line 5 to 21, other prototype subsets are added until *div* obtains the desired size *ndiv*. The diversity between a prototype subset S and a set of prototype subsets L is defined as the sum of the diversities between S and the prototype subsets contained in L. In each iteration of the while loop, the prototype subset S for which the diversity between S and *div* is maximal is added to *div*. In line 10 and 11 the diversity between S and *div* is calculated, and if this diversity is the highest diversity obtained so far the set S is stored in S_{best}. In line 18 this subset S_{best} that adds most to the diversity is removed from the set of prototype subsets and added to *div*.

Determine weights Once the set *div* consisting of good and diverse prototype subsets is established, the classification of test instances can begin. When classifying a test instance t, we want to use prototype subsets in *div* that are good at classifying instances in the region of t. We assume that a prototype subset is good in the region of t if it classifies its nearest neighbors correctly. The outline of the process that assigns weights to the prototype subsets is listed in Algorithm 7.3. In line 2, the *nk* nearest neighbors of t are determined within X. These neighbors are classified in line 4 to 13 using a leave-one-out procedure with S as pool of candidate nearest neighbors, and each time one of the neighbors is classified correctly, the weight of the prototype subset S is raised by one minus the normalized distance d_{norm} between the nearest neighbor and the instance t in line 10 and 11. Using this approach, prototype subsets that are suited as training data to classify instances near t are associated with high weights.

Weighted voting The weights assigned to the prototype subsets are used in a weighted voting procedure. The procedure is outlined in Algorithm 7.4. The scores of all class labels are initialized 0 at the beginning of the weighted voting process. Next, the test instance t is classified using each of the prototype subsets as pool of nearest neighbors in line 5 to 7. The score of the predicted class is augmented by the weight of the corresponding prototype subset. This implies that the classifications based on prototype subsets that are well suited to classify instances in the region of t are taken more into account. In line 9 to 16 the class label with the highest score is determined, and this label is returned as the final class label of t.

The time complexity of GGA-ENS depends on the number of test instances l, the number of train instances n, the number of features m and the parameters *nk*, *ndiv* and *nbest*.
Executing the GGA algorithm requires $\mathcal{O}(\text{nev}n^2m)$ calculations. In the traditional setting, the l test instances are classified using the KNN classifier, requiring $\mathcal{O}(lnm)$ operations. In the GGA-ENS setting, first the *ndiv* most diverse prototype subsets need to be selected. Calculating the diversity between two prototype subsets requires $\mathcal{O}(n^2m)$ operations, as all instances need to be classified using the two prototype subsets as pool candidate nearest neighbors. Each time one of the *ndiv* prototype subsets is selected, the diversity between the previously added prototype subsets and the candidate prototype subsets needs to be calculated. As a result, there are $\mathcal{O}(\text{ndiv}^2\text{nbest}n^2m)$ calculations needed for this procedure. Next, in order to classify one out of the l test instances, the *nk* nearest neighbors of the test instance need to be calculated, requiring $\mathcal{O}(l\text{nk}nm)$ operations. These nearest neighbors need to be classified using the *ndiv* most diverse prototype subsets, resulting in $\mathcal{O}(l\text{nkndiv}nm)$ calculations. Finally, the test instance needs to be classified using the *ndiv* prototype subsets, which has time complexity $\mathcal{O}(l\text{ndiv}nm)$. In total, there are $\mathcal{O}(l\text{nkndiv}nm)$ calculations needed to classify all test instances. Summarized, applying GGA and KNN requires $\mathcal{O}(nm(l + \text{nev}n))$ operations, whereas GGA-ENS requires $\mathcal{O}(nm(\text{nev}n + n\text{nbestndiv}^2 + l\text{nkndiv})$ calculations. At first sight, the time complexity of GGA-ENS seems to be much larger than the time complexity of GGA, but when low values are choosen for *nk*, *ndiv*, and *nbest*, the time complexities of both strategies are the same.

7.2 Experimental Evaluation

In this section we study the performance of GGA-ENS. We present the experimental set-up in Section 7.2.1, analyze the influence of the parameters in Section 7.2.2 and compare GGA-ENS against GGA in Section 7.2.3.

7.2.1 Experimental set-up

We apply GGA-ENS and GGA in combination with the 1NN classifier to the datasets described in Table 3.1. We use a 10 fold cross validation scheme and report the average accuracy, Cohen's kappa and running time over each fold. As opposed to other chapters, the running time of GGA now includes the 1NN classification process, as we compare it against GGA-ENS which is not a preprocessing method but a classifier. The parameters of the GGA algorithm and

Algorithm 7.1 Procedure to measure the diversity between two prototype subsets

Input: Decision system $(X, \mathcal{A} \cup \{d\})$, prototype subsets S_1 and S_2
$n_{00} \leftarrow 0$, $n_{01} \leftarrow 0$, $n_{10} \leftarrow 0$, $n_{11} \leftarrow 0$
for all $x \in X$ **do**
  if $x \in S_1$ **then**
    Determine the nearest neighbor y of x in $S_1 \setminus \{x\}$
  else
    Determine the nearest neighbor y of x in S_1
  Class$(x, S_1) \leftarrow d(y)$
  if $x \in S_2$ **then**
    Determine the nearest neighbor y of x in $S_2 \setminus \{x\}$
  else
    Determine the nearest neighbor y of x in S_2
  Class$(x, S_2) \leftarrow d(y)$
  if $d(x) =$ Class(x, S_1) **then**
    if $d(x) =$ Class(x, S_2) **then**
      $n_{11} \leftarrow n_{11} + 1$
    else
      $n_{10} \leftarrow n_{01} + 1$
  else
    if $d(x) =$ Class(x, S_2) **then**
      $n_{01} \leftarrow n_{01} + 1$
    else
      $n_{00} \leftarrow n_{00} + 1$
$Q_{av}(S_1, S_2) \leftarrow \frac{n_{00}n_{11} - n_{01}n_{10}}{n_{00}n_{11} + n_{01}n_{10}}$
Output: $1 - Q_{av}(S_1, S_2)$

Algorithm 7.2 Procedure to select the most diverse prototype subsets among a set of prototype subsets

Input: A set of prototype subsets stored in T, parameter *ndiv*
div $\leftarrow S$, with S prototype subset in T with highest value $E(S)$
$T \leftarrow T \setminus S$
$ndiv \leftarrow ndiv - 1$
while $ndiv > 0$ **do**
  diversity$_{max} \leftarrow 0$
  $S_{best} \leftarrow null$
  for all $S \in T$ **do**
    $div_{current} \leftarrow 0$
    for all $P \in$ div **do**
      diversity$_{\text{current}} \leftarrow$ diversity$_{\text{current}}$ + diversity(S, P)
    if diversity$_{\text{current}} >$ diversity$_{max}$ **then**
      diversity$_{max} \leftarrow$ diversity$_{\text{current}}$
      $S_{best} \leftarrow S$
  $T \leftarrow T \setminus S_{best}$
  div $\leftarrow$ div $\cup S_{best}$
  $ndiv \leftarrow ndiv - 1$
Output: div

Algorithm 7.3 Procedure to assign weights to a prototype subset based on how well it classifies instances in the region of a test instance

Input: Decision system $(X, \mathcal{A} \cup \{d\})$, prototype subset $S \subseteq X$, test instance t, parameter *nk*
$N \leftarrow nk$ nearest neighbors of t in X
$W(S) \leftarrow 0$
for all $x \in N$ **do**
  if $x \in S$ **then**
    Determine the nearest neighbor y of x in $S \setminus x$
  else
    Determine the nearest neighbor y of x in S
  if $d(y) = d(x)$ **then**
    $W(S) \leftarrow W(S) + d_{\text{norm}}(x, y)$
Output: $W(S)$

Algorithm 7.4 Weighted voting strategy used to classify test instances

Input: Decision system $(X, \mathcal{A} \cup \{d\})$, class labels C, set of prototype subsets *div*, weight $W(S)$ associated to each prototype subset S in *div*, test instance t
for all $c \in C$ **do**
  score$(c) \leftarrow 0$
for all Prototype subsets S in *div* **do**
  Determine the nearest neighbor y of t in S
  score$(d(y)) \leftarrow$ score$(d(y)) + W(S)$
score$_{\text{best}} \leftarrow -1$
$c_{\text{best}} \leftarrow$ null
for all $c \in C$ **do**
  if score$(d(y)) >$ score$_{\text{best}}$ **then**
    score$_{\text{best}} \leftarrow$ score$(d(y))$
    $c_{\text{best}} \leftarrow c$
Output: Class label c_{best}

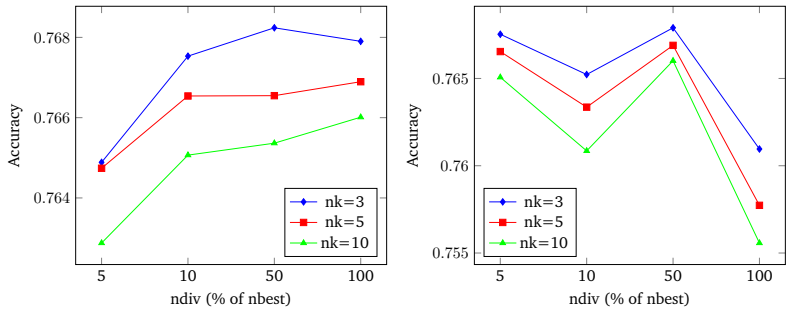
the GGA part in GGA-ENS are fixed: the size of the population is 100, the α parameter that balances the accuracy and reduction in the fitness function is 0.5, the mutation probability is 0.01 and the crossover probability 0.5. We vary the number of evaluations from 1000 to 10000 in steps of 1000.

7.2.2 Analysis of the influence of the parameters

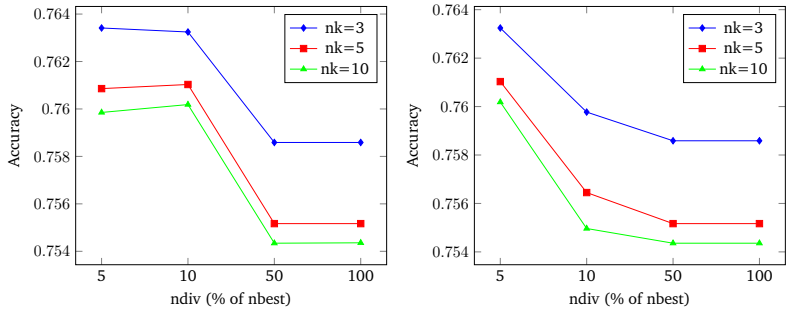
Recall that there are three user-defined parameters in the GGA-ENS algorithm:

- **nk**: the number of nearest neighbors of the test instance that is used to determine the weights of the prototype subsets. We use *nk*$= 3, 5, 10$.
- **nbest**: the number of fittest prototype subsets selected among all encountered prototype subsets during the course of GGA. We use *nbest*$= 50, 100, 500, 1000$.
- **ndiv**: the number of most diverse prototype subsets selected among the nbest prototype subsets. We set *ndiv* equal to 5%, 10%, 50% and 100% of the *nbest* parameter. For instance, if *nbest* is 1000, we consider *ndiv*$= 50, 100, 500, 1000$.

We first analyze the influence of the *nk* parameter on the performance of GGA-ENS. In Figure 7.2.1 and Figure 7.2.2 we compare the performance of GGA-ENS for *nk* equal to 3, 5 and 10 for all possible settings of *nbest* and *ndiv*. For all settings *nk*$= 3$ leads to the best performance, both with respect to accuracy



(a) nbest=50 (b) nbest=100

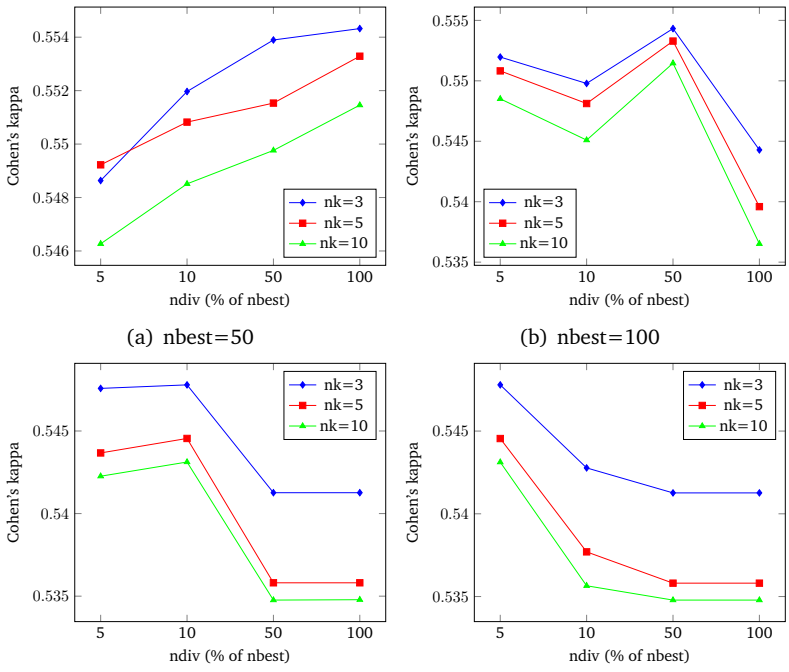


(c) nbest=500 (d) nbest=1000

Figure 7.2.1: Influence of the *nk* parameter on the classification accuracy of GGA-ENS

and Cohen's kappa. This means that, in order to determine which prototype subsets should be used to classify a test instance, one should evaluate the prototype subsets in the close neighborhood of the test instance. This also implies that our strategy, where different prototype subsets are used to classify instances in different regions, is valuable. Indeed, we see that a more general neighborhood and general decision on which prototype subsets to use results in lower performance rates.

We fix *nk* to 3 and verify which combination of *ndiv* and *nbest* results in the best performance. In Figure 7.2.3 we compare the different settings with respect to accuracy and Cohen's kappa. For both evaluation measures we observe a downwards trend for higher values of *nbest*: GGA-ENS performs better when *nbest* is lower, independent from the value of *ndiv*, except when *ndiv* is 5% of



(a) nbest=50 (b) nbest=100

(c) nbest=500 (d) nbest=1000

Figure 7.2.2: Influence of the *nk* parameter on Cohen's kappa of GGA-ENS

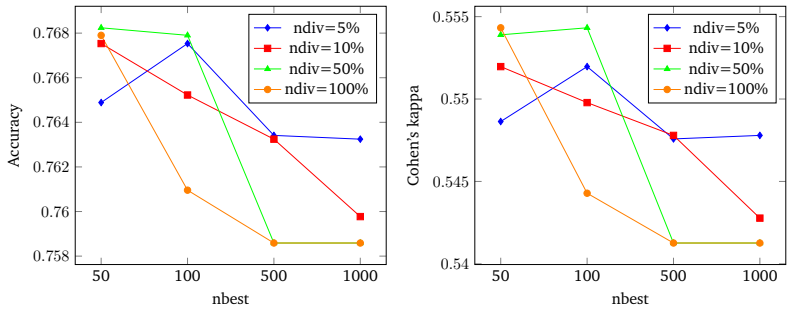


Figure 7.2.3: Influence of the *nbest* and *ndiv* parameter on the performance of GGA-ENS

nbest. This exception may be due to the fact that if *ndiv* is 5% and *nbest* 50, only two prototype subsets are used for classification. In general, we can conclude that selecting only the best prototype subsets is the most valuable strategy. When prototype subsets of lower quality are used to classify the test instance, the performance rates are lower.
The influence of the parameter *ndiv* is less clear: the results highly depend on the parameter *nbest*. A general trend is that for *nbest* larger than 100 it holds that GGA-ENS is more accurate for lower values of *ndiv*. This implies that it is indeed necessary to select the most diverse prototype subsets. However, this conclusion does not hold for *nbest* equal to 50. This can again be explained by the fact that in those cases, there is a very low number of prototype subsets selected.
There are three settings that perform equally well both with respect to accuracy and Cohen's kappa, namely *nbest*$= 50$ and *ndiv* 50% or 100% of *nbest*, and *nbest*$= 100$ and *ndiv* 50% of *nbest*. As using nbest$= 50$ and *ndiv* 50% requires the least computations, we suggest to use this parameter setting for GGA-ENS and use it in the comparison of GGA-ENS with GGA in the next subsection.

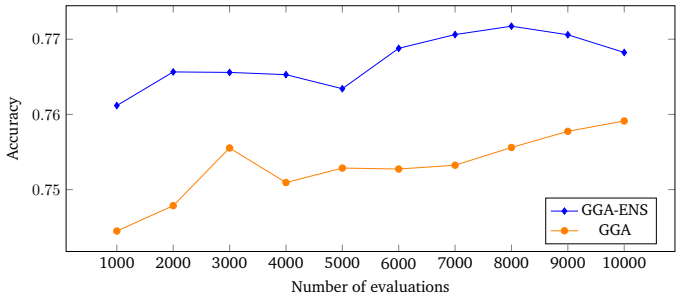
7.2.3 Comparison of GGA-ENS against GGA

In Figure 7.2.4 we compare GGA with GGA-ENS for different numbers of evaluations (*nev*). GGA-ENS outperforms GGA on average for all values of *nev*, both with respect to accuracy and Cohen's kappa. This improvement comes with only a small increase in computational complexity. Moreover, GGA-ENS achieves high accuracy rates and high values for Cohen's kappa even when *nev* is low. For

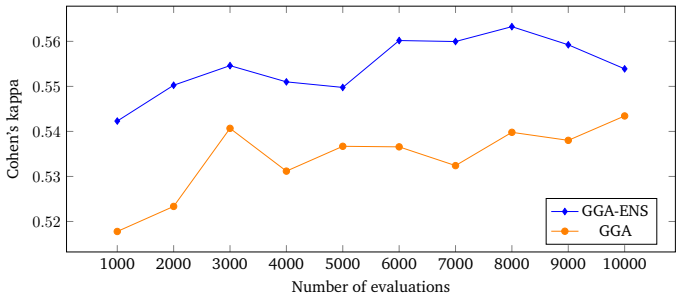
instance, the accuracy obtained by GGA-ENS after 1000 evaluations is slightly higher than the accuracy rate obtained by GGA after 10000 evaluations, and this high accuracy rate is obtained in about 10 seconds on average while GGA with 10000 evaluations requires about 70 seconds on average.
To test if GGA-ENS significantly outperforms GGA we carry out the Wilcoxon test. In Table 7.1 we show the values of the statistics of the Wilcoxon test comparing GGA-ENS with GGA for different values of *nev* with respect to accuracy and in Table 7.2 with respect to Cohen's kappa. It follows that GGA-ENS always significantly outperforms GGA when they use the same number of evaluations and additionally, GGA-ENS does never perform worse than GGA for less evaluations: for each comparison the $R+$ values are higher than $R-$. Moreover, using GGA-ENS with 7000 evaluations significantly outperforms GGA for all considered values of *nev* and the low p-values suggest that GGA-ENS with 5000 evaluations outperforms GGA for all considered values of *nev*.

7.3 Conclusion

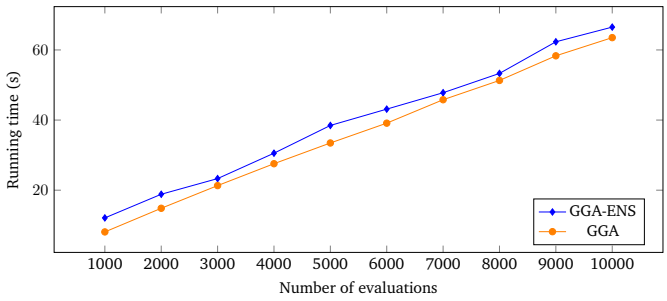
In the traditional setting of PS techniques, the capacities of evolutionary algorithms are not fully used. During the course of their execution, evolutionary PS algorithms generate many solutions, of which only the fittest one is used to classify new instances. Motivated by this observation, we designed a strategy in which multiple prototype subsets encountered during the course of an evolutionary algorithm are used to classify new instances. In this ensemble strategy, we only use the fittest prototype subsets and select the most diverse among them. Additionally, in order to classify a test instance, we aim to only use prototype subsets that are well suited for classification of instances in the neighborhood of that test instance.
This strategy can be applied for any evolutionary PS algorithm, in this chapter we focused on GGA. Our strategy, called GGA-ENS, outperforms the traditional setting where PS is followed by NN classification. Moreover, we observed that GGA-ENS needs less evaluations than GGA followed by NN in order to obtain similar and even better performance rates. This demonstrates the use of ensemble techniques for evolutionary PS.



(a)



(b)



(c)

Figure 7.2.4: Comparison of GGA with GGA-ENS with respect to accuracy, Cohen's kappa and running time.

Table 7.1: Values of the statistics of the Wilcoxon test comparing the GGA-ENS method against GGA with respect to accuracy for a range of number of evaluations

		GGA-ENS nev=1000	GGA-ENS nev=2000	GGA-ENS nev=3000	GGA-ENS nev=4000	GGA-ENS nev=5000	GGA-ENS nev=6000	GGA-ENS nev=7000	GGA-ENS nev=8000	GGA-ENS nev=9000	GGA-ENS nev=10000
GGA	R+	**634.0**	**677.0**	**715.0**	**717.0**	**686.0**	**681.0**	**730.5**	**739.0**	**694.5**	**739.0**
nev=1000	R-	**146.0**	**103.0**	**105.0**	**103.0**	**134.0**	**99.0**	**89.5**	**81.0**	**85.5**	**81.0**
	P-value	**0.000645**	**0.00006**	**0.00004**	**0.000036**	**0.000202**	**0.000047**	**0.000045**	**0.000009**	**0.00002**	**0.000009**
GGA	R+	**590.0**	**677.0**	**737.0**	**766.0**	**752.0**	**734.0**	**747.0**	**726.0**	**736.0**	**744.0**
nev=2000	R-	**230.0**	**143.0**	**83.0**	**54.0**	**68.0**	**86.0**	**73.0**	**54.0**	**44.0**	**76.0**
	P-value	**0.01526**	**0.000316**	**0.000011**	**0.000002**	**0.000004**	**0.000013**	**0.000006**	**0.000003**	**0.000001**	**0.000007**
GGA	R+	482.0	490.0	**594.0**	**594.0**	**580.0**	**623.0**	**655.5**	**658.0**	**646.0**	**652.0**
nev=3000	R-	338.0	290.0	**226.0**	**186.0**	**200.0**	**197.0**	**164.5**	**122.0**	**134.0**	**168.0**
	P-value	0.329811	0.160773	**0.013141**	**0.00432**	**0.007851**	**0.004108**	**0.000926**	**0.000179**	**0.000344**	**0.001116**
GGA	R+	504.0	**561.0**	**634.0**	**654.0**	**743.0**	**695.0**	**744.0**	**751.5**	**736.0**	**724.0**
nev=4000	R-	276.0	**259.0**	**186.0**	**126.0**	**77.0**	**85.0**	**76.0**	**68.5**	**84.0**	**96.0**
	P-value	0.110077	**0.041715**	**0.002548**	**0.000223**	**0.000007**	**0.00002**	**0.000007**	**0.000011**	**0.000011**	**0.000024**
GGA	R+	511.0	533.0	**577.0**	**586.0**	**647.0**	**698.5**	**757.0**	**731.0**	**675.0**	**678.0**
nev=5000	R-	309.0	287.0	**243.0**	**194.0**	**133.0**	**121.5**	**63.0**	**89.0**	**108.0**	**102.0**
	P-value	0.172476	0.096914	**0.024359**	**0.006104**	**0.000326**	**0.000297**	**0.000003**	**0.000013**	**0.000232**	**0.000055**
GGA	R+	508.0	492.0	**611.0**	**640.0**	**660.0**	**702.5**	**744.0**	**735.0**	**739.0**	**703.0**
nev=6000	R-	312.0	328.0	**209.0**	**140.0**	**120.0**	**117.5**	**36.0**	**45.0**	**81.0**	**117.0**
	P-value	0.185514	0.26747	**0.00676**	**0.000473**	**0.00016**	**0.000237**	**0.000001**	**0.000001**	**0.000009**	**0.00008**
GGA	R+	485.0	504.0	**590.0**	**625.0**	**628.5**	**661.5**	**712.0**	**771.5**	**736.0**	**695.0**
nev=7000	R-	295.0	316.0	**230.0**	**195.0**	**191.5**	**158.5**	**68.0**	**48.5**	**84.0**	**85.0**
	P-value	0.18263	0.204012	**0.01526**	**0.003714**	**0.003192**	**0.000692**	**0.000007**	**0.000003**	**0.000011**	**0.00002**
GGA	R+	432.0	485.5	550.0	**550.0**	**595.0**	**602.0**	**655.0**	**752.0**	**741.0**	**657.0**
nev=8000	R-	348.0	334.5	270.0	**230.0**	**185.0**	**178.0**	**125.0**	**68.0**	**79.0**	**123.0**
	P-value	0.553123	0.306186	0.058534	**0.025105**	**0.004134**	**0.003023**	**0.000211**	**0.000004**	**0.000008**	**0.000189**
GGA	R+	422.0	397.0	532.5	581.5	553.5	**608.5**	**642.0**	**707.5**	**692.0**	**668.0**
nev=9000	R-	358.0	423.0	287.5	238.5	266.5	**211.5**	**138.0**	**112.5**	**128.0**	**152.0**
	P-value	0.650162	1	0.225918	0.054224	0.129649	**0.007377**	**0.000426**	**0.000178**	**0.000146**	**0.000512**
GGA	R+	425.0	399.0	479.0	512.0	514.0	551.0	**634.5**	**616.0**	**599.0**	**617.0**
nev=10000	R-	395.0	381.0	301.0	268.0	266.0	269.0	**185.5**	**164.0**	**181.0**	**163.0**
	P-value	0.834963	0.894531	0.211675	0.087361	0.081783	0.057179	**0.007142**	**0.001573**	**0.00346**	**0.0015**

Table 7.2: Values of the statistics of the Wilcoxon test comparing the GGA-ENS method against GGA with respect to Cohen's kappa for a range of number of evaluations

		GGA-ENS nev=1000	GGA-ENS nev=2000	GGA-ENS nev=3000	GGA-ENS nev=4000	GGA-ENS nev=5000	GGA-ENS nev=6000	GGA-ENS nev=7000	GGA-ENS nev=8000	GGA-ENS nev=9000	GGA-ENS nev=10000
GGA	R+	**621.0**	**667.0**	**692.0**	**695.0**	**672.0**	**664.0**	**704.0**	**723.0**	**692.0**	**695.0**
nev=1000	R-	**159.0**	**153.0**	**128.0**	**125.0**	**148.0**	**116.0**	**116.0**	**97.0**	**128.0**	**125.0**
	P-value	**0.001235**	**0.000538**	**0.000146**	**0.000124**	**0.000418**	**0.000128**	**0.000075**	**0.000025**	**0.000146**	**0.000124**
GGA	R+	**574.0**	**624.0**	**721.0**	**748.0**	**718.0**	**717.0**	**733.0**	**764.0**	**706.0**	**690.0**
nev=2000	R-	**246.0**	**196.0**	**99.0**	**72.0**	**102.0**	**103.0**	**87.0**	**56.0**	**74.0**	**130.0**
	P-value	**0.027029**	**0.003937**	**0.000028**	**0.000005**	**0.000034**	**0.000036**	**0.000014**	**0.000002**	**0.00001**	**0.000163**
GGA	R+	446.0	471.0	**585.0**	**577.0**	**567.0**	**601.0**	**638.0**	**693.0**	**617.0**	**601.0**
nev=3000	R-	374.0	349.0	**235.0**	**243.0**	**253.0**	**219.0**	**182.0**	**127.0**	**163.0**	**219.0**
	P-value	0.623704	0.408441	**0.018327**	**0.024359**	**0.034259**	**0.010053**	**0.002131**	**0.000139**	**0.0015**	**0.010053**
GGA	R+	490.0	537.0	**600.0**	**648.0**	**670.0**	**669.0**	**694.0**	**712.0**	**700.0**	**643.0**
nev=4000	R-	290.0	283.0	**220.0**	**172.0**	**150.0**	**111.0**	**126.0**	**108.0**	**120.0**	**177.0**
	P-value	0.160773	0.086572	**0.01045**	**0.001347**	**0.000463**	**0.000096**	**0.000131**	**0.000048**	**0.000094**	**0.001698**
GGA	R+	489.0	481.0	**557.0**	**560.0**	**573.0**	**637.0**	**695.0**	**680.0**	**636.0**	**609.0**
nev=5000	R-	331.0	339.0	**263.0**	**220.0**	**207.0**	**143.0**	**125.0**	**140.0**	**144.0**	**171.0**
	P-value	0.285258	0.336526	**0.047413**	**0.017344**	**0.010311**	**0.000553**	**0.000124**	**0.000277**	**0.000582**	**0.00219**
GGA	R+	489.0	444.0	**586.0**	**597.0**	**628.0**	**651.0**	**745.0**	**751.0**	**680.0**	**627.0**
nev=6000	R-	331.0	376.0	**234.0**	**223.0**	**192.0**	**169.0**	**75.0**	**69.0**	**140.0**	**193.0**
	P-value	0.285258	0.642845	**0.017673**	**0.011727**	**0.003315**	**0.00117**	**0.000006**	**0.000004**	**0.000277**	**0.003461**
GGA	R+	525.0	512.0	**606.0**	**594.0**	**609.0**	**651.0**	**730.0**	**772.0**	**738.0**	**709.0**
nev=7000	R-	295.0	308.0	**214.0**	**226.0**	**211.0**	**169.0**	**90.0**	**48.0**	**82.0**	**111.0**
	P-value	0.12055	0.168286	**0.008261**	**0.013141**	**0.007328**	**0.00117**	**0.000016**	**0.000001**	**0.00001**	**0.000057**
GGA	R+	431.0	439.0	534.0	522.0	**561.0**	**550.0**	**655.0**	**733.0**	**707.0**	**638.0**
nev=8000	R-	349.0	381.0	286.0	298.0	**259.0**	**230.0**	**165.0**	**87.0**	**113.0**	**182.0**
	P-value	0.5625	0.691723	0.09424	0.130497	**0.041715**	**0.025105**	**0.000967**	**0.000014**	**0.000064**	**0.002131**
GGA	R+	447.0	389.0	504.0	505.0	522.0	**582.0**	**626.0**	**663.0**	**669.0**	**637.0**
nev=9000	R-	373.0	391.0	276.0	275.0	258.0	**238.0**	**154.0**	**117.0**	**151.0**	**183.0**
	P-value	0.614227	1	0.110077	0.107005	0.064452	**0.020415**	**0.000966**	**0.000135**	**0.000487**	**0.002229**
GGA	R+	419.0	446.0	530.0	500.0	526.0	533.0	**602.0**	**639.0**	**573.0**	**586.0**
nev=10000	R-	401.0	374.0	290.0	320.0	294.0	287.0	**218.0**	**181.0**	**207.0**	**234.0**
	P-value	0.898392	0.623704	0.105302	0.223818	0.117369	0.096914	**0.009669**	**0.002037**	**0.010445**	**0.017673**

8. Fuzzy Rough Prototype Selection for Imbalanced Classification

Imbalanced classification, i.e. classification of data where at least one class is underrepresented, is an important and often occurring problem in data mining. There are many examples of real-world imbalanced problems like fraud detection [48], oil spill detection [94], medical problems [114] and many more.
Blindly applying traditional data mining techniques to imbalanced problems can cause problems, as the interest of most data mining techniques is to improve the general classification accuracy of the data. As a result, it can happen that no instance in the minority class is classified correctly. However, the minority class is often the class of most interest. Consider for instance a medical diagnosis problem. The main interest of researchers is to detect patients with the disease, and falsely diagnosing patients with a disease is not as weighty as failing to recognize that a patient has this disease.
This definitely holds for PS techniques. When applying traditional PS techniques to imbalanced problems, it is very likely that many minority instances are removed and that primarily majority instances are retained. The FRPS algorithm works very well for classical classification problems, but when applying FRPS to an imbalanced problem, we expect that many minority instances will be classified incorrectly. Indeed, candidate prototype subsets with few minority instances will be associated with high train accuracies, as these prototype subsets classify most instances as majority. Consider for instance a problem with 10 minority instances and 90 majority instances. Any prototype subset with only majority instances is associated with a train accuracy of 90 percent, as all majority instances in the train data are classified correctly.
The goal of this chapter is twofold. On the one hand, we want to improve FRPS such that it can handle imbalanced data. We propose a fuzzy rough PS tech-

nique for imbalanced data, called Fuzzy Rough Imbalanced Prototype Selection (FRIPS). FRIPS proceeds like FRPS, except that candidate prototype subsets are evaluated based on the Adjusted Area Under the Curve (AAUC, see Section 2.1.3.2) rather than on accuracy, and as a result both classes are taken equally into account when evaluating candidate subsets.
Secondly, we want to study if FRPS can be used for imbalanced classification in combination with the Synthetic Minority Oversampling TEchnique (SMOTE, [23]). SMOTE is a very successful and widely used technique to preprocess imbalanced data that introduces artificial minority instances to enlarge the minority class. This research path is motivated by the good performance of state-of-the-art preprocessing techniques were editing is applied after balancing the data using SMOTE.
We first discuss the state-of-the-art in preprocessing techniques for imbalanced data in Section 8.1. Then, we present FRIPS in Section 8.2 and we evaluate our proposals in Section 8.3.

8.1 State-of-the-art in imbalanced classification: SMOTE and its improvements

As multi-class problems can easily be decomposed into two-class problems using e.g. pairwise coupling [52, 73, 123], we assume that we are given a two-class dataset consisting of the minority or positive class and the majority or negative class.
Many techniques handle imbalanced classification by preprocessing the imbalanced data rather than focusing on the classification directly. Examples of such techniques are undersampling techniques [8, 57, 95, 101, 155, 179, 180] that remove majority instances to balance the data, while oversampling techniques [9, 11, 19, 23, 69] balance the data by adding artificial or duplicating existing minority instances. Hybrid techniques [131, 148] first balance the data and then apply some editing technique to the new balanced dataset. In the following we only list the most accurate and widely used preprocessing techniques for preprocessing imbalanced data. We also note that besides preprocessing, there are many other techniques to handle imbalanced data, like cost-sensitive learning [70, 113, 150, 152, 158, 164] and boosting algorithms [24, 53, 58, 135, 143, 165].

- **SMOTE**
  An important technique is the SMOTE oversampling technique. SMOTE

has been widely used in research on imbalanced datasets, and is considered as the de facto method in oversampling techniques. As SMOTE generates new instances rather than duplicating existing ones, it is particularly useful in combination with 1NN classification.
The outline of SMOTE is given in Algorithm 8.1. We present a specific version of SMOTE that ensures that the data is balanced after applying SMOTE, the more general SMOTE algorithm that we do not present here allows the user to choose how many artificial instances are generated.
In line 3 the K nearest neighbors of all minority instances are calculated and stored for further use. These nearest neighbors can be majority or minority instances. The number of instances to be generated is G, this number is initialized in line 5 such that after G positive instances are generated, the dataset is balanced. In line 7 to 18 the instances are generated. Creating one positive instance depends on one original minority instance x_i. The index i iterates over these instances such that each minority instance is used an equal amount of times. In step 8 one among the K nearest neighbors of x_i is chosen randomly, we call this neighbor y. In line 9 to 12 the new instance is generated, by constructing the attribute values individually. The new value of attribute a lies between $a(x_i)$ and $a(y)$ and is determined by a random number r between 0 and 1. If r is 1, the attribute value of the new instance x equals $a(y)$ and if r is 0, $a(x)$ equals $a(x_i)$. Each new instance is stored in S and this set is added to the decision system.

Many preprocessing techniques are based on SMOTE. SMOTE-Edited Nearest Neighbors (SMOTE-ENN, [11]), SMOTE-Tomek Links (SMOTE-TL, [11]) and SMOTE-Fuzzy Rough Set Theory (SMOTE-FRST, [132]) are three techniques that first apply SMOTE and subsequently carry out a cleaning phase where instances are removed from the balanced data:

- **SMOTE-ENN**
  First, SMOTE is applied to balance the data. Next, ENN is applied to the new dataset, that is, instances x that are classified incorrectly using the other instances as pool of candidate nearest neighbors are marked for removal and all marked instances are removed at the end.

- **SMOTE-TL**
  A Tomek-link is a pair of instances x and y that belong to different classes, and for which there is no element z such that the distance between x and y is larger than the distance between x and z or the distance between y and z, in other words, x and y are closer to each other than to any other instance.

Algorithm 8.1 The SMOTE algorithm

Input: Decision system $(U, \mathcal{A} \cup \{d\})$ with p positive instances $P = \{x_1, \ldots, x_p\} \subseteq U$ and attributes $\mathcal{A} = \{a_1, \ldots, a_m\}$, parameter K
for all $i = 1, \ldots, p$ **do**
  $nn_i \leftarrow K$ nearest neighbors of x_i
$i \leftarrow 1$
$G \leftarrow |U| - 2p$
$S \leftarrow \{\}$
while $G > 0$ **do**
  Select random nearest neighbor y in nn_i
  for all $i = 1, \ldots, m$ **do**
    $r \leftarrow$ random number between 0 and 1
    $v_i \leftarrow a_i(x_i) + r(a_i(y) - a_i(x_i))$
  Create new positive instance x with attribute values $\langle v_1, \ldots, v_m \rangle$
  $S \leftarrow S \cup \{x\}$
  if $i < p$ **then**
    $i \leftarrow i + 1$
  else
    $i = 1$
  $G \leftarrow G - 1$
Output: Decision system $(U \cup S, \mathcal{A} \cup \{d\})$

The SMOTE-TL algorithm first applies SMOTE to balance the data, and then identifies all Tomek-links in the new data. For each Tomek-link the negative instance is removed.

- **SMOTE-FRST**
  SMOTE-FRST is a technique where SMOTE and cleaning are repeated until a certain number of iterations is reached, or until the dataset is balanced. Each time SMOTE is applied, the membership values of the instances to the fuzzy rough positive region are calculated. Majority instances or artificial minority instances generated by SMOTE for which this membership value exceeds a certain threshold γ are retained in the instance set, others are removed.

The SMOTE-Border Line technique (SMOTE-BL, [69]), SMOTE-Safe-Level technique (SMOTE-SL, [11]) and Majority Weighted Minority Oversampling TEchnique (MWMOTE, [9]) improve the way in which the artificial minority instances are generated:

- **SMOTE-BL**
  The goal of SMOTE-BL is to generate artificial minority instances near the borders between the two classes. There are two versions of the algorithm. SMOTE-BL1 defines a minority instance *in danger* as a minority instance for which the majority of its nearest neighbors belongs to the negative class, and for which at least one nearest neighbor belongs to the positive class. These minority instances in danger are supposed to be on the border between the positive and negative class, hence these instances are used to generate instances. Moreover, new instances are only introduced between these minority instances in danger and their nearest neighbors from the positive class.
  SMOTE-BL2 introduces new instances between the minority instances in danger and all nearest neighbors, independent from their class. An additional modification of SMOTE-BL2 is that the random factor r for interpolation is 0.5, and as a result the introduced instances are closer to the minority instances.

- **SMOTE-SL**
  SMOTE-SL defines the safe-level sl of a positive instance as the ratio of nearest neighbors of the positive class and the nearest neighbors of the negative class. If the safe-level is larger than 1, i.e., there are more neighbors of the positive class, the random number r is chosen between 0 and $1/sl$, that is, the new instance is placed close to the original positive instance. If the safe-level is smaller than 1, r is chosen between $1 - sl$ and

1, which means that the instance is placed closer to the neighbor. If sl is one, r is chosen between 0 and 1 as usual. Apart from this modification, SMOTE-SL proceeds like SMOTE.

- **MWMOTE**
  The MWMOTE technique is based on clustering. First, the minority class is clustered using K-means, where the number of clusters is determined such that the minimum difference between two clusters does not exceed a certain threshold τ. This threshold is the average distance between all instances multiplied with a parameter C_p. Then, the filtered minority instance set is calculated, it contains minority instances that have no minority instances among their K_1 nearest neighbors. The borderline majority set is the set of majority instances that are included in the K_2 nearest neighbors of any instance in the filtered minority instance set. Finally, the informative minority set S_{imin} is defined, it contains the minority instances that are included in the K_3 nearest neighbor set of a borderline majority instance. For each instance $x \in S_{imin}$, a weight is calculated that expresses how important x is. This weight is based on the distance between x and the nearest majority instances, the underlying idea is that instances close to the decision boundaries, minority instances in a sparse cluster and minority instances near a dense majority cluster should get higher weights. These weights are transformed into probabilities, and for each new instance that needs to be generated, a minority instance x is picked among S_{imin} according to that probability. Next, it is determined in which cluster x lies, and an artificial instance is generated on the line between x and a random other instance in that cluster.

Finally, the Selective Preprocessing technique (SPIDER, [148]) does not make use of any SMOTE component:

- **SPIDER**
  SPIDER labels instances as *safe* if they are classified correctly using the 3NN classifier with the remaining instances as pool of candidate nearest neighbors. Minority instances that are unsafe are duplicated m times, where m is the number of instances among the 3 nearest neighbors that are safe and belong to the majority class. Additionally, SPIDER removes unsafe majority instances.

8.2 Fuzzy Rough Imbalanced Prototype Selection

In this section we present our PS method for imbalanced data. The idea of FRIPS is that instances are ordered according to a fuzzy rough quality measure, and that a threshold τ, that says if an instance should be removed or retained, is automatically determined.
FRIPS proceeds like FRPS, except that candidate subsets of prototypes are not evaluated using the leave-one-out-training accuracy but by the leave-one-out training AAUC. Recall that, given the confusion matrix C, the AAUC is calculated as follows:

$$AAUC(C) = \frac{1 + \frac{C_{11}}{C_{11}+C_{12}} - \frac{C_{21}}{C_{21}+C_{22}}}{2}. \tag{8.1}$$

We use the AAUC measure as this is a discrete translation of the widely used AUC measure for probabilistic classifiers. If we would want to use the FRIPS procedure for probabilistic classifiers like for instance SVMS, the AAUC measure can be replaced by the traditional AUC measure. In order to attain a high AAUC value, one needs a balance between a high number of correctly classified minority instance and not too many instances falsely classified as minority. Note that if all instances are classified to one class (either the majority class or either the minority class), the value for AAUC is 0.5. If all instances are classified correctly, the value of AAUC is 1 and if no instance is classified correctly, the value of AAUC is 0.
This AAUC function is used in the FRIPS algorithm, listed in Algorithm 8.2. In line 2, the quality measure Q is applied to each instance $x \in U$ and these thresholds are ordered in line 3. We use the quality measure that was optimal for the FRPS algoritm in Chapter 4. In line 4 to 9 the AAUC that corresponds to the first threshold τ_1 (i.e. that corresponds to the entire training set) is calculated. This is done by constructing the confusion matrix and then using the formula for AAUC in Equation (8.1). Next, all remaining thresholds are considered in line 10 to 21. In line 11 the instances for which the quality is lower than the current threshold are removed, and in line 12 to 16 the new confusion matrix is calculated. Note that only the nearest neighbors of instances that were removed in the last step are updated in this process. In line 17 to 21 the best AAUC found so far is updated, and finally in line 22, the median among the optimal thresholds is chosen and the subset that corresponds to this threshold is returned in line 23.

The FRIPS algorithm can be applied directly to the imbalanced data but it might also be interesting to use FRIPS in combination with an oversampling

Algorithm 8.2 The FRIPS algorithm

Input: Decision system $(U, \mathcal{A} \cup \{d\})$
$T \leftarrow \{Q(x) | x \in U\}$
Order thresholds: $T = \{\tau_1, \tau_2, \ldots, \tau_t\}$ such that $\tau_1 < \tau_2 < \ldots < \tau_t$
Initialize confusion matrix $C = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$
for all $x \in X$ **do**
    $NN(x) =$ nearest neighbor of x in $U \setminus \{x\}$
    $C(d(x), d(NN)) \leftarrow C(d(x), d(NN)) + 1$
$bestAAUC \leftarrow AAUC(C)$
$best_\tau = \{\tau_1\}$
for all $i = 2, \ldots, t$ **do**
    $S = \{x \in U | Q(x) \geq \tau_i\}$
    Initialize confusion matrix $C = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$
    for all $x \in X$ **do**
        if $Q(NN(x)) = \tau_{i-1}$ **then**
            $NN(x) =$ nearest neighbor of x in $S \setminus \{x\}$
        $C(d(x), d(NN)) \leftarrow C(d(x), d(NN)) + 1$
    if $bestAAUC = AAUC(C)$ **then**
        $best_\tau = best_\tau \cup \{\tau_i\}$
    else if $bestAAUC < AAUC(C)$ **then**
        $best_\tau = \{\tau_i\}$
        $bestAAUC \leftarrow AAUC(C)$
$\tau =$ median$(best_\tau)$
Return $S = \{x \in U | Q(x) \geq \tau\}$

technique. Moreover, the original FRPS technique can also be used to improve SMOTE. The idea of combining SMOTE with FRIPS or FRPS relies on two ideas. The first idea is that the original train data can be noisy, and that the instances generated by SMOTE rely on this noisy data and can hence be of low quality. By cleaning the data using FRIPS before applying SMOTE this problem can be avoided. The second motivation is that SMOTE can generate artificial minority instances that can be of low quality, and using FRPS to clean the data after applying SMOTE can handle this shortcoming.
Summarized, we consider the following settings:

- FRIPS: apply the FRIPS algorithm to the original imbalanced data
- FRIPS-SMOTE: first clean the data using the FRIPS algorithm and then apply the SMOTE algorithm to balance the data
- FRIPS-SMOTE-FRPS: After balancing the cleaned data, the FRPS algorithm is applied to clean the balanced data
- SMOTE-FRPS: first balance the data using SMOTE and then clean the data using the FRPS algorithm

8.2.1 Theoretical time complexity of the proposals

Denote by n the number of instances and m the number of features. The time complexity of FRIPS is the same as the time complexity of FRPS, as calculating the training AAUC instead of calculating the training accuracy does not require any significant additional calculations. That is, the average time complexity is $\mathcal{O}(n^2 log(n)m)$ and the worst time complexity is $\mathcal{O}(n^3 m)$.
Applying SMOTE requires $\mathcal{O}(n^2 m)$ operations, the dominant cost is related to calculating the nearest neighbors of all minority instances. When applying FRPS after SMOTE, the theoretical time complexity is the same as FRPS, but one should take into account that in practice the FRPS procedure might take longer as the data can have doubled in size.

8.3 Experimental Evaluation

In this Section we study which of our proposals performs best and if our proposals outperform the state-of-the-art methods described in Section 8.1. In Section 8.3.1 we explain the set-up of our experiments, in Section 8.3.2 we compare our proposals against each other and in Section 8.3.3 we compare our best proposal against the state-of-the-art.

8.3.1 Experimental Set-up

We consider 50 imbalanced datasets that were retrieved from the KEEL dataset repository. The properties of those datasets are listed in Table 8.1. The Imbalance Ratio (IR), defined as the size of the majority class divided by the size of the minority class, expresses how imbalanced the data is. As we are working with imbalanced data, we choose to use a 5 fold cross validation procedure. Using 10 fold cross validation would cause problems as some folds can contain almost none or no minority instances. All algorithms that are based on the SMOTE algorithm are repeated 5 times as SMOTE has a random component.
The parameters of SMOTE are fixed as follows: we use $K = 5$ and follow the procedure in Algorithm 8.1, which means that after applying SMOTE the data is balanced. The algorithms that are based on SMOTE like SMOTE-TL, SMOTE-BL, SMOTE-ENN, SMOTE-SL and our proposals all use the same parameters for the SMOTE part. SMOTE-ENN requires an additional parameter for the number of nearest neighbors in the ENN part, we fix this number to 3. The parameter γ in the SMOTE-FRST is fixed to 0.8 and the maximal number of iterations is 10. The parameters of the MWMOTE algorithm are chosen as proposed by the authors: C_p is 3, $K_1 = 5$, $K_2 = 3$ and K_3 is half of the size of the minority class. The number of instances generated is twice the size of the minority class.
For each method and dataset we calculate the average AAUC over all folds and the average running time. As usual, the running time only covers the preprocessing part and not the classification part.

8.3.2 Proposal Selection

The first question we study in this experimental evaluation is if FRIPS performs better than the original FRPS approach. In Figure 8.3.1 we compare FRIPS against FRPS. We consider 5 groups of 10 datasets with different ranges of IR and show the average AAUC for each group. The average AAUC of FRIPS is better than the AAUC of FRPS for each group. The differences are larger for datasets with average IR, which shows that the imbalance ratio of the data has indeed an important influence on the performance of FRPS, and that FRIPS is able to alleviate the imbalance problem.
Next, we study which of the proposed settings performs best. In Figure 8.3.2 we compare the proposals against each other, with 1NN and SMOTE as baselines. It is clear that SMOTE improves 1NN classification for all IR ranges. The FRIPS algorithm improves 1NN classification except for datasets with a very high IR. The reason for this can be that when there are very few minority instances, they

Table 8.1: Datasets used in the experimental evaluation

	# inst.	# feat.	IR		# inst.	# feat.	IR
glass1	214	0	1.82	ecoli-0-1-4-7_vs_2-3-5-6	336	7	10.59
ecoli-0_vs_1	220	7	1.86	led7digit-0-2-4-5-6-7-8-9_vs_1	443	7	10.97
iris0	150	4	2	glass-0-6_vs_5	108	9	11
glass0	214	9	2.06	ecoli-0-1_vs_5	240	6	11
haberman	306	3	2.78	glass-0-1-4-6_vs_2	205	9	11.06
glass-0-1-2-3_vs_4-5-6	214	9	3.2	glass2	214	9	11.59
ecoli1	336	7	3.36	ecoli-0-1-4-7_vs_5-6	332	6	12.28
appendicitisimb	106	7	4.05	cleveland-0_vs_4	177	13	12.62
new-thyroid2	215	5	5.14	ecoli-0-1-4-6_vs_5	280	6	13
new-thyroid1	215	5	5.14	movement_libras-1	336	7	13
ecoli2	336	7	5.46	yeast-1_vs_7	459	7	14.3
glass6	214	8	6.38	glass4	214	9	15.47
ecoli3	336	7	8.6	ecoli4	336	7	15.8
ecoli-0-3-4_vs_5	200	7	9	page-blocks-1-3_vs_4	472	10	15.86
ecoli-0-6-7_vs_3-5	222	7	9.09	glass-0-1-6_vs_5	184	9	19.44
ecoli-0-2-3-4_vs_5	202	7	9.1	shuttle-c2-vs-c4	129	9	20.5
glass-0-1-5_vs_2	172	9	9.12	cleveland-4	106	7	21.85
ecoli-0-4-6_vs_5	203	6	9.15	shuttle-6_vs_2-3	230	9	22
ecoli-0-1_vs_2-3-5	244	7	9.17	ionosphere-bred_vs_g	235	33	22.5
ecoli-0-2-6-7_vs_3-5	224	7	9.18	glass5	214	9	22.78
glass-0-4_vs_5	92	9	9.22	wdbc-mredb_vs_b	372	30	23.8
ecoli-0-3-4-6_vs_5	205	7	9.25	winequality-white-9_vs_4	168	11	32.6
ecoli-0-3-4-7_vs_5-6	257	7	9.28	ionosphere-bredb_vs_g	231	33	37.5
ecoli-0-6-7_vs_5	220	6	10	ecoli-0-1-3-7_vs_2-6	281	7	39.14
glass-0-1-6_vs_2	192	9	10.29	wdbc-mred_vs_b	365	30	44.63

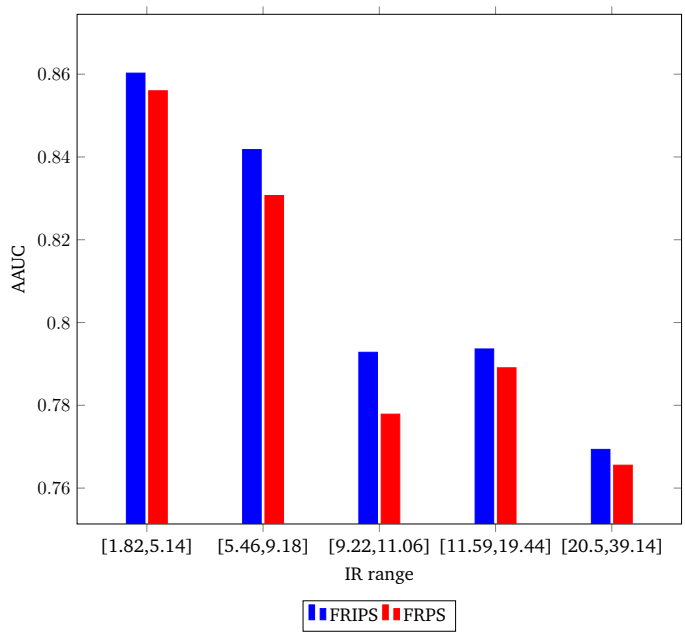


Figure 8.3.1: Comparison of the AAUC of FRIPS and FRPS for different IR ranges.

have the lowest values for the quality measure defined by FRIPS. As a result, most of the candidate subsets mainly contain majority instances and it is likely that such a prototype subset is returned by FRIPS. When FRIPS is applied before using SMOTE, the AAUC is lower than the AAUC of SMOTE, except when the IR is in [5.46,9.18]. It seems that FRIPS removes instances that are useful for SMOTE oversampling. This also explains why FRIPS-SMOTE-FRPS does not perform well for the higher IR ranges.
However, when we first apply SMOTE and then clean the data using FRPS, the result is much better: the AAUC is higher than when only SMOTE is applied. The differences are larger for the intermediate IR ranges, the result is less clear for very low and very high IR rates. This means that FRPS is able to select the best instances from the balanced dataset. In the remainder of this section we will compare our proposal, SMOTE-FRPS, against the state-of-the-art.

8.3.3 Comparison against the baselines

In Table 8.2 we compare our proposal SMOTE-FRPS against the baselines with respect to AAUC and running time. SMOTE-FRPS is the most accurate method on average, the closest competitors are SMOTE-ENN and SMOTE-TL. Note that these techniques both apply SMOTE, followed by a cleaning phase. This means that FRPS is more suited to clean the dataset after applying SMOTE than ENN or Tomek Links. The good performance of SMOTE-FRPS unfortunately comes with a higher computational cost. Note that after applying SMOTE, the dataset is much larger than the original dataset, and as a result the running time required for FRPS is remarkably longer than the reported running time of FRPS in Chapter 4, even though the datasets used in Chapter 4 are bigger.
In Figure 8.3.3 we compare SMOTE-FRPS with its closest comptetitors. Again, we make a distinction between the different IR ranges. SMOTE-FRPS always outperforms SMOTE-TL, but the differences are smaller for highly imbalanced datasets. Recall that SMOTE-TL only removes majority instances, which is apparently good for highly imbalanced datasets. SMOTE-FRPS outperforms SMOTE-ENN except when the IR is rather low.
To test if SMOTE-FRPS significantly outperforms SMOTE, SMOTE-ENN and SMOTE-TL, we carry out the Friedman test. The value of the Friedman statistic is 14.742 and the associated p-value is 0.002051, which means that significant differences are detected amongst these four methods. In Table 8.3 we show the Friedman rankings in the second column. SMOTE-FRPS gets the best ranking, followed by SMOTE-TL, SMOTE-ENN and SMOTE respectively. The adjusted p-values returned by Holm's post-hoc procedure are listed in the last column, the

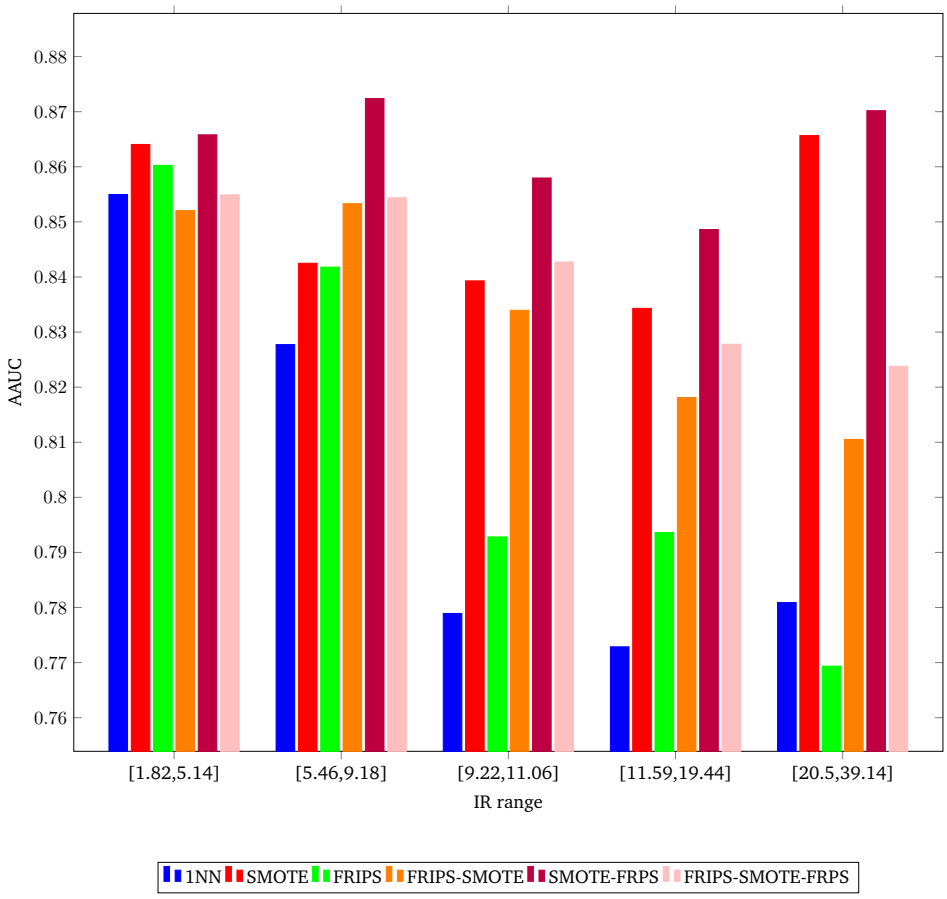


Figure 8.3.2: Comparison of the AAUC of our proposals for different IR ranges.

Table 8.2: Average AAUC and running time of our proposal SMOTE-FRPS and the baselines over the 50 datasets.

	1NN	SMOTE	SMOTE-FRST	SMOTE-ENN	MWMOTE	SPIDER
AAUC	**0.8031**	**0.8492**	**0.7904**	**0.8530**	**0.8249**	**0.8165**
Time (in s.)	0	**4.74**	**240.93**	**3.09**	**3.33**	**0.34**

	SMOTE-SL	SMOTE-BL1	SMOTE-BL2	SMOTE-TL	SMOTE-FRPS
AAUC	**0.8031**	**0.8259**	**0.8245**	**0.8552**	**0.8630**
Time (in s.)	**2.88**	**4.38**	**3.14**	**2.74**	**39.06**

Table 8.3: Friedman ranks comparing SMOTE-FRPS and the baselines among each other and adjusted p-values obtained by the Holm post-hoc procedure comparing SMOTE-FRPS against the baselines

	Friedman rank	Adj. P-value
SMOTE-FRPS	2.01	-
SMOTE-TL	2.47	0.096486
SMOTE-ENN	2.52	0.096486
SMOTE	3	0.000378

low adjusted p-values suggest that SMOTE-FRPS outperforms all other methods.

8.4 Conclusion

Motivated by the good performance of FRPS for classical datasets, we studied in this chapter to what extent FRPS can be used to enhance imbalanced classification. On the one hand, we propose FRIPS, a PS method for imbalanced data which improves FRPS. On the other hand, we study the effect of using FRIPS and FRPS in combination with SMOTE, an oversampling technique for imbalanced data. We conclude that FRIPS improves FRPS for imbalanced data, showing that FRPS does indeed suffer from the imbalance in data. Secondly, we see that using SMOTE to balance the data and subsequently applying FRPS to clean the data improves the state-of-the-art preprocessing methods for imbalanced data. This shows that SMOTE introduces low-quality instances in the data, and that fuzzy rough set theory is a good tool to clean the data after applying SMOTE.

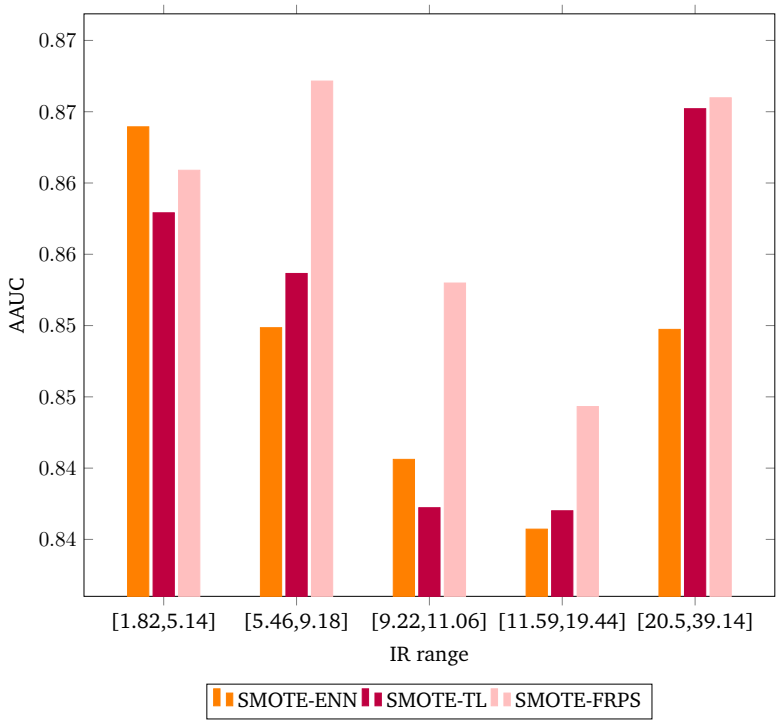


Figure 8.3.3: Comparison of our proposal SMOTE-FRPS against the best baselines for different IR ranges.

9. Improving Nearest Neighbor Classification with Fuzzy Rough Sets

In previous chapters we have used fuzzy rough set theory to preprocess the data. Through instance selection, we improved the quality of the data such that classifiers performed better. In this chapter we want to tackle the classification problem directly, more specifically, we want to improve KNN classification using fuzzy rough set theory.

We start from the Fuzzy Nearest Neighbor (FNN, [87]) classifier proposed by Keller et al.. An important drawback of the KNN algorithm is that is that it considers each of the K neighbors equally important during the classification of a target instance t, independent of the neighbor's distance to t. To overcome this problem, Keller et al. suggested to introduce fuzzy set theory into the classical KNN decision rule. By means of an indiscernibility relation, instances can now partially belong to the set of nearest neighbors and are weighted accordingly.

There have been several attempts in the literature to improve FNN by means of fuzzy rough set theory. In [142], the author aims to improve FNN using a so-called fuzzy rough ownership function. However, it was noted in [79] that the main ideas of fuzzy rough set theory, the lower and upper approximation, are not used in this work.

Later on, two other techniques that aim to improve FNN by means of fuzzy rough set theory were introduced. The first method, called Fuzzy Rough Nearest Neighbor (FRNN, [79, 81]) measures the extent to which the nearest neighbors belong to the fuzzy lower and upper approximation of a certain class to predict the class of the target instance t. The Vaguely Quantified Nearest Neighbor (VQNN, [79, 81]) method takes the FRNN approach one step further by using vaguely quantified fuzzy rough sets instead of traditional fuzzy rough sets. These methods are recalled in Section 9.1.

Although the idea of using fuzzy rough set theory to improve FNN is very valuable, neither FRNN nor VQNN are very useful methods. We will show in Section 9.1.2 that FRNN only takes into account one instance, and in Section 9.1.3 that VQNN coincides with FNN if the same indiscernibility relation is used. In Section 9.2.1 we propose a new method, called Fuzzy Rough Positive Region based Nearest Neighbor (POSNN, [161]) that aims to improve FNN by using the membership to the fuzzy positive region of instances to measure to what extent they are typical for their class. This method is extended in Section 9.2.2, where three aspects of the nearest neighbors are taken into account: the quality, based on fuzzy rough set theory, the frequency with which its class is occurring among the nearest neighbors, and the indiscernibility to the target instance. The QFSNN algorithm tunes the weights associated to each of these aspects. Finally, in Section 9.3, we compare our proposals against each other and FNN.

9.1 State-of-the-art: Presentation and Analysis

In this section we first recall Keller et al.'s FNN method and experimentally show that it outperforms KNN. Next, we present two attempts to improve FNN by means of fuzzy rough set theory and prove that their application potential is limited.

9.1.1 Fuzzy Nearest Neighbor (FNN)

In [87], Keller et al. noted that a major drawback of KNN is that each of the selected nearest neighbors (we denote the set of K nearest neighbors by NN) is considered equally important when assigning a class to the target instance t. This makes KNN highly dependent on the parameter K. Consider the example in Figure 9.1.1. The target instance is represented by a star and the classification problem is whether this instance should be classified to the circle class or to the diamond class. When $K = 10$, there are 4 nearest neighbors of the target instance that belong to the circle class and 6 that belong to the diamond class. This means that the target instance will be classified to the diamond class, although it is clear from the figure that the circle class should be preferred. When K is lowered, the classification changes. For instance, when $K = 7$, there are 4 nearest neighbors belonging to the circle class and 3 belonging to the diamond class, which means that the target instance is classified to the circle class. This shows that KNN highly relies on K.

The fact that we intuitively prefer the circle class above the diamond class comes

from the fact that the nearest neighbors of the circle class are closer to the target instance. In [87], Keller et al. took this fact into account by incorporating the indiscernibility $R(x,t)$ between the target instance t and the nearest neighbors $x \in NN$, defined as follows:

$$R(x,t) = \frac{1}{d_{eucl}(x,t)^{\frac{2}{m-1}}}, \tag{9.1}$$

where d_{eucl} is the Euclidean distance. The parameter m determines to what extent the distance is weighted when calculating each neighbor's contribution. We use $m = 2$ as suggested in [87]. When a nearest neighbor $x \in NN$ is close to the target instance t, the value $R(x,t)$ will be high. In the example in Figure 9.1.1, the circle instances will have a higher value than the diamond instances. Before introducing Keller et al.'s FNN algorithm, we also need the notion of class membership function. For each class C, the class membership function $memb_C$ reflects for each $x \in NN$ the extent $memb_C(x)$ to which x belongs to that class. We use two definitions of $memb_C$. The crisp membership function $memb_C^{crisp}(x)$ is 1 if the instance x belongs to class C and 0 otherwise. A more involved approach was proposed by Keller et al. in [87]. They defined the membership function $memb_C^{gradual}$, where

$$memb_C^{gradual}(x) = \begin{cases} 0.51 + 0.49\frac{n_c}{K} & \text{if } x \text{ is in class } C \\ 0.49\frac{n_c}{K} & \text{else} \end{cases} \tag{9.2}$$

for each $x \in NN$ and class C. Here, n_C is the number of instances in NN that belong to class C. Using this definition, the class membership $memb_C^{gradual}(x)$ will be more than 0.51 if x belongs to class C and higher if many of the elements in NN belong to C. On the other hand, $memb_C^{gradual}(x)$ is smaller than 0.49 if x does not belong to class C and is smaller if fewer instances belong to class C. The class membership function $memb_C$ and the indiscernibility relation R are the key ingredients of the FNN algorithm. For each class C, the following value is calculated:

$$\frac{\sum_{x \in NN} R(x,t) memb_C(x)}{\sum_{x \in NN} R(x,t)} \tag{9.3}$$

and the class with the highest value is returned as the class of the target instance t. This means that a target instance t will be classified to class C if there are many nearest neighbors of that class ánd if these neighbors are similar to the target instance.

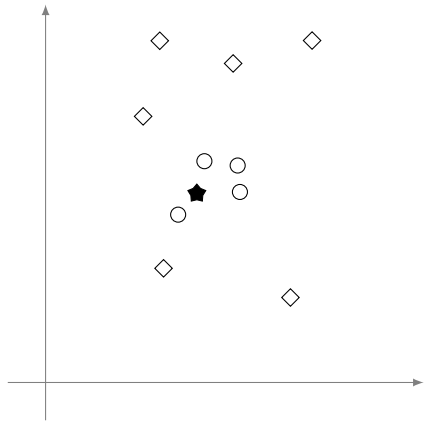


Figure 9.1.1: KNN with $K = 10$ assigns the target instance (represented by a star) to the diamond class although it is clear that it should be assigned to the circle class.

To show that FNN outperforms KNN, we evaluate both algorithms on the 40 datasets described in Table 3.1. We use a 10 fold cross validation strategy. The average test accuracy over all datasets is plotted in Figure 9.1.2 for several values of K.
From Figure 9.1.2 it is clear that FNN improves KNN for all values of K. The higher K, the clearer the difference. For higher values of K, the crisp class membership function performs clearly better than the gradual one. In the remainder of this chapter we will only use the crisp class membership function, therefore from now on we shortly denote $memb_C^{crisp}(x)$ by $C(x)$.

9.1.2 Fuzzy Rough Nearest Neighbor (FRNN)

As fuzzy rough set theory was proven to be successful to model uncertain and inconsistent data, the authors in [79, 81] tried to incorporate fuzzy rough set theory in the FNN technique. The FRNN algorithm first looks up the K nearest neighbors (NN) of the target instance t and then classifies the target instance to the class C for which the sum

$$(R \downarrow C)(t) + (R \uparrow C)(t) \tag{9.4}$$

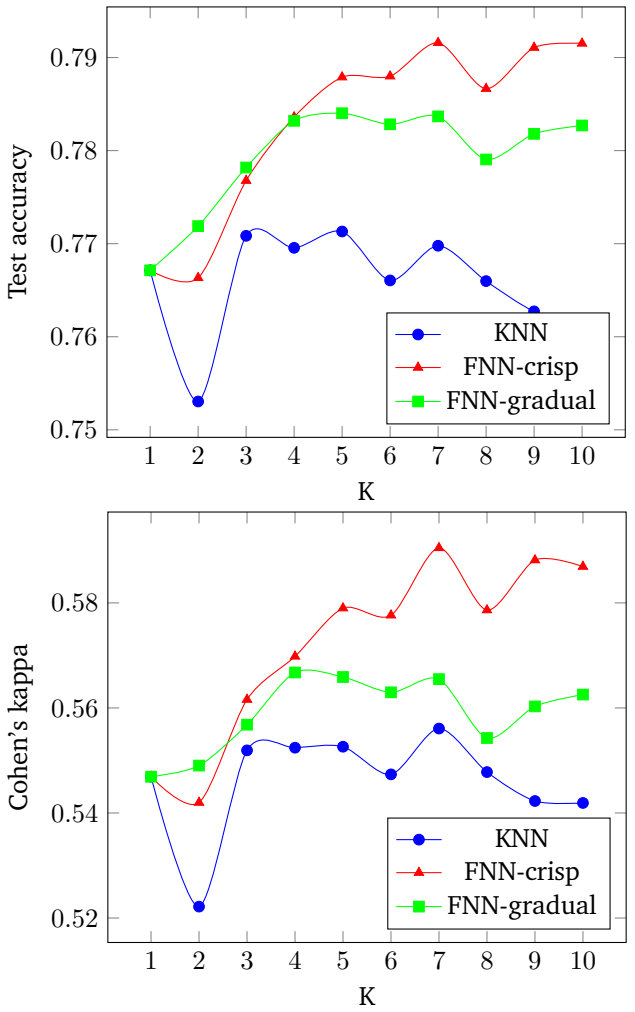


Figure 9.1.2: Comparison between the crisp and gradual class membership variants of the FNN algorithm and KNN for different values of K for the datasets in Table 3.1.

is maximal, with R a fuzzy indiscernibility function. The upper and lower approximations only take into account the instances of NN:

$$(R \downarrow C)(t) = \min_{x \in NN} \mathcal{I}(R(x,t), C(x)) \tag{9.5}$$

$$(R \uparrow C)(t) = \max_{x \in NN} \mathcal{T}(R(x,t), C(x)). \tag{9.6}$$

The idea behind this algorithm is that the lower and the upper approximation of a class C, calculated by means of the NN of the target instance, provide good clues to predict the membership of the target instance to that class. In particular, if $(R \downarrow C)(t)$ is high, it reflects that many of t's neighbors belong to C, while a high value of $(R \uparrow C)(t)$ means that there exist neighbors that belong to C. Unfortunately, when we look in more detail at this method, we see that only one instance influences the classification of the target instance, as proven in the next theorem:

Theorem 9.1.1. *FRNN assigns a target instance t to the class of the instance $x \in NN$ for which $R(x,t)$ is maximal.*

Proof. First we note that the theorem obviously holds if all instances belong to the same class. In the remainder of the proof we assume that there are at least two classes. We first rewrite the lower approximation of the target instance t:

$$\begin{aligned}(R \downarrow C)(t) &= \min_{x \in NN} \mathcal{I}(R(x,t), C(x)) \\ &= \min_{x \in NN, C(x)=0} \mathcal{I}(R(x,t), 0).\end{aligned}$$

The upper approximation can be written as follows:

$$\begin{aligned}(R \uparrow C)(t) &= \max_{x \in NN} \mathcal{T}(R(x,t), C(x)) \\ &= \max_{x \in NN, C(x)=1} \mathcal{T}(R(x,t), 1) \\ &= \max_{x \in NN, C(x)=1} R(x,t).\end{aligned}$$

The target instance t is assigned to the class C for which the following expression is maximal:

$$\min_{x \in NN, C(x)=0} \mathcal{I}(R(x,t), 0) + \max_{x \in NN, C(x)=1} R(x,t).$$

Now suppose that $x \in NN$ is an instance that maximizes $R(x,t)$, and denote the class of x by D, that is, $D(x) = 1$. Then for each other class C,

$$\max_{x \in NN, C(x)=1} R(x,t) \leq \max_{x \in NN, D(x)=1} R(x,t).$$

At the same time,

$$\min_{x \in NN, C(x)=0} \mathcal{I}(R(x,t),0) \leq \min_{x \in NN, D(x)=0} \mathcal{I}(R(x,t),0),$$

which means that the target instance t is assigned to class D, which is by definition the class of the instance $x \in NN$ for which $R(x,t)$ is maximal. □

This theorem shows that FRNN can be reduced to a very simple algorithm that does not take into account fuzzy rough concepts. It also means that if the indiscernibility relation R is the complement of the distance d, applying FRNN is the same as applying the KNN algorithm with $K = 1$. For instance, if d is normalized such that for each $x, y \in U$ it holds that $d(x,y) \in [0,1]$, and if $R(x,y) = 1 - d(x,y)$ for all instances, the instance y for which $R(x,t)$ is maximal will be included in the NN set, and the class of y will be assigned to the target instance t. When applying KNN with $K = 1$, the nearest neighbor of t is y, which means the class of y will be assigned to t.
We note that this only holds for classification problems, and that FRNN is a useful method for regression problems, as showed in [81] where also regression problems are considered.

9.1.3 Vaguely Quantified Nearest Neighbor (VQNN)

In [79, 81], the authors suggested to improve the FRNN approach using the VQRS model instead of the traditional fuzzy rough set model. Again, after determining the K nearest neighbor set NN, the target instance t is assigned to the class C for which

$$(R \downarrow_{Q_u} C)(t) + (R \uparrow_{Q_l} C)(t) \tag{9.7}$$

is maximal. Only the instances in NN are considered for calculating the VQRS lower and upper approximations:

$$(R \downarrow_{Q_u} C)(t) = Q_u\left(\frac{\sum_{x \in NN} \min(R(x,t), C(x))}{\sum_{x \in NN} R(x,t)}\right) \tag{9.8}$$

$$(R \uparrow_{Q_l} C)(t) = Q_l\left(\frac{\sum_{x \in NN} \min(R(x,t), C(x))}{\sum_{x \in NN} R(x,t)}\right), \tag{9.9}$$

with Q_u and Q_l two fuzzy quantifiers that represent *most* and *some* respectively. The rationale behind using the VQRS model is that noise should be handled better. However, the next theorem shows that VQNN is the same classifier as FNN provided they use the same indiscernibility measure.

Theorem 9.1.2. *Assume VQNN and FNN use the same indiscernibility relation. Then FNN and VQNN classify each target instance to the same class.*

Proof. Consider a target instance t. As the fuzzy quantifiers Q_u and Q_l are increasing, VQNN maximizes the following sum (the denominators in the arguments of Q_u and Q_l are equal):

$$\sum_{x \in NN} \min(R(x,t), C(x)).$$

As $C(x)$ only takes values in $\{0,1\}$, this sum can be rewritten as:

$$\sum_{x \in NN} R(x,t)C(x),$$

which is exactly the sum FNN is maximizing over. □

9.2 Improving FNN using fuzzy rough set theory

In this section we introduce our two proposals, POSNN and QFSNN, that aim to improve FNN using fuzzy rough set theory. Next, we calculate the theoretical time complexity of POSNN and QFSNN and compare it against the theoretical time complexity of KNN and FNN.

9.2.1 Fuzzy Rough Positive Region based Nearest Neighbor Classification (POSNN)

Motivated by the fact that neither FRNN nor VQNN are able to meaningfully improve FNN by incorporating fuzzy rough set theory, we present a new fuzzy rough nearest neighbor algorithm in this section. The strength of the FNN algorithm is that it takes the frequency of the classes among the nearest neighbors into account on the one hand and the indiscernibility between these nearest neighbors and the target instance on the other hand. We build upon this idea and consider an additional property of the neighbors, namely their quality or

typicality. Our original POSNN proposal in [161] bases the quality of instances on the traditional fuzzy rough positive region, but as the fuzzy rough positive region based on the OWA model resulted in better accuracies in Chapter 4, we use the same quality measure here.
The POSNN algorithm proceeds as follows to classify the target instance t. First, the K nearest neighbors NN are calculated. Then, the target instance t is classified to the class C for which

$$\frac{\sum_{x \in NN} R(x,t)C(x)POS(x)}{\sum_{x \in NN} R(x,t)} \tag{9.10}$$

is maximal. The quality of a neighbor x is expressed by means of its membership value to the OWA fuzzy rough positive region, where inverse additive weights are used and where the indiscernibilities with respect to individual attributes are aggregated using the average. This quality measure was shown to be optimal for the FRPS algorithm and we expect it to work well for POSNN as well. The POSNN algorithm is similar to FNN, the difference is that high-quality instances will be taken more into account.

9.2.2 Quality, Frequency and Similarity based Fuzzy Nearest Neighbor Classification (QFSNN)

The POSNN algorithm takes three aspects of the nearest neighbors into account: the distance to the target instance, its quality and indirectly it also takes into account how many times its class occurs. These aspects are aggregated by multiplying them. The QFSNN algorithm takes a different approach to combine these three aspects. We first summarize the three properties of the nearest neighbors $x \in NN$ that are taken into account :

- The frequency $F(x)$ of the class x among the instances in NN. That is, if the class of x is C and there are c instances in NN with class C, the frequency $F(x)$ is given by $\frac{c}{|NN|}$. Note that this aspect is not explicitly used in the POSNN algorithm, but it is clear that when $F(x)$ is higher, there are more non-zero arguments in the sum in Equation (9.10) and the class of x has a higher probability of being chosen.
- The indiscernibility $S(x) = R(x,t)$ between x and t, using the indiscernibility measure proposed by Keller in [87].
- The quality $Q(x)$ of x , determined as for POSNN and the FRPS algorithm.

These three aspects of the nearest neighbors are all important for the classification. The POSNN algorithm combines these measures by multiplying them. An alternative approach is to combine the different aspects in a weighted evaluation measure E as follows:

$$\forall x \in X : E(x) = w_Q Q(x) + w_F F(x) + w_S S(x). \tag{9.11}$$

To classify a target instance t, the QFSNN algorithm calculates the nearest neighbors NN of t and returns the class of the nearest neighbor $x \in NN$ for which $E(x)$ is maximal.
An important question is which weights w_Q, w_F and w_S to use. The QFSNN algorithm automatically tunes these weights: many weight triplets are considered, and the QFSNN procedure is carried out with each of these weight triplets on the training data. The triplet corresponding to the highest training accuracy is used to classify the test data.
The detailed outline of the algorithm is given in Algorithm 9.1. The algorithm consists of two main parts: in the first part (line 4 to 18), the weights are tuned, while in line 19 to 25, the target instance t is classified based on these tuned weights. The parameter r is a natural number that determines how fine the weights are tuned. When r is larger, more weight combinations are tried. The weight triplets considered in line 4 are all possible combinations of natural numbers between 1 and r. For each of these weight triplets, all train instances x are classified in line 6 to 19 and the training accuracy is calculated. This classification is carried out by looking up the nearest neighbors of the training instance at hand in the training data, where x itself is of course excluded. In line 8 to 15, the neighbor N_{max} for which the evaluation measure E is maximal is searched. The instance x is classified to the class of N_{max}, so if this class corresponds with the actual class of the training instance, the accuracy is raised by one. In line 20 to 23, the QFSNN algorithm keeps track of the triplet of weights W_{opt} for which the training accuracy is maximal. Once the optimal weights are found, the classification of the target instance can start. First the K nearest neighbors NN are looked up, and then the neighbor $x \in NN$ for which the value $E(x)$ is maximal is determined. The class returned for t is the class of this particular neighbor.

9.2.3 Theoretical time complexity of our proposals

Denote by n the number of train instances, by l the number of test instances and by m the number of features. In order to classify all test instances, both KNN

Algorithm 9.1 The QFSNN algorithm

Input: Train data X, target instance t, parameter K, parameter r
$W_{opt} = \langle w_Q^{opt}, w_F^{opt}, w_S^{opt} \rangle \leftarrow \langle 0, 0, 0 \rangle$
$Acc_{opt} \leftarrow -1$
for all $\langle w_Q, w_F, w_S \rangle \in \{(i, j, k) | i, j, k \in \{1, \ldots, r\}\}$ **do**
  $Acc \leftarrow 0$
  for all $x \in X$ **do**
    $NN \leftarrow K$ nearest neighbors of x in $X \setminus \{x\}$
    $E_{max} \leftarrow 0$
    $N_{max} \leftarrow null$
    for all $y \in NN$ **do**
      if $E(y) = w_Q Q(y) + w_F F(y) + w_S S(y) \geq E_{max}$ **then**
        $E_{max} \leftarrow E(y)$
        $N_{max} \leftarrow y$
    if $C(x) = C(N_{max})$ **then**
      $Acc \leftarrow Acc + 1$
  if $Acc \geq Acc_{opt}$ **then**
    $Acc_{opt} \leftarrow Acc$
    $W_{opt} \leftarrow \langle w_Q, w_F, w_S \rangle$
$NN \leftarrow K$ nearest neighbors of t in X
$E_{max} \leftarrow 0$
$N_{max} \leftarrow null$
for all $x \in NN$ **do**
  if $E(x) = w_Q^{opt} Q(x) + w_F^{opt} F(x) + w_S^{opt} S(x) \geq E_{max}$ **then**
    $E_{max} \leftarrow E(x)$
    $N_{max} \leftarrow x$
Output: $C(N_{max})$

and FNN require $\mathcal{O}(Klmn)$ calculations: for each test instance the K nearest neighbors need to be calculated. POSNN first calculates the nearest neighbors of a test instance, and then calculates the OWA fuzzy rough positive region of all nearest neighbors and the distance to the nearest neighbors. This accounts for $\mathcal{O}(lKn(m + log(n)))$ operations. The QFSNN algorithm first goes through a training phase, requiring $\mathcal{O}(r^3n^2m)$ operations. Then, in order to classify all test instances, $\mathcal{O}(lKn(m + log(n)))$ operations are needed.

9.3 Experimental Evaluation

In this section we verify if POSNN or QFSNN can improve FNN classification. We apply the algorithms to the datasets described in Table 3.1 and use a 10 fold cross validation scheme. We report the average accuracy, Cohen's kappa and running time over the 10 folds. We vary the parameter K between 1 and 10. The parameter m in the FNN algorithm is set to $m = 2$ as suggested in [87], and we use the crisp class membership function as this resulted in the best performance in Section 9.1.1. The quality measure used in the POSNN algorithm is the quality measure that leads to the best performance of the FRPS algorithm. That is, we use the OWA fuzzy rough positive region to model the quality and use the indiscernibility function that uses the average to aggregate the indiscernibility with respect to individual attributes. The OWA weights are the inverse additive weights. The same quality measure is used in the QFSNN algorithm. We evaluate the QFSNN algorithm with range r equal to 10 until 100 in steps of 10.
Before we compare POSNN and QFSNN against FNN, we study the effect of the range r on the performance of the QFSNN algorithm. In Table 9.1 we show the average accuracy of QFSNN over the 40 datasets, and in Table 9.2 the average Cohen's kappa. For each value of K we indicate the best result in bold. A general trend is that for lower values of K, the optimal range is lower. When K is larger than 5, the optimal range is higher, and higher values for r would probably result in an even better performance. This might be due to the fact that for low values of K, tuning the weights has less influence as fewer neighbors are considered. The optimal combination of weights is reached fast for low values of K and overfitting occurs when the weights are tuned in too much detail. For higher values of K, the weights have more influence and there is more space to tune them.
However, if r is too large, the QFSNN algorithm would require too much time and therefore we do not consider higher values of r than 100. As the results for high K are of more interest (note that for lower values of K the algorithms

tend more to the KNN algorithm), we consider $r = 100$ in the remainder of the analysis.
In Figure 9.3.1 we show the average results over all datasets for K ranging from 2 to 10. We do not show the results for $K = 1$ in the plots as all algorithms equal KNN for $K = 1$. Both QFSNN and POSNN perform better than FNN on average with respect to accuracy and Cohen's kappa, for all considered values of K. The differences between QFSNN and POSNN are small for accuracy, but QFSNN outperforms POSNN with respect to Cohen's kappa for all values of K. The differences are small however, and QFSNN is computationally more demanding than POSNN.
To test if POSNN or QFSNN outperforms FNN significantly, we carry out the Wilcoxon test. The results of this test are shown in Table 9.3 for accuracy and in Table 9.4 for Cohen's kappa. It is clear that QFSNN does not improve FNN significantly, neither with respect to accuracy, nor with respect to Cohen's kappa. POSNN does not improve FNN significantly with respect to accuracy, but it does improve FNN significantly with respect to Cohen's kappa for most values of K, except for $K = 3$, $K = 5$ and $K = 7$. These results are remarkable, as QFSNN performs better on average with respect to Cohen's kappa. It seems that for some datasets, QFSNN improves FNN much more than POSNN, but that POSNN improves FNN for more datasets than QFSNN.
We conclude that the extra computational cost that QFSNN requires is not reconcilable with its performance. On the other hand, the POSNN method is computationally less demanding and outperforms the FNN algorithm with respect to Cohen's kappa for the majority of values of K. The reason why QFSNN is less suited can be due to overfitting, another reason why we are not able to significantly improve upon FNN can be that we reach some limit to what is possible when tackling the KNN problem directly. That is, when none of the K nearest neighbors of a test instance have the same class as the test instance, we cannot classify it correctly, no matter what weighting strategy is used to improve the classification.

9.4 Conclusion

In the previous chapters we improved classifiers by preprocessing the train data that they use. In the traditional instance selection followed by classification setting, a yes/no decision needs to be made about each train instance. In this chapter, we take a different approach and associate quality weights to the instances based on fuzzy rough set theory. We propose two classifiers, POSNN

Table 9.1: Average accuracy of QFSNN over the 40 datasets for different values of r and K.

	$K=2$	$K=3$	$K=4$	$K=5$	$K=6$	$K=7$	$K=8$	$K=9$	$K=10$
$r=10$	**0.7762**	**0.7870**	**0.7899**	**0.7917**	0.7913	0.7921	0.7893	0.7918	0.7908
$r=20$	0.7757	0.7866	0.7897	0.7910	0.7903	0.7927	0.7900	0.7914	0.7912
$r=30$	0.7752	0.7848	0.7897	0.7910	**0.7914**	0.7935	0.7910	0.7917	0.7923
$r=40$	0.7756	0.7849	0.7890	0.7913	0.7907	0.7928	0.7900	0.7908	0.7927
$r=50$	0.7753	0.7859	0.7880	0.7911	0.7901	0.7933	0.7901	0.7907	0.7920
$r=60$	0.7757	0.7857	0.7883	0.7914	0.7913	0.7933	0.7911	0.7917	0.7922
$r=70$	0.7752	0.7859	0.7885	0.7905	0.7911	0.7935	0.7908	0.7914	**0.7927**
$r=80$	0.7754	0.7858	0.7865	0.7907	0.7911	0.7943	**0.7916**	0.7909	0.7925
$r=90$	0.7756	0.7862	0.7883	0.7906	0.7912	0.7936	0.7915	0.7923	0.7923
$r=100$	0.7757	0.7860	0.7881	0.7907	0.7905	**0.7944**	0.7912	**0.7928**	0.7924

Table 9.2: Average Cohen's kappa of QFSNN over the 40 datasets for different values of r and K.

	$K=2$	$K=3$	$K=4$	$K=5$	$K=6$	$K=7$	$K=8$	$K=9$	$K=10$
$r=10$	**0.5745**	0.5936	**0.5981**	0.5973	0.5986	0.5988	0.5945	0.5992	0.5963
$r=20$	0.5736	0.5933	0.5974	0.5972	0.5965	0.5994	0.5954	0.5968	0.5978
$r=30$	0.5724	0.5933	0.5978	0.5968	**0.5990**	0.6018	0.5975	0.5966	0.5980
$r=40$	0.5732	0.5936	0.5972	0.5975	0.5977	0.6007	0.5952	0.5951	0.5990
$r=50$	0.5726	0.5951	0.5952	0.5989	0.5965	0.6017	0.5951	0.5949	0.5991
$r=60$	0.5731	0.5948	0.5953	**0.5994**	0.5987	0.6017	0.5969	0.5973	0.6001
$r=70$	0.5717	0.5950	0.5957	0.5979	0.5982	0.6018	0.5962	0.5964	**0.6015**
$r=80$	0.5728	0.5953	0.5974	0.5980	0.5985	0.6034	0.5979	0.5954	0.6004
$r=90$	0.5731	**0.5962**	0.5960	0.5981	0.5987	0.6022	**0.5993**	0.6003	0.5997
$r=100$	0.5733	0.5957	0.5960	0.5980	0.5970	**0.6050**	0.5989	**0.6015**	0.6000

Table 9.3: Values of the statistics of the Wilcoxon test comparing QFSNN and POSNN against FNN with respect to accuracy

	QFSNN vs. FNN			POSNN vs. FNN		
	R+	R-	p-value	R+	R-	p-value
K=2	420	400	0.887764	417	403	0.919702
K=3	400	420	1	406	414	1
K=4	402	418	1	409	411	1
K=5	399	421	1	381	399	1
K=6	396	424	1	394	426	1
K=7	394	426	1	398	422	1
K=8	407	413	1	416	404	0.930379
K=9	401	419	1	408	412	1
K=10	403	417	1	401	379	0.872501

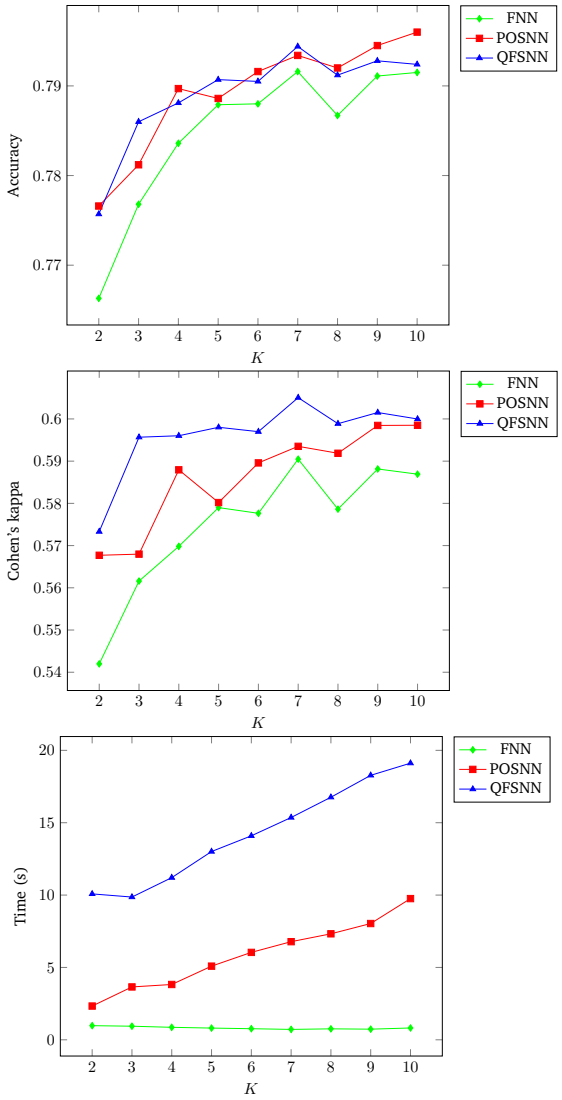


Figure 9.3.1: Average accuracy, Cohen's kappa and running time over the 40 datasets.

Table 9.4: Values of the statistics of the Wilcoxon test comparing QFSNN and POSNN against FNN with respect to Cohen's kappa

	QFSNN vs. FNN			POSNN vs. FNN		
	R+	R-	p-value	R+	R-	p-value
K=2	421	399	0.877156	361	149	0.000751
K=3	415	405	0.941068	561	259	0.918087
K=4	413	407	0.962478	601.5	218.5	0.026718
K=5	391	429	1	489	331	1
K=6	400	420	1	584.5	200.5	0.059619
K=7	410	410	0.994683	536	251	0.591934
K=8	408	412	1	699.5	120.5	0.00028
K=9	401	419	1	601.5	183.5	0.030615
K=10	400	420	1	590.5	192.5	0.015917

and QFSNN, that aim to improve FNN, an extension of KNN that takes into account the distances between the test instance and its nearest neighbors.
Our proposals are able to improve FNN, meaning that fuzzy rough set theory can be used to tackle the classification problem directly, but the differences are not significant. The latter may be due to the fact that there is a limit to what can be done using a KNN approach: if none of the nearest neighbors of a test instance has the same class as this test instance, our proposals cannot predict the class of the test instance correctly.

10. Conclusion and Future Research Directions

In this dissertation we explore the use of fuzzy rough set theory and evolutionary algorithms for instance selection. Our main goal is to improve the accuracy of classifiers by means of techniques related to instance selection.

We start off in Chapter 3 by introducing a new robust fuzzy rough set model, motivated by the fact that the minimum and maximum operations in the traditional fuzzy rough set model cause small changes in the data to be drastically reflected in the values of the fuzzy rough lower and upper approximations. Moreover, existing robust fuzzy rough set models are not able to preserve the fuzzy rough concept or violate its theoretical properties. The OWA fuzzy rough set model introduced in this thesis partially alleviates these problems. The philosophy of fuzzy rough set theory is maintained in this new model, an experimental study shows that this model is indeed more robust than the traditional fuzzy rough set model and important theoretical properties like set monotonicity are maintained. However, this model can still be improved. For instance, set inclusion, the property that states that the lower approximation is included in the set approximated and that this set is in turn included in the upper approximation, does not hold for the OWA fuzzy rough set model. An additional challenge is the selection of the weight weight vector used in the OWA operations. For now, these weights are selected by the user restricted by the orness and andness constraints, but it would be more appropriate to select these weights based on the data and application at hand.

Even though there are still some weaknesses related to the current OWA fuzzy rough set model, we show in this thesis that it is highly valuable. A good example is given in Chapter 4, where we propose a new PS method, called FRPS. This method assesses the quality of instances using the OWA fuzzy rough set model and automatically finds a good threshold to determine which instances to retain and which instances to remove from the data. Our experimental evaluation

shows that FRPS performs very well: it significantly improves all state-of-the-art methods, and moreover, FRPS is faster than the other most accurate PS techniques. This result is due to the combination of two factors: on the one hand, the OWA fuzzy rough set model seems to be an excellent tool to model the quality of instances, while on the other hand, using the train accuracy to determine a good threshold further improves the results. The fact that FRPS improves genetic approaches to PS shows that fuzzy rough set theory is indeed valuable to model the quality of instances in PS, and that the fuzzy rough component in the search strategy benefits the performance. In order to further improve FRPS, one could automatically tune the weights used in the OWA fuzzy rough set model, or use different thresholds for different classes within the training data. Another interesting future research path is to develop a condensation PS technique based on fuzzy rough set theory. FRPS focuses on removing instances in order to improve the classification, independent of the reduction. One could develop a technique that aims to remove inner points from the data in order to reduce the storage requirements and to speed up the classification applied afterward. This could be achieved by using the fuzzy rough upper approximation values, as these values express how close instances from the same class are. This technique could be applied in combination with FRPS in order to obtain a hybrid approach.

In Chapter 5 we study the combination of FRPS with FS. Motivated by the fact that FS-SSGA, a genetic approach to FS, has obtained good results in earlier studies, we decide to combine FRPS with FS-SSGA. We do not use fuzzy rough set theory for the FS part, as the existing approaches to FS using fuzzy rough set theory focus on maintaining the predictive power of training data rather than improving it. We develop a simultaneous approach to PS and FS, called SIM-FRPS-SSGA, that carries out FS-SSGA and applies FRPS at certain stages in the algorithm. An experimental evaluation shows that this approach performs better than FRPS or FS-SSGA, meaning that the PS and FS components enhance each other. Moreover, SIM-FRPS-SSGA achieves better results than the sequential application of FRPS and FS-SSGA, showing that the simultaneous approach pays off. Our new approach also outperforms SIM-SSGA-QR, a simultaneous approach that uses fuzzy rough set theory for the FS component and a genetic approach for the PS component. This result could have been expected as the components of SIM-SSGA-QR perform worse than the components of SIM-FRPS-SSGA. In order to further improve SIM-FRPS-SSGA, one could improve the components it is built on or further elaborate on how these components are merged together. For instance, one could optimize the points at which the FRPS algorithm is carried out. One should also elaborate on speeding up the algorithm as it is rather slow

now.
Using PS for KNN classification is well-studied in the literature and many techniques have been proposed. On the other hand, using TSS for SVMs is almost unexplored. The reason why researchers mostly apply PS for KNN is that PS directly affects KNN, whereas the classification model of SVM buffers between TSS and the classification of new instances. Additionally, as SVMs allow training instances to fall on the wrong sides of the decision margins, SVMs are not largely affected by noisy instances. As a result, it is harder to see the effect of TSS on SVMs. In Chapter 6, we adjust the successful genetic PS techniques and FRPS for SVMs by calculating the training accuracy using SVMs instead of KNN. Additionally, we study the effect of using genetic and fuzzy rough FS approaches on SVMs, both as stand-alone techniques and in combination with TSS. The experimental evaluations learns us that FRPS does not perform well for SVMs. The reason is probably that FRPS removes boundary instances that are crucial to build a good SVM model. The genetic TSS approaches do perform well and are able to significantly improve the performance of SVMs, but when combining them with FS the performance reduces. Using TSS for SVMs has potential, proven by the good performance of genetic TSS algorithms, that should be further explored in the future. For instance, the capabilities of fuzzy rough set theory for TSS should be investigated again, focusing on retaining the boundary instances but removing noisy instances. Additionally, one should study if other classifiers can also benefit from instance selection.
In Chapter 7, we further elaborate on genetic approaches to PS and propose an ensemble strategy that uses the information generated by genetic PS algorithms more efficiently. Genetic approaches to PS generate many good candidate prototype subsets, but only use one solution to classify new instances. Many of these generated prototype subsets might not be globally optimal, but can be optimal in certain regions of the feature space. Our framework starts from these ideas and keeps track of all prototype subsets generated during the course of the genetic PS algorithm. Then, it selects the fittest prototype subsets among them, and subsequently selects the most diverse among them. In order to classify a new instance, it is determined which among this group of diverse prototype subsets is most suited to classify instances in the neighborhood of that new instance, based on the performance of these prototype subsets for neighboring instances. We apply this approach to GGA, one of the most accurate PS algorithms. The experimental evaluation shows that our strategy, called GGA-ENS, achieves high accuracy rates, even when the GGA part only uses a limited number of evaluations in its execution. Moreover, GGA-ENS does significantly outperform GGA. This ensemble approach should be applied to other genetic PS algorithms in the

future, and could possibly be extended to genetic FS approaches or for SVM classification. Additionally, some ideas of this approach could be used for other classifiers. The good performance of random forests suggests that selecting good decision trees in the neighborhood of a certain test instance might be worthwhile.
When applying traditional classification and preprocessing techniques to imbalanced data, the minority class is generally neglected. This especially applies to FRPS, as this PS technique aims to find a prototype subset that has a good global classification accuracy, independent from the amount of minority instances correctly classified. In Chapter 8 we first adjust FRPS for imbalanced classification by changing the evaluation function. The experimental evaluation shows that this technique, called FRIPS, does indeed improve FRPS for imbalanced data. Additionally, we study the use of FRIPS and FRPS in combination with the oversampling technique SMOTE. We consider different settings where the data is cleaned using FRIPS before balancing the data using SMOTE, or where the data is cleaned using FRPS after applying SMOTE. We conclude that SMOTE followed by FRPS leads to the best results and moreover improves the state-of-the-art preprocessing techniques for imbalanced data. Possible further research on preprocessing imbalanced data could focus on a simultaneous execution of oversampling and cleaning techniques. Another option could be to elaborate on adjusting the fuzzy rough measures for imbalanced data. For instance, the OWA weights could be adjusted such that minority instances are weighted more when calculating the fuzzy rough lower and upper approximation membership values.
In the traditional setting of PS, an instance is either removed or retained in the prototype subset. In Chapter 9, we take a different approach where the instances are weighted according to their quality, based on the OWA fuzzy rough set model. We propose two classifiers, POSNN, which extends the FNN algorithm by taking the quality of instances into account, and the QFSNN classifier that extends POSNN by tuning the different weights associated to the aspects of the nearest neighbors. An experimental evaluation shows that POSNN significantly outperforms FNN, but that the improvement by QFSNN is not significant. This research shows the potential of directly tackling the classification problem using fuzzy rough set theory, but on the other hand we are also faced with its limitations, being that the removal of some instances is necessary to obtain good results. A possible future challenge could be to combine POSNN with FRPS, where FRPS only removes instances of very low quality, such that instances about which doubt exists are taken care of by POSNN.
A question that raises is which among all these proposed techniques perform best. Therefore, we give an overview in Table 10.1 of the proposed techniques in all chapters except Chapter 8 that uses different data and evaluation techniques.

Table 10.1: Average accuracy Cohen's kappa and running time over the 40 datasets in Table 3.1 for the techniques proposed in this dissertation

	Acc.	κ	Running time (s)
1NN	0.7293	0.4997	0.6769
SVM	0.7596	0.5557	1.3500
FRPS (1NN)	0.7771	0.5696	3.9383
SIM-FRPS-SSGA (1NN)	0.7983	0.6050	67.3458
GGA (SVM)	0.8068	0.5880	3384.74
GGA (1NN)	0.7591	0.5434	63.5139
GGA-ENS (1NN)	0.7682	0.5539	73.5188
POSNN (10NN)	0.7960	0.5985	9.7567
QFSNN (10NN)	0.7924	0.6000	19.1143
10NN	0.7614	0.5419	0.7020
FNN (10NN)	0.7915	0.5869	0.8174

In Table 10.1, we list the average results over the 40 datasets described in Table 3.1.
Among the techniques that aim to improve 1NN classification, SIM-FRPS-SSGA performs best, followed by FRPS and GGA-ENS respectively. As the computational cost of SIM-FRPS-SSGA is high, FRPS is more suited if time constraints are imposed. Both POSNN and QFSNN improve 10NN classification, but recall that POSNN is faster than QFSNN and should therefore be preferred. As we showed in Chapter 5, GGA improves SVM classification. SVM classification enhanced with GGA TSS clearly performs best among all methods with respect to accuracy. However, the computational time required by this method is very high, therefore other techniques may be preferred. When we compare the methods with respect to Cohen's kappa, QFSNN, POSNN and SIM-FRPS-SSGA perform well, POSNN is faster than SIM-FRPS-SSGA and QFSNN but is slightly less accurate. We conclude that there is no clear answer to the question which technique should be preferred for an application, and the user should base the selection based on which performance measure is most important for him.
Besides further enhancing the techniques proposed in this thesis, there are many new research directions that could be explored.
One important research topic is instance selection for regression. On the one hand, this requires improved fuzzy rough set based quality measures, and on the other hand one should carefully select and study regression techniques that can benefit from instance selection. One option could be to consider SVMs for

regression, and to study if genetic approaches to instance selection for these SVMs perform as well as for classification problems. An additional challenge in this topic is to find a good measure to assess the fitness of candidate instance subsets, as the training accuracy used for classification problems does not apply.
A promising research path is Prototype Generation (PG, [156]), a technique that removes instances from the data like PS but also generates new instances. As such, regions in the feature space without representative examples can be filled. The use of fuzzy rough set theory in this field is yet unexplored, but seems to lend itself very well to the problem. Indeed, fuzzy rough set theory is an excellent tool to assess the quality of instances, and candidate artificial instances can be easily evaluated using the fuzzy rough set model.
Another important challenge is instance selection techniques for big data. The datasets used in this thesis contain up to 2000 instances, which is a low number in this era of big data. There are several options to adjust instance selection techniques to big data. One could plug in instance selection techniques into existing frameworks for big data [33, 34, 60, 61] that approximate the behavior of the instance selection technique. A simple example is to divide the data into strata, apply the instance selection technique to each stratus and merge the resulting instance subsets. Another option is to try to perfectly mimic the instance selection techniques using big data software platforms.
To conclude, we mention that for certain research topics considered in this thesis, improvements in accuracy were hard to obtain. We believe that in some cases we attained some limit to what can be done for generic datasets. Keeping this in mind, future research should maybe focus on tailor-made solutions to specific problems.

Publication List

Publications in International Journals

- VERBIEST, N., CORNELIS, C., SAEYS, Y. AND GARCÍA, N. Improving Nearest Neighbor Classification using Ensembles of Evolutionary Generated Prototype Subsets, Submitted
- D'EER, L. VERBIEST, N., CORNELIS, C. AND GODO, L. A Comprehensive Study of Implicator-Conjunctor Based and Noise-Tolerant Fuzzy Rough Sets: Definitions, Properties and Robustness Analysis, Submitted
- VERBIEST, N., DERRAC, J., CORNELIS, C. AND HERRERA, F. Evolutionary Wrapper Approaches for Training Set Selection as Preprocessing Mechanism for Support Vector Machines: an Experimental Evaluation, Submitted
- RAMENTOL, E., VLUYMANS, S., VERBIEST, N., Caballero, Y., BELLO, R., CORNELIS, C. AND HERRERA, F. IFROWANN: Imbalanced Fuzzy-Rough Ordered Weighted Average Nearest Neighbor Classification, Submitted
- VERBIEST, N., RAMENTOL, E., CORNELIS, C. AND HERRERA, F. Preprocessing Noisy Imbalanced Datasets using SMOTE enhanced with Fuzzy Rough Prototype Selection, Accepted to Applied Soft Computing
- CORNELIS, C., MEDINA, J. AND VERBIEST, N. Multi-Adjoint Fuzzy Rough Sets: Definition, Properties and Attribute Selection. International Journal of Approximate Reasoning 55 (2014),412-426.
- DERRAC, J., VERBIEST, N., GARCÍA, S., CORNELIS, C. AND HERRERA, F. On the Use of Evolutionary Feature Selection for Improving Fuzzy Rough Set Based Prototype Selection. Soft Computing 17(2) (2013), 223-238.
- VERBIEST, N., CORNELIS, C. AND HERRERA, F., FRPS: a Fuzzy Rough Prototype Selection Method, Pattern Recognition. 46(10) (2013), 2770-2782.

- VICTOR, P., VERBIEST, N., CORNELIS, C. AND DE COCK, M., Enhancing the Trust-Based Recommendation Process with Explicit Distrust. ACM Transactions on the Web 7(2) (2013), 6:1-6:19.
- VERBIEST, N., CORNELIS, C., VICTOR, P. AND HERRERA-VIEDMA, E., Trust and Distrust Aggregation Enhanced with Path Length Incorporation. Fuzzy Sets and Systems 202 (2012), 61-74.

Chapters in Books

- VERBIEST, N., VERMEULEN, K., TEREDESAI, A., Evaluation of Classification Methods. In: Data Classification: Algorithms and Applications, CRC Press, 2014.

Conference Proceedings

- D'EER, L. VERBIEST, N., CORNELIS, C. AND GODO, L., Modelos de conjuntos difusos rugosos tolerantes al ruido: definiciones y propiedades. In: Proceedings of XVII Congreso Español sobre Tecnologías y Lógica Fuzzy (2014), 27-32.
- VERBIEST, N., CORNELIS, C. AND HERRERA, F., OWA-FRPS: A Prototype Selection method based on Ordered Weighted Average Fuzzy Rough Set Theory, In: Proceedings of the 14th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing (2013), 180-190.
- D'EER, L. VERBIEST, N., CORNELIS, C. AND GODO, L., Implicator-Conjunctor Based Models of Fuzzy Rough Sets: Definitions and Properties. In: Proceedings of the 14th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing (2013), 169-179.
- MEADEM, N., VERBIEST, N., ZOLFAGHAR, K., AGARWAL, J., CHIN, S., BASU ROY, S. AND TEREDESAI, A. Exploring Preprocessing Techniques for Prediction of Risk of Readmission for Congestive Heart Failure Patients. In: Proceedings of the Data Mining for Healthcare (DMH) Workshop at KDD 2013
- ZOLFAGHAR, K., AGARWAL, J., CHIN, S., BASU ROY, S. AND TEREDESAI, A., Risk-O-Meter: An Intelligent Healthcare Risk Calculator. In: Proceedings of the KDD Demo-track at KDD 2013
- VERBIEST, N., CORNELIS, C. AND JENSEN, R. Quality, Frequency and Similarity Based Fuzzy Nearest Neighbor Classification. In: Proceedings of the IEEE International Conference on Fuzzy Systems (2013)
- RAMENTOL, E., VERBIEST, N., BELLO, R., Caballero, Y., CORNELIS, C. AND HERRERA, F. SMOTE-FRST: A new resampling method using fuzzy

rough set theory. In: Proceedings of the 10th International FLINS Conference on uncertainty Modeling in Knowledge Engineering and Decision Making (2012), 800-805.

- VERBIEST, N., RAMENTOL, E., CORNELIS, C. AND HERRERA, F. 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 (2012), 169-178.
- MOURISSE, D., LEFEVER, E., VERBIEST, N., SAEYS, Y., DE COCK, M., CORNELIS, C. SBFC: An Efficient Feature Frequency-Based Approach to Tackle Cross-Lingual Word Sense Disambiguation. In: Proceedings of the 15th International Conference on Text, Speech and Dialogue (2012), 248-255.
- VERBIEST, N., CORNELIS, C. AND JENSEN, R. Fuzzy Rough Positive Region-based Nearest Neighbour Classification. In: Proceedings of the 20th International Conference on Fuzzy Systems (2012), 1961-1967.
- VERBIEST, N., CORNELIS, C. AND HERRERA, F., Selección de Prototipos Basada en Conjuntos Rugosos Difusos. In: Proceedings of XVI Congreso Español sobre Tecnologías y Lógica Fuzzy (2012), 638-643.
- CORNELIS, C., MEDINA, J. AND VERBIEST, N., Multi-Adjoint Fuzzy Rough Sets, In: Proceedings of the 3rd Workshop on Rough Set Theory (2011)
- CORNELIS, C., VERBIEST, N. AND JENSEN, R., Ordered Weighted Average Based Fuzzy Rough Sets, In: Proceedings of the 5th International Conference on Rough Sets and Knowledge Technology (2010),78-85.
- VERBIEST, N., CORNELIS, C., VICTOR, P. AND HERRERA-VIEDMA, E., Strategies for Incorporating Knowledge Defects and Path Length in Trust Aggregation, In: Proceedings of The 23rd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (2010), 450-459.
- VERBIEST, N., CORNELIS, C. AND SAEYS, Y., Valued Constraint Satisfaction Problems Applied to Functional Harmony, In: Proceedings of the 13th International Fuzzy Systems Association World Congress and 6th European Society for Fuzzy Logic and Technology Conference (2009), 925-930.

Samenvatting

Eén van de belangrijkste onderzoeksgebieden in data ontginning is classificatie. Classificatietechnieken proberen de klassen van nieuwe datapunten waarvoor enkel de beschrijvende attributen gegeven zijn te voorspellen aan de hand van trainingsdata bestaande uit datapunten waarvoor zowel de beschrijvende attributen als de klassen gekend zijn.
In veel situaties wordt de data niet direct gebruikt voor classificatie maar is het aangewezen om deze eerst te bewerken. Een veel bestudeerde techniek is attribuutselectie, waarbij overbodige en misleidende attributen worden verwijderd. In deze thesis focussen we op datapuntselectie, de duale techniek van attribuutselectie (Feature Selection, FS). Datapuntselectie bewerkt de data door een deelverzameling van de datapunten als trainingsdata te selecteren. Enerzijds kan de nauwkeurigheid van de classificatietechniek die achteraf wordt toegepast worden verbeterd door datapunten met ruis te verwijderen, terwijl anderzijds de computationele vereisten van het leer- en classificatieproces kunnen worden verkleind.
In deze thesis focussen we op datapuntselectie voor twee classificatietechnieken: K Nearest Neighbors (KNN) en Support Vector Machines (SVMs). Als datapuntselectie wordt gebruikt in de context van KNN spreken we over Prototype Selectie (PS), in de context van SVM gebruiken we de term Training Set Selectie (TSS).
De technieken voor datapuntselectie die we in deze thesis bespreken zijn gebaseerd op twee modellen, evolutionaire algoritmen en de vaagruwverzamelingenleer. Het gebruik van evolutionaire algoritmen voor datapuntselectie is grondig bestudeerd in de literatuur. Veel methoden zijn ontwikkeld en experimentele studies hebben aangetoond dat deze technieken de meest nauwkeurige datapuntselectietechnieken zijn [55]. Anderzijds is slechts weinig onderzoek gedaan over datapuntselectie gebaseerd op vaagruwverzamelingentheorie. Er is slechts

één methode voorgesteld [80] maar deze is niet zo nauwkeurig als andere datapuntselectietechnieken.
Gemotiveerd door de hoge nauwkeurigheid van evolutionaire algoritmen voor datapuntselectie en door het feit dat vaagruwverzamelingen hun nut bewezen hebben in veel andere aspecten van data onginning, focussen we in deze thesis op deze modellen en hun onderlinge combinatie.
Vaagruwverzamelingen [29, 45, 46] combineren ruwverzamelingen [124] en vaagverzamelingen [181]. Ze benaderen concepten door middel van de onder- en bovenbenadering. De traditionele definitie van vaagruwverzamelingen is gebaseerd op de strikte minimum en maximum bewerkingen, wat robuustheidsproblemen veroorzaakt: kleine veranderingen in de data kunnen drastische gevolgen hebben voor de waarden van de boven- en onderbenadering. Dit gebrek aan robuustheid is vooral een probleem als men met reële datasets werkt, aangezien deze ruis of verkeerd gelabelde datapunten kunnen bevatten. In Hoofdstuk 2 behandelen we dit probleem door een nieuw robuust model voor vaagruwverzamelingen in te voeren, Ordered Weighted Average (OWA, [176]) vaagruwverzamelingen. Door de strikte minimum en maximum bewerkingen te vervangen door hun minder strikte OWA uitbreidingen bekomen we vaagruwverzamelingen die robuust zijn tegen ruis zowel in de klassen als in de voorwaardelijke attributen.
Alle technieken in deze thesis die gebaseerd zijn op de vaagruwverzamelingenleer maken gebruik van dit nieuwe robuuste OWA vaagruwverzamelingenmodel. In Hoofdstuk 3 voeren we Fuzzy Rough Prototype Selection (FRPS) in, een nieuwe PS techniek gebaseerd op vaagruwverzamelingen. Het belangrijkste idee achter FRPS is dat de datapunten die een hoge lidmaatschapsgraad hebben tot de boven- en onderbenadering van hun eigen klasse het meest bruikbaar zijn voor classificatie. Deze datapunten liggen dicht bij datapunten van hun eigen klasse en ver van datapunten van andere klassen. Een experimentele evaluatie toont aan dat het gebruik van het OWA vaagruwverzamelingenmodel voor de onderbenadering het meest geschikt is om de kwaliteit van datapunten te meten. FRPS ordent de datapunten volgens deze kwaliteitsmaat en bepaalt vervolgens automatisch een drempelwaarde om te beslissen welke datapunten moeten verwijderd worden, gebaseerd op de 1NN trainingsnauwkeurigheid van kandidaat drempelwaarden en hun corresponderende prototype deelverzamelingen. Een experimentele studie toont aan dat FRPS significant nauwkeuriger is dan andere PS technieken.
In Hoofdstuk 4 gaan we na of FRPS nog verbeterd kan worden door gebruik te maken van FS. Het na elkaar uitvoeren van FRPS en FS kan problemen veroorzaken aangezien FRPS datapunten kan verwijderen die belangrijk zijn voor het

FS proces, terwijl FS belangrijke attributen kan verwijderen die noodzakelijk zijn voor het FRPS algoritme. We stellen een techniek voor die simultaan FRPS en FS uitvoert, SIM-FRPS-SSGA. Deze techniek voert een Steady State Genetic FS Algoritme uit (FS-SSGA) en past FRPS toe op bepaalde punten in het FS proces. We tonen aan dat SIM-FRPS-SSGA significant beter werkt dan elk van zijn componenten FRPS en FS-SSGA, wat betekent dat FRPS en FS-SSGA elkaar versterken. We concluderen ook dat SIM-FRPS-SSGA beter werkt dan het na elkaar uitvoeren van FS en PS.

In Hoofdstuk 3 en 4 ontwikkelen we technieken die 1NN classificatie verbeteren, in Hoofdstuk 5 breiden we dit uit naar SVM classificatie. We passen de evolutionaire PS technieken en FRPS aan voor SVMs door de evaluatiefunctie in de algoritmen te vervangen. Het FRPS algoritme bepaalt de kwaliteit van een kandidaat deelverzameling van prototypes door de corresponderende trainingsnauwkeurigheid te berekenen, gebruik makend van de 1NN classificatietechniek. We passen FRPS aan voor SVMs door de 1NN classificatie te vervangen door SVM. Dezelfde strategie passen we toe voor de evolutionaire algoritmen: de trainingsnauwkeurigheid in de fitness functie wordt berekend door SVM toe te passen. Dit principe kan ook worden gebruikt voor de technieken uit Hoofdstuk 4 die FS en PS combineren. Een experimentele evaluatie toont aan dat evolutionaire TSS technieken goed scoren en de classificatie van SVMs significant kunnen verbeteren. Het combineren van TSS met FS kan de prestatie van SVMs niet verbeteren doordat FS de nauwkeurigheid van SVMs slecht beïnvloedt.

Het grootste nadeel van evolutionaire PS technieken is dat deze veel tijd vergen. Enkel door het evalueren en verbeteren van vele generaties van kandidaat prototype deelverzamelingen kunnen goede prestaties worden bereikt. In Hoofdstuk 6 stellen we een nieuwe classificatietechniek voor, Generational Genetic Algorithm-Ensemble (GGA-ENS), die gebaseerd is op een evolutionaire PS methode maar die minder evaluaties nodig heeft om nauwkeurige resultaten te bereiken. We vertrekken van het idee dat tijdens het verloop van een evolutionair PS algoritme veel goede kandidaat prototype deelverzamelingen worden gegenereerd. GGA-ENS gebruikt meer dan één van deze deelverzamelingen voor de uiteindelijke classificatie van nieuwe datapunten. Het Generational Genetic PS Algoritme (GGA) wordt uitgevoerd, de beste prototype deelverzamelingen die tijdens het verloop van het algoritme worden gegenereerd worden opgeslagen en op het einde worden de meest diverse deelverzamelingen geselecteerd. Om een nieuw datapunt te classificeren worden zijn dichtste trainingsdatapunten (dichtste buren) geclassificeerd gebruik makend van alle prototype deelverzamelingen. Elke deelverzameling krijgt een gewicht afhankelijk van zijn classificatie: een deelverzameling die veel van de dichtste buren van het nieuwe datapunt cor-

rect classificeert krijgt een hoog gewicht. Vervolgens worden de prototype deelverzamelingen gebruikt in een gewogen stemproces om het nieuwe datapunt te classificeren. Een experimentele evaluatie toont aan dat GGA-ENS significant beter presteert dan GGA en bovendien nauwkeurigere resultaten behaalt voor minder evaluaties van het genetisch algoritme.
In Hoofdstuk 7 evalueren we FRPS op ongebalanceerde datasets, waarbij één of meerdere klassen beduidend minder datapunten bevatten dan de andere klassen. We tonen aan dat FRPS verbeterd kan worden voor ongebalanceerde datasets door de trainingsnauwkeurigheid te vervangen door de Adjusted Area Under the Curve (AAUC) maat, een evaluatiemaat specifiek bedoeld voor ongebalanceerde data. We noemen deze methode Fuzzy Rough Imbalanced Prototype Selection (FRIPS). We gaan na hoe FRIPS en FRPS presteren in combinatie met de Synthetic Minority Over Sampling Technique (SMOTE, [23]), een methode die artificiële datapunten toevoegt aan de data om deze te balanceren. We beschouwen verschillende opstellingen waarbij FRIPS kan worden gebruikt om de data te verbeteren vooraleer SMOTE wordt toegepast, of waarbij FRPS kan worden gebruikt om de data te verbeteren nadat SMOTE nieuwe datapunten heeft toegevoegd aan de data. Een experimentele evaluatie toont aan dat de laatste opstelling de beste resultaten behaalt en dat deze de huidige technieken voor ongebalanceerde data significant verbetert.
FRPS maakt gebruik van vaagruwverzamelingen om de kwaliteit van datapunten te meten en verwijdert datapunten die een lage waarde hebben voor deze maat. In Hoofdstuk 8 benaderen we het probleem op een andere manier: in plaats van datapunten te verwijderen die een lage waarde hebben voor de kwaliteitsmaat brengen we ze minder in rekening tijdens de classificatie. We gebruiken de Fuzzy Nearest Neighbor (FNN, [87]) methode als uitgangspunt. FNN verbetert KNN classificatie door de afstand tussen de nieuwe datapunten en hun dichtste buren in rekening te brengen. We stellen een nieuwe classificatietechniek voor, Fuzzy Rough Positive Nearest Neighbor (POSNN) die FNN uitbreidt door ook de kwaliteit van de datapunten in rekening te brengen. We breiden deze methode verder uit naar de Quality Frequency and Similarity (QFSNN) classificatietechniek die de mate waarin de verschillende componenten in rekening worden gebracht tijdens de classificatie automatisch bepaalt. Een experimentele evaluatie toont aan dat POSNN FNN significant verbetert, de verbetering door QFSNN is niet significant.

Summary

One of the most important fields of data mining is classification. Given train data consisting of instances for which conditional feature values and class labels are known, classification techniques aim to predict the class labels of new test instances for which only the conditional feature values are known.
In many real-world situations, the data obtained is not used directly for classification of new instances but is first subjected to a preprocessing step. One preprocessing technique that has been studied extensively is Feature Selection (FS), where redundant and misleading features are removed. In this dissertation we focus on instance selection, the complimentary technique of FS. Instance selection preprocesses the data by selecting a subset of instances as train data. The goal of instance selection is twofold: on the one hand the performance of the classifier applied afterwards can be improved by removing noisy instances, while on the other hand the computational time required for the training and classification process can be reduced.
In this dissertation we focus on instance selection for two classifiers: K Nearest Neighbors (KNN) and Support Vector Machines (SVM). When instance selection is used in the context of KNN, we refer to it as Prototype Selection (PS), in the context of SVMs we refer to instance selection as Training Set Selection (TSS).
The techniques for instance selection that we study in this dissertation rely on two models, evolutionary algorithms and fuzzy rough set theory. Using evolutionary algorithms to approach instance selection has been studied extensively in the literature. Many methods have been proposed and experimental studies have shown that these techniques are the most accurate among the state-of-the-art [55]. On the other hand, instance selection based on fuzzy rough set theory is yet unexplored. Only one method has been proposed [80]but it does not perform as well as the best state-of-the-art techniques.

Motivated by the good performance of evolutionary algorithms for instance selection and by the fact that fuzzy rough set theory has proven to be useful in other data mining fields, we focus on these models and their hybridization in this dissertation.

Fuzzy rough set theory [29, 45, 46] is the hybridization of rough set theory [124] and fuzzy set theory [181] and approximates concepts by means of the lower and upper approximation. The traditional definition of fuzzy rough sets is based on the strict minimum and maximum operators, which causes robustness problems: small changes in the data can result in large changes in the lower and upper approximation membership values. This lack of robustness is particularly inconvenient when working with real-world data sets, as they may contain noisy or mislabeled instances. In Chapter 2 we deal with this problem by introducing a new robust fuzzy rough set model, called Ordered Weighted Average (OWA, [176]) fuzzy rough sets. By replacing the minimum and maximum operators by their more general OWA counterparts, we obtain a fuzzy rough set model that is robust against both class and attribute noise.

All fuzzy rough set theory based techniques proposed in this dissertation are based on the OWA fuzzy rough set model. In Chapter 3 we introduce a new fuzzy rough PS technique, called Fuzzy Rough Prototype Selection (FRPS). The main idea of FRPS is that instances that have a high membership degree to the lower and upper approximation of their own class are most useful for classification. These instances are close to instances from their own class and distant from instances from different classes. An experimental evaluation shows that using the OWA fuzzy rough lower approximation is most suited to model the quality of instances. The FRPS algorithm orders the instances according to this quality measure and automatically determines a good threshold to decide which instances to remove from the data, based on the 1NN train accuracy of candidate thresholds and their corresponding prototype subsets. An experimental evaluation shows that the FRPS algorithm significantly outperforms the state-of-the-art in PS, and moreover, FRPS is faster than the best state-of-the-art PS techniques.

In Chapter 4 we study if FRPS can be further enhanced using FS. Applying FRPS and FS subsequently can cause problems as FRPS might remove instances that are relevant for the FS process, or as FS might remove features that are important for the FRPS algorithm. Therefore, we propose a technique that carries out FRPS and FS simultaneously, called SIM-FRPS-SSGA. This technique carries out a Steady State Genetic FS Algorithm (FS-SSGA), and applies FRPS at certain stages of this FS process. We experimentally demonstrate that SIM-FRPS-SSGA significantly outperforms its components FRPS and FS-SSGA, that is, the FRPS and FS-SSGA algorithm reinforce each other. We also conclude that SIM-FRPS-SSGA

performs better than applying subsequent FS and PS.
In Chapter 3 and Chapter 4 we focus on improving 1NN classification, in Chapter 5 we extend this to SVM classification. We adjust the evolutionary PS techniques and FRPS for SVMs by replacing the evaluation function in the respective algorithms. The FRPS algorithm assesses the quality of candidate prototype subsets by calculating the train accuracy using the 1NN classifier, the FRPS algorithm for SVM replaces this 1NN classifier by SVM. The same strategy is applied to the evolutionary algorithms: the train accuracy component in the fitness function is now calculated using the SVM classifier. This process can also be applied to the techniques that combine FS with PS, proposed in Chapter 4. An experimental evaluation shows that evolutionary TSS performs well and can improve SVM classification significantly. Combining TSS with FS does not improve the performance of SVMs, due to the fact that FS deteriorates the SVM classification.
The main drawback of evolutionary PS techniques are their high computational requirements. Only by evaluating and improving many generations of candidate prototype subsets, good accuracy rates are achieved. In Chapter 6 we focus on this problem and propose a classifier called Generational Genetic Algorithm Ensemble (GGA-ENS) based on evolutionary PS that requires less evaluations to achieve higher accuracy rates. We start from the idea that during the course of evolutionary PS algorithms, many good candidate prototype subsets are encountered. The GGA-ENS classifier exploits this idea by using more than one prototype subset for the final classification. The Generational Genetic Algorithm (GGA) for PS is carried out, the best prototype subsets are kept track of and the most diverse among them are selected. In order to classify a new instance, its neighbors are classified using these prototype subsets. Each prototype subset gets a weight according to this classification. Prototype subsets that can classify the nearest neighbors of the test instances well get a high weight. Then, the prototype subsets are used in a weighted voting process to classify the test instance. An experimental evaluation shows that GGA-ENS outperforms GGA, and moreover, it achieves high accuracy rates for less evaluations of the genetic algorithm.
In Chapter 7 we evaluate FRPS on imbalanced datasets, these are datasets where at least one class is substantially outnumbered by the other classes. We show that FRPS can be enhanced for imbalanced datasets by replacing the train accuracy measure by the Adjusted Area Under the Curve (AAUC) measure, an evaluation measure developed for imbalanced data. We call this method Fuzzy Rough Imbalanced Prototype Selection (FRIPS). We study the behavior of FRIPS and FRPS in combination with the Synthetic Minority Over Sampling TEchnique (SMOTE, [23]), a preprocessing technique for imbalanced datasets that introduces artificial

minority instances in order to balance the dataset. We consider different settings where FRIPS is used to clean the data before applying SMOTE and where FRPS is used to clean the data after applying SMOTE. An experimental evaluation shows that applying SMOTE and FRPS subsequently outperforms the state-of-the-art in SMOTE-based preprocessing techniques for imbalanced data.

The FRPS technique uses fuzzy rough set theory to assess the quality of instances, and removes instances of low quality. In Chapter 8 we take a different approach: instead of removing low-quality instances, we give them a lower weight. We use the Fuzzy Nearest Neighbor (FNN, [87]) method as starting point. This method improves KNN classification by taking the distance between the target test instance and its nearest neighbors into account. We propose a classifier called Fuzzy Rough Positive Region based Nearest Neighbor (POSNN) that extends FNN by additionally taking the quality of instances into account, using the FRPS quality measure. We further enhance this method to the Quality, Frequency and Similarity (QFSNN) classifier, that tunes the extent to which the separate components are taken into account. An experimental evaluation shows that POSNN significantly outperforms FNN, while for QFSNN the improvement is not significant.

Bibliography

[1] AGGARWAL, C. *Data Classification: Algorithms and Applications*. Chapman and Hall, 2014.

[2] AHA, D., KIBLER, D., AND ALBERT, M. Instance-based learning algorithms. *Machine Learning 6*, 1 (1991), 37–66.

[3] ALCALÁ, J., FERNANDEZ, A., LUENGO, J., DERRAC, J., GARCÍA, S., SÁNCHEZ, L., AND HERRERA, F. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *Journal of Multiple-Valued Logic and Soft Computing 17*, 2 (2011), 255–287.

[4] ALLWEIN, E. L., SCHAPIRE, R., AND SINGER, Y. Reducing multiclass to binary: A unifying approach for margin classifiers. *Journal of Machine Learning Research 1* (2000), 113–141.

[5] ANGIULLI, F. Fast nearest neighbor condensation for large data sets classification. *IEEE Transactions on Knowledge and Data Engineering 19*, 11 (2007), 1450–1464.

[6] BACHE, K., AND LICHMAN, M. UCI machine learning repository, 2013.

[7] BAKIRLI, G., BIRANT, D., AND KUT, A. An incremental genetic algorithm for classification and sensitivity analysis of its parameters. *Expert Systems With Applications 38* (2011), 2609–2620.

[8] BARANDELA, R., FERRI, F., AND SÁNCHEZ, J. Decision boundary preserving prototype selection for nearest neighbor classification. *International Journal of Pattern Recognition and Artificial Intelligence 19*, 6 (2005), 787–806.

[9] BARUA, S., ISLAM, M., YAO, X., AND MURASE, K. Mwmote-majority weighted minority oversampling technique for imbalanced data set learn-

ing. *IEEE Transactions on Knowledge and Data Engineering 26*, 2 (2014), 405–425.

[10] BATISTA, G., AND MONARD, M. An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence 17*, 5 (2003), 519–533.

[11] BATISTA, G., PRATI, R., AND MONARD, M. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explorations 6*, 1 (2004), 20–29.

[12] BEN-DAVID, A. Comparison of classification accuracy using cohen's weighted kappa. *Expert Systems with Applications 34*, 2 (2008), 825–832.

[13] BHATT, R., AND GOPAL, M. Frct: fuzzy-rough classification trees. *Pattern Analysis and Applications 11* (2008), 73–88.

[14] BHATT, R. B., AND GOPAL, M. On fuzzy-rough sets approach to feature selection. *Pattern Recognition Letters 26*, 7 (2005), 965 – 975.

[15] BIAN, H., AND L. MAZLACK, L. Fuzzy-rough nearest-neighbor classification approach. In *22nd International Conference of the North American Fuzzy Information Processing Society* (2003), pp. 500 – 505.

[16] BREIMAN, L., FRIEDMAN, J., OLSHEN, R., AND STONE, C. *Classification and Regression Trees*. Chapman and Hall, 1984.

[17] BRIGHTON, H., AND MELLISH, C. Advances in instance selection for instance-based learning algorithms. *Data Mining and Knowledge Discovery 6* (2002), 153–172.

[18] BRODLEY, C. Adressing the selective superiority problem: Automatic algorithm/model class selection. In *Proceedings of the 10th International Machine Learning Conference* (1993), pp. 17–24.

[19] BUNKHUMPORNPAT, C., SINAPIROMSARAN, K., AND LURSINSAP, C. Safe-level-smote: Safe-level-synthetic minority over-sampling technique for handling the class imbalanced problem. In *Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining* (2009), vol. 5476, pp. 475–482.

[20] CAMERON-JONES, R. Instance selection by encoding length heuristic with random mutation hill climbing. In *Proceedings of the 8th Australian Joint Conference on Artificial Intelligence* (1995), pp. 99–106.

[21] CANO, J. R., HERRERA, F., AND LOZANO, M. Using evolutionary algorithms as instance selection for data reduction in kdd: an experimental study. *IEEE Transactions on Evolutionary Computation 7*, 6 (2003), 561–575.

[22] CASILLAS, J., CORDÓN, O., DEL JESUS, M., AND HERRERA, F. Genetic feature selection in a fuzzy rule-based classification system learning process. *Information Sciences 136* (2001), 135–157.

[23] CHAWLA, N., BOWYER, K., HALL, L., AND KEGELMEYER, W. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research 16* (2002), 321–357.

[24] CHAWLA, N., LAZAREVIC, A., HALL, L., AND BOWYER, K. Smoteboost: Improving prediction of the minority class in boosting. In *Proceedings of the Conference on Knowledge Discovery in Databases* (2003), pp. 107–109.

[25] CHEN, D., HE, Q., AND WANG, X. Frsvms: Fuzzy rough set based support vector machines. *Fuzzy Sets and Systems 161*, 4 (2010), 596 – 607.

[26] CHEN, D., KWONG, S., HE, Q., AND WANG, H. Geometrical interpretation and applications of membership functions with fuzzy rough sets. *Fuzzy Sets and Systems 193* (2012), 122–135.

[27] CHEN, J., ZHANG, C., XUE, X., AND LIU, C. L. Fast instance selection for speeding up support vector machines. *Knowledge-Based Systems 45* (2013), 1–7.

[28] CORNELIS, C., DE COCK, M., AND RADZIKOWSKA, A. Vaguely quantified rough sets. In *Proceedings of the 11th International Conference on Rough Sets, Fuzzy Sets, Data Mining and Granular Computing* (2007), pp. 87–94.

[29] CORNELIS, C., DE COCK, M., AND RADZIKOWSKA, A. M. Fuzzy rough sets: from theory into practice. In *Handbook of Granular Computing* (2008), W. Pedrycz, A. Skowron, and V. Kreinovich, Eds., pp. 533–552.

[30] CORNELIS, C., JENSEN, R., HURTADO MARTIN, G., AND SLEZAK, D. Attribute selection with fuzzy decision reducts. *Information Sciences 180*, 2 (2010), 209–224.

[31] CORNELIS, C., VERBIEST, N., AND JENSEN, R. Ordered weighted average based fuzzy rough sets. In *Proceedings of the 5th International Conference on Rough Sets and Knowledge Technology* (2010), pp. 78–85.

[32] COVER, T., AND HART, P. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory 13*, 1 (1967), 21–27.

[33] DE HARO-GARCÍA, A., AND GARCÍA-PEDRAJAS, N. A divide-and-conquer recursive approach for scaling up instance selection algorithms. *Data Mining and Knowledge Discovery 18*, 3 (2009), 392–418.

[34] DE HARO-GARCÍA, A., GARCÍA-PEDRAJAS, N., AND DEL CASTILLO, J. A. R. Large scale instance selection by means of federal instance selection. *Data and Knowledge Engineering 75* (2012), 58–77.

[35] DEHURI, S., PATNAIK, S., GHOSH, A., AND MALL, R. Application of elitist multi-objective genetic algorithm for classification rule generation. *Applied Soft Computing 8* (2008), 477–487.

[36] DERRAC, J., CORNELIS, C., GARCÍA, S., AND HERRERA, F. Enhancing evolutionary instance selection algorithms by means of fuzzy rough set based feature selection. *Information Sciences 186*, 1 (2012), 73–92.

[37] DERRAC, J., GARCÍA, S., MOLINA, D., AND HERRERA, F. 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 (2011), 3–18.

[38] DERRAC, J., VERBIEST, N., GARCÍA, S., CORNELIS, C., AND HERRERA, F. On the use of evolutionary feature selection for improving fuzzy rough set based prototype selection. *Soft Computing* (2012), 1–16.

[39] DEVI, V. S., AND MURTY, M. N. An incremental prototype set building technique. *Pattern Recognition 35*, 2 (2002), 505–513.

[40] DEVROYE, L., AND WAGNER, T. Distribution-free performance bounds for potention function rules. *IEEE Transaction in Information Theory 25*, 5 (1979), 601–604.

[41] DIAMANTIDIS, N. A., KARLIS, D., AND GIAKOUMAKIS, E. A. Unsupervised stratification of cross-validation for accuracy estimation. *Artificial Intelligence 116*, 1-2 (2000), 1–16.

[42] DIAO, R., AND SHEN, Q. A harmony search based approach to hybrid fuzzy-rough rule induction. In *Proceedings of the 21st International Conference on Fuzzy Systems* (2012), pp. 1549–1556.

[43] DOMINGOS, P., AND PAZZANI, M. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning 29* (1997), 103–137.

[44] DORIGO, M., AND BLUM, C. Ant colony optimization theory: A survey. *Theoretical Computer Science 344*, 2–3 (2005), 243–278.

[45] DUBOIS, D., AND PRADE, H. Rough fuzzy sets and fuzzy rough sets. *International Journal of General Systems 17* (1990), 191–209.

[46] DUBOIS, D., AND PRADE, H. Putting fuzzy sets and rough sets together. In *Intelligent decision support- Handbook of Applications and Advances of the Rough Sets Theory* (1992), R. Slowinski, Ed., pp. 203–232.

[47] ELASHIRI, M., HEFNY, H., AND ELWAHAB, A. A. Induction of fuzzy decision trees based on fuzzy rough set techniques. In *Proceedings of the International Conference on Computer Engineering Systems* (2011), pp. 134–139.

[48] FAWCETT, T., AND PROVOST, F. Adaptive fraud detection. *Data Mining and Knowledge Discovery 1*, 3 (1997), 291–316.

[49] FRIEDMAN, M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association 32* (1937), 674–701.

[50] FRIEDMAN, M. A comparison of alternative tests of significance for the problem of m rankings. *Annals of Mathematical Statistics 11* (1940), 86–92.

[51] FRITZKE, B. A growing neural gas network learns topologies. In *Advances in Neural Information Processing Systems 7*. MIT Press, 1995, pp. 625–632.

[52] GALAR, M., FERNANDEZ, A., BARRENECHEA, E., BUSTINCE, H., AND HERRERA, F. 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 (2011), 1761–1776.

[53] GALAR, M., FERNÁNDEZ, A., BARRENECHEA, E., AND HERRERA, F. Eusboost: Enhancing ensembles for highly imbalanced data-sets by evolutionary undersampling. *Pattern Recognition 46*, 12 (2013), 3460–3471.

[54] GARCÍA, S., CANO, J. R., AND HERRERA, F. A memetic algorithm for evolutionary prototype selection: A scaling up approach. *Pattern Recognition 41* (2008), 2693–2709.

[55] GARCÍA, S., DERRAC, J., CANO, J., AND HERRERA, F. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE Transactions on Pattern Analysis and Machine Intelligence 34*, 3 (2012), 417–435.

[56] GARCÍA, S., FERNÁNDEZ, A., LUENGO, J., AND HERRERA, F. Advanced nonparametric tests for multiple comparisons in the design of experiments

in computational intelligence and data mining: Experimental analysis of power. *Information Sciences 180* (2010), 2044–2064.

[57] GARCÍA, S., AND HERRERA, F. Evolutionary under-sampling for classification with imbalanced data sets: Proposals and taxonomy. *Evolutionary Computation 17*, 3 (2009), 275–306.

[58] GARCÍA, S., AND HERRERA, F. Evolutionary undersampling for classification with imbalanced datasets: Proposal and taxonomy. *Evolutionary Computation 17*, 3 (2009), 275–306.

[59] GARCÍA, S., LUENGO, J., SÁEZ, J. A., LÓPEZ, V., AND HERRERA, F. A survey of discretization techniques: Taxonomy and empirical analysis in supervised learning. *IEEE Transactions on Knowledge and Data Engineering 25*, 4 (2013), 734–750.

[60] GARCÍA-OSORIO, C., DE HARO-GARCÍA, A., AND GARCÍA-PEDRAJAS, N. Democratic instance selection: A linear complexity instance selection algorithm based on classifier ensemble concepts. *Artificial Intelligence 174* (2010), 410–441.

[61] GARCÍA-PEDRAJAS, N. Constructing ensembles of classifiers by means of weighted instance selection. *IEEE Transactions on Neural Networks 20*, 2 (2009), 258–277.

[62] GATES, G. The reduced nearest neighbor rule. *IEEE Transactions on Information Theory 18*, 3 (1972), 431–433.

[63] GOLDBERG, D. E. *The Design of Innovation: Lessons from and for Competent Genetic Algorithms*. Kluwer Academic Publishers, 2002.

[64] GOURRAUD, P., GININ, E., AND CAMBON-THOMSEN, A. Handling missing values in population data: Consequences for maximum likelihood estimation of haplotype frequencies. *European Journal of Human Genetics 12*, 10 (2004), 805–812.

[65] GRECO, S., INUIGUCHI, M., AND SLOWINSKI, R. Fuzzy rough sets and multiple premise gradual decision rules. *International Journal of Approximate Reasoning 41* (2005), 179–211.

[66] GRZYMALA-BUSSE, J. On the unknown attribute values in learning from examples. In *Proceedings of the 6th International Symposium on Methodologies For Intelligent Systems* (1991), pp. 368–377.

[67] GRZYMALA-BUSSE, J., GOODWIN, L., GRZYMALA-BUSSE, W., AND ZHENG, X. Handling missing attribute values in preterm birth data sets. In

Proceedings of the 10th International Conference of Rough Sets and Fuzzy Sets and Data Mining and Granular Computing (2005), pp. 342–351.

[68] GUYON, I., AND ELISSEEFF, A. An introduction to variable and feature selection. *Journal of Machine Learning Research 3* (2003), 1157–1182.

[69] HAN, H., WANG, W., AND MAO, B. Borderline-smote: A new over-sampling method in imbalanced data sets learning. In *Advances in Intelligent Computing*, D.-S. Huang, X.-P. Zhang, and G.-B. Huang, Eds., vol. 3644. 2005, pp. 878–887.

[70] HAND, D., AND VINCIOTTI, V. Choosing k for two-class nearest neighbor classifiers with unbalanced classes. *Pattern recognition letters 24*, 9 (2003), 1555–1562.

[71] HANS, C. Bayesian lasso regression. *Biometrika 96*, 4 (2009), 835–845.

[72] HART, P. E. The condensed nearest neighbor rule. *IEEE Transactions on Information Theory 18* (1968), 515–516.

[73] HASTIE, T., AND TIBSHIRANI, R. Classification by pairwise coupling. *Annals of Statistics 26*, 2 (1998), 451–471.

[74] HATTORI, K., AND TAKAHASHI, M. A new edited k-nearest neighbor rule in the pattern classification problem. *Pattern Recognition 32* (2000), 521–528.

[75] HO, S., LIU, C., AND LIU, S. Design of an optimal nearest neighbor classifier using an intelligent genetic algorithm. *Pattern Recognition Letters 23* (2002), 1495–1503.

[76] HOLM, S. A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics 6* (1979), 65–70.

[77] HONG, T., LIOU, Y., AND WANG, S. Fuzzy rough sets with hierarchical quantitative attributes. *Expert Systems and Applications 36*, 3 (2009), 6790–6799.

[78] HU, Q., ZHANG, L., AN, S., ZHANG, D., AND YU, D. On robust fuzzy rough set models. *IEEE transactions on Fuzzy Systems 20*, 4 (2012), 636–651.

[79] JENSEN, R., AND CORNELIS, C. A new approach to fuzzy-rough nearest neighbour classification. In *Proceedings of the 6th International Conference on Rough Sets and Current Trends in Computing* (2008), pp. 310–319.

[80] JENSEN, R., AND CORNELIS, C. Fuzzy-rough instance selection. In *Proceedings of the 19th International Conference on Fuzzy Systems* (2010), pp. 1–7.

[81] JENSEN, R., AND CORNELIS, C. Fuzzy-rough nearest neighbour classification and prediction. *Theoretical Computer Science 412* (2011), 5871–5884.

[82] JENSEN, R., CORNELIS, C., AND SHEN, Q. Hybrid fuzzy-rough rule induction and feature selection. In *Proceedings of the IEEE International Conference on Fuzzy Systems* (2009), pp. 1151–1156.

[83] JENSEN, R., AND SHEN, Q. Fuzzy-rough feature significance for decision trees. In *Proceedings of the 2005 UK Workshop on Computational Intelligence* (2005), pp. 89–96.

[84] JENSEN, R., AND SHEN, Q. Fuzzy-rough sets assisted attribute selection. *IEEE Transactions on Fuzzy Systems 15*, 1 (2007), 73–89.

[85] JENSEN, R., AND SHEN, Q. New approaches to fuzzy-rough feature selection. *IEEE Transactions onFuzzy Systems 17*, 4 (2009), 824–838.

[86] KARABOGA, D., AND BASTURK, B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (abc) algorithm. *Journal of Global Optimization 39*, 3 (2007), 459–471.

[87] KELLER, J. M., GRAY, M. R., AND GIVENS, J. R. A fuzzy k-nearest neighbor algorithm. *IEEE Transactions on Systems, Man, and Cybernetics 15* (1985), 580–585.

[88] KENNEDY, J., AND EBERHART, R. Particle swarm optimization. In *Proceedings of the IEEE International Conference on Neural Networks* (1995), vol. 4, pp. 1942–1948.

[89] KERBER, R. Chimerge: Discretization of numeric attributes. In *Proceedings of the National Conference on Artifical Intelligence American Association for Artificial Intelligence* (1992), pp. 123–128.

[90] KLEINBAUM, D. G., KUPPER, L. L., AND MULLER, K. E., Eds. *Applied Regression Analysis and Other Multivariable Methods*. 1988.

[91] KOHAVI, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the 14th international joint Conference on Artificial intelligence* (1995), pp. 1137–1143.

[92] KOHAVI, R., AND JOHN, G. H. Wrappers for feature subset selection. *Artificial Intelligence 97*, 12 (1997), 273 – 324.

[93] KOTSIANTIS, S. Decision trees: a recent overview. *Artificial Intelligence Review 39*, 4 (2013), 261–283.

[94] KUBAT, M., HOLTE, R. C., AND MATWIN, S. Machine learning for the detection of oil spills in satellite radar images. *Machine Learning 30*, 2-3 (1998), 195–215.

[95] KUBAT, M., AND MATWIN, S. Addressing the curse of imbalanced training sets: one-sided selection. In *Proceedings of the 14th International Conference on Machine Learning* (1997), pp. 179–186.

[96] KUNCHEVA, L., AND WHITAKER, C. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine Learning 51*, 2 (2003), 181–207.

[97] KUNCHEVA, L. I. Editing for the k-nearest neighbors rule by a genetic algorithm. *Pattern Recognition Letters 16*, 8 (1995), 809–814.

[98] KUNCHEVA, L. I., AND JAIN, L. C. Nearest neighbor classifier: Simultaneous editing and feature selection. *Pattern Recognition Letters 20* (1999), 1149–1156.

[99] LANZI, P. Fast feature selection with genetic algorithms: A filter approach. In *Proceedings of the IEEE International Conference on Evolutionary Computation* (1997), pp. 537–540.

[100] LARSON, S. C. The shrinkage of the coefficient of multiple correlation. *Journal of Educational Psychology 22*, 1 (1931), 45–55.

[101] LAURIKKALA, J. Improving identification of difficult small classes by balancing class distribution. In *Proceedings of the 8th Conference on AI in Medicine in Europe* (2001), pp. 63–66.

[102] LI, F., MIN, F., AND LIU, Q. Intra-cluster similarity index based on fuzzy rough sets for fuzzy c-means algorithm. In *Proceedings of the 3th International Conference on Rough Sets and Knowledge Technology*, vol. 5009. 2008, pp. 316–323.

[103] LIU, H., HUSSAIN, F., TAN, L., AND DASH, M. Discretization: An enabling technique. *Data Mining and Knowledge Discovery 6*, 4 (2002), 393–423.

[104] LIU, H., MOTODA, H., SETIONO, R., AND ZHAO, Z. Feature selection: An ever evolving frontier in data mining. In *Proceedings of the 4th Workshop on Feature Selection in Data Mining* (2010), pp. 4–13.

[105] LIU, W., YAO, J., AND YAO, Y. Rough approximations under level fuzzy sets. In *Proceedings of the Conference on Rough Sets and Current Trends in Computing* (2004), pp. 78–83.

[106] LIU, Y., ZHOU, Q., RAKUS-ANDERSSON, E., AND BAI, G. 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* (2012), pp. 63–70.

[107] LOZANO, M. T., SANCHEZ, J. S., AND PLA, F. Using the geometrical distribution of prototypes for training set condensing. In *Proceedings of the 10th Conference of the Spanish Association for Artificial Intelligence and the 5th Conference on Technology Transfer* (2003), pp. 618–627.

[108] LUENGO, J., GARCÍA, S., AND HERRERA, F. A study on the use of statistical tests for experimentation with neural networks: Analysis of parametric test conditions and non-parametric tests. *Expert Systems with Applications 36* (2009), 7798–7808.

[109] MAC PARTHALAIN, N., JENSEN, R., SHEN, Q., AND ZWIGGELAAR, R. Fuzzy-rough approaches for mammographic risk analysis. *Intelligent Data Analysis 13*, 2 (2010), 225–244.

[110] MAJI, P. Fuzzy rough supervised attribute clustering algorithm and classification of microarray data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 41*, 1 (2011), 222 –233.

[111] MAJNIK, M., AND BOSNIĆ, Z. Roc analysis of classifiers in machine learning: A survey. *Intelligent Data Analysis 17*, 3 (2011), 531–558.

[112] MARCHIORI, E. Hit miss networks with applications to instance selection. *Journal of Machine Learning Research 9* (2008), 997–1017.

[113] MARGINEANTU, D. Class probability estimation and cost-sensitive classification decisions. In *Proceedings of the European Conference on Machine Learning* (2002), pp. 270–281.

[114] MENA, L., AND GONZALEZ, J. A. Machine learning for imbalanced datasets: Application in medical diagnostic. In *Proceedings of the 19th International Florida Artificial Intelligence Research Society Conference* (2006), pp. 574–579.

[115] MIESZKOWICZ-ROLKA, A., AND ROLKA, L. *Variable Precision Fuzzy Rough Sets*. Springer-Verlag, 2004, pp. 144–160.

[116] MIESZKOWICZ-ROLKA, A., AND ROLKA, L. Fuzzy rough approximations of process data. *International Journal of Approximate Reasoning 49* (2008), 301–315.

[117] MORENO-TORRES, J. G., RAEDER, T., ALAÍZ-RODRÍGUEZ, R., CHAWLA, N. V., AND HERRERA, F. A unifying view on dataset shift in classification. *Pattern Recognition 45*, 1 (2012), 521–530.

[118] MORENO-TORRES, J. G., RAEDER, T., RODRIGUEZ, R. A., CHAWLA, N. V., AND HERRERA, F. A unifying view on dataset shift in classification. *Pattern recognition 45*, 1 (2012), 521–530.

[119] MORENO-TORRES, J. G., SÁEZ, J. A., AND HERRERA, F. Study on the impact of partition-induced dataset shift on k -fold cross-validation. *IEEE Transactions on Neural Networks and Learning Systems 23*, 8 (2012), 1304–1312.

[120] N. MAC PARTHALAIN, AND JENSEN, R. Simultaneous feature and instance selection using fuzzy-rough bireducts. In *Proceedings of the IEEE International Conference on Fuzzy Systems* (2013).

[121] NICHOLS, T., AND HAYASAKA, S. Controlling the familywise error rate in functional neuroimaging: a comparative review. *Statistical Methods in Medical Research 12* (2003), 419–446.

[122] NOCEDAL, J., AND WRIGHT, S. *Numerical Optimization*. Springer, 2006.

[123] PARK, S. H., AND FÜRNKRANZ, J. Efficient pairwise classification. In *Proceedings of the 18th European Conference on Machine Learning* (2007), pp. 658–665.

[124] PAWLAK, Z. Rough sets. *International Journal of Computer Information Science 11* (1982), 341–356.

[125] PLATT, J. *Fast Training of Support Vector Machines using Sequential Minimal Optimization*. MIT Press, 1998, pp. 185–208.

[126] PLATT, J. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in large margin classifiers* (1999), pp. 61–74.

[127] POLI, R., KENNEDY, J., AND BLACKWELL, T. Particle swarm optimization. *Swarm Intelligence 1*, 1 (2007), 33–57.

[128] Q. HE, C. W. Membership evaluation and feature selection for fuzzy support vector machine based on fuzzy rough sets. *Soft Computing 15*, 6 (2011), 1105–1114.

[129] QU, Y., SHANG, C., SHEN, Q., PARTHALAIN, N. M., AND WU, W. Kernel-based fuzzy-rough nearest neighbour classification. In *Proceedings of the IEEE International Conference on Fuzzy Systems* (2011), pp. 1523–1529.

[130] QUINLAN, J. *C4.5: Programs for Machine Learning*. Morgan Kauffman, 1993.

[131] RAMENTOL, E., CABALLERO, Y., BELLO, R., AND HERRERA, F. 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* (2011), 1–21.

[132] RAMENTOL, E., VERBIEST, N., BELLO, R., CABALLERO, Y., CORNELIS, C., AND HERRERA, F. Smote-frst: A new resampling method using fuzzy rough set theory. In *10th International FLINS Conference on Uncertainty Modeling in Knowledge Engineering and Decision Making* (2012).

[133] RIFKIN, R., AND KLAUTAU, A. In defence of one-vs-all classification. *Journal of Machine Learning Research 5* (2004), 101–141.

[134] RIQUELME, J., AGUILAR-RUIZ, J., AGUILAR-RUIZ, J. S., AND TORO, M. Finding representative patterns with ordered projections. *Pattern Recognition 36*, 4 (2003), 1009–1018.

[135] ROKACH, L. Ensemble-based classifiers. *Artificial Intelligence Review 33* (2010), 1–39.

[136] RUSTAGI, J. *Optimization Techniques in Statistics*. Academic Press, 1994.

[137] SAEYS, Y., ABEEL, T., AND PEER, Y. Robust feature selection using ensemble feature selection techniques. In *Machine Learning and Knowledge Discovery in Databases*. 2008, pp. 313–325.

[138] SAEYS, Y., INZA, I., AND LARRAÑAGA, P. A review of feature selection techniques in bioinformatics. *Bioinformatics 23*, 19 (2007), 2507–2517.

[139] SALIDO, J. M. F., AND MURAKAMI, S. On β-precision aggregation. *Fuzzy Sets and Systems 139* (2003), 547–558.

[140] SALIDO, J. M. F., AND MURAKAMI, S. Rough set analysis of a general type of fuzzy data using transitive aggregations of fuzzy similarity relations. *Fuzzy Sets and Systems 139* (2003), 635–660.

[141] SÁNCHEZ, J., PLA, F., AND FERRI, F. Prototype selection for the nearest neighbor rule through proximity graphs. *Pattern Recognition Letters 18* (1997), 507–513.

[142] SARKAR, M. Fuzzy-rough nearest neighbor algorithms in classification. *Fuzzy Sets and Systems 158*, 19 (2007), 2134–2152.

[143] SEIFFERT, C., KHOSHGOFTAAR, T., HULSE, J. V., AND NAPOLITANO, A. Rusboost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans 40*, 1 (2010), 185–197.

[144] SHESKIN, D. J. *Handbook of Parametric and Nonparametric Statistical Procedures, 4th ed.* Chapman and Hall/CRC, 2006.

[145] SHIMODAIRA, H. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of Statistical Planning and Inference 90*, 2 (2000), 227–244.

[146] SKALAK, D. B. Prototype and feature selection by sampling and random mutation hill climbing algorithms. In *Proceedings of the Eleventh International Conference on Machine Learning* (1994), pp. 293–301.

[147] SLEZAK, D., AND JANUSZ, A. Ensembles of bireducts: Towards robust classification and simple representation. In *Proceedings of the 3th Conference on Future Generation Information Technology* (2011), pp. 64–77.

[148] STEFANOWSKI, J., AND WILK, S. Selective pre-processing of imbalanced data for improving classification performance. In *Proceedings of the 10th International Conference in Data Warehousing and Knowledge Discovery* (2008), pp. 283–292.

[149] STONE, M. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistics Society 36* (1974), 111–147.

[150] SUN, Y., WONG, A., AND KAMEL, M. Classification of imbalanced data: A review. *International Journal of Pattern Recognition and Artificial Intelligence 23*, 4 (2009), 687–719.

[151] SVETNIK, V., LIAW, A., TONG, C., CULBERSON, J. C., SHERIDAN, R. P., AND FEUSTON, B. P. Random forest: a classification and regression tool for compound classification and qsar modeling. *Journal of Chemical Information and Computer Sciences 43*, 6 (2003), 1947–1958.

[152] TING, K. An instance-weighting method to induce cost-sensitive trees. *IEEE Transactions on Knowledge and Data Engineering 14*, 3 (2002), 659–665.

[153] TING-FAN, W., CHIH-JEN, L., AND RUBY, W. C. Probability estimates for multiclass classification by pairwise coupling. *Journal of Machine Learning Research 5* (2004), 975–1005.

[154] TOMEK, I. An experiment with the edited nearest-neighbor rule. *IEEE Transactions on Systems, Man and Cybernetics 6*, 6 (1976), 448–452.

[155] TOMEK, I. Two modifications of cnn. *IEEE Transactions on Systems and Man and Cybernetics 6* (1976), 769–772.

[156] TRIGUERO, I., DERRAC, J., GARCIA, S., AND HERRERA, F. A taxonomy and experimental study on prototype generation for nearest neighbor classification. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 42*, 1 (Jan 2012), 86–100.

[157] TSANG, E., ZHAO, S., AND LEE, J. Rule induction based on fuzzy rough sets. In *Proceedings of the International Conference on Machine Learning and Cybernetics* (2007), pp. 3028–3033.

[158] V. LÓPEZ, A. FERNÁNDEZ, J. M., AND HERRERA, F. An insight into classification with imbalanced data: Empirical results and current trends on using data intrinsic characteristics. *Information Sciences 250* (2012), 113–141.

[159] VERBIEST, N., CORNELIS, C., AND HERRERA, F. Frps: a fuzzy rough prototype selection method. *Pattern Recognition 46*, 10 (2013), 2770–2782.

[160] VERBIEST, N., CORNELIS, C., AND HERRERA, F. 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* (2013), pp. 180–190.

[161] VERBIEST, N., CORNELIS, C., AND JENSEN, R. Fuzzy rough positive region-based nearest neighbour classification. In *Proceedings of the 20th International Conference on Fuzzy Systems* (2012), pp. 1961–1967.

[162] VERBIEST, N., CORNELIS, C., AND JENSEN, R. Quality, frequency and similarity based fuzzy nearest neighbor classification. In *Proceedings of the 21st International Conference on Fuzzy Systems* (2013).

[163] VERBIEST, N., RAMENTOL, E., CORNELIS, C., AND HERRERA, F. 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* (2012), pp. 169–178.

[164] VEROPOULOS, K., CAMPBELL, C., AND CRISTIANINI, N. Controlling the sensitivity of support vector machines. In *Proceedings of the International Joint Conference on Artificial Intelligence* (1999), pp. 55–60.

[165] WANG, S., AND YAO, X. Diversity analysis on imbalanced data sets by using ensemble models. In *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining* (2009), pp. 32–331.

[166] WANG, X., TSANG, E., ZHAO, S., CHEN, D., AND YEUNG, D. Learning fuzzy rules from fuzzy samples based on rough set technique. *InformationSciences 177*, 20 (2007), 4493–4514.

[167] WILCOXON, F. Individual comparisons by ranking methods. *Biometrics Bulletin 6* (1945), 80–83.

[168] WILSON, D. L. Asymptotic properties of nearest neighbor rules using edited data. *IEEE Transactions on Systems, Man and Cybernetics 2*, 3 (1972), 408–421.

[169] WILSON, D. R., AND MARTINEZ, T. R. Reduction techniques for instance-based learning algorithms. *Machine Learning 38* (2000), 257–286.

[170] WOLPERT, D. H. The supervised learning no-free-lunch theorems. In *Proceedings of the 6th Online World Conference on Soft Computing in Industrial Applications* (2001).

[171] WU, W. Z., MI, J., AND ZHANG, W. X. Generalized fuzzy rough sets. *Information Sciences 151* (2003), 263–282.

[172] WU, W. Z., AND ZHANG, W. X. Constructive and axiomatic approaches of fuzzy approximation operators. *Information Sciences 159* (2004), 233–254.

[173] XU, R., AND II, D. W. Survey of clustering algorithms. *IEEE Transactions on Neural Networks 16* (2005), 645–678.

[174] XUE, S., YANG, M., LI, C., AND NIE, J. Meteorological Prediction Using Support Vector Regression with Genetic Algorithms. In *Proceedings of the International Conference on Information Science and Engineering* (2009).

[175] XUE, Z., AND LIU, W. A fuzzy rough support vector regression machine. In *Proceedings of the 9th International Conference on Fuzzy Systems and Knowledge Discovery* (2012), pp. 840 –844.

[176] YAGER, R. On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Transactions on Systems, Man and Cybernetics 18*, 1 (1988), 183 –190.

[177] YANG, J., AND HONAVAR, V. Feature subset selection using a genetic algorithm. *IEEE Intelligent Systems 13*, 2 (1998), 44–49.

[178] YAO, Y. Combination of rough and fuzzy sets based on α-level sets. In *Rough Sets and Data Mining: Analysis for Imprecise Data* (1997), T. Lin and N. Cercone, Eds., pp. 301–321.

[179] YEN, S., AND LEE, Y. Under-sampling approaches for improving prediction of the minority class in an imbalanced dataset. In *Proceedings of the International Conference on Intelligent Computing* (2006), pp. 731–740.

[180] YOON, K., AND KWEK, S. 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* (2005), pp. 303–308.

[181] ZADEH, L. A. Fuzzy sets. *Information and Control 8* (1965), 338–353.

[182] ZAR, J. H. *Biostatistical Analysis*. Prentice Hall, 1999.

[183] ZECHNER, M., AND GRANITZER, M. A competitive learning approach to instance selection for support vector machines. In *Knowledge Science, Engineering and Management*. 2009, pp. 146–157.

[184] ZENG, X., AND MARTINEZ, T. R. Distribution-balanced stratified cross-validation for accuracy estimation. *Journal of Experimental and Theoretical Artificial Intelligence 12*, 1 (2000), 1–12.

[185] ZHAI, J. Fuzzy decision tree based on fuzzy-rough technique. *Soft Computing 15*, 6 (2011), 1087–1096.

[186] ZHANG, Z., CHEN, D., HE, Q., AND H.WANG. Least squares support vector machines based on fuzzy rough set. In *Proceedings of the IEEE International Conference on Systems Man and Cybernetics* (2010), pp. 3834 –3838.

[187] ZHAO, K.-P., ZHOU, S.-G., GUAN, J.-H., AND ZHOU, A.-Y. C-pruner: an improved instance pruning algorithm. In *Proceedings of the International Conference on Machine Learning and Cybernetics* (2003), pp. 94–99.

[188] ZHAO, S., TSANG, E., AND CHEN, D. The model of fuzzy variable precision rough sets. *IEEE transactions on Fuzzy Systems 17*, 2 (2009), 451–467.

[189] ZHAO, S., TSANG, E., CHEN, D., AND X.Z.WANG. Building a rule-based classifier - a fuzzy-rough set approach. *IEEE Transactions on Knowledge and Data Engineering 22*, 5 (2010), 624–638.

[190] ZHU, X., AND WU, X. Class noise vs attribute noise: a quantitative study of their impacts. *Artificial Intelligence Review 22* (2004), 177–210.