Oliver Urs
Lenz

FUZZY ROUGH NEAREST NEIGHBOUR CLASSIFICATION ON REAL-LIFE DATASETS

Oliver Urs Lenz

Fuzzy rough nearest neighbour classification on real-life datasets

Thesis submitted in partial satisfaction of the requirements for the degree of Doctor in Computer Science

Advisors: Dr Chris Cornelis, Dr Daniel Peralta

UNIVERSITEIT GENT

Vakgroep
Toegepaste Wiskunde, Informatica en Statistiek

Typeset by the author using pdfLaTeX. Typeface: TeXGyrePagellaX (based on Palatino). Cover typeface: Roboto Slab.

Published by the author, Ghent, January 2023.

Printed by University Press, Wachtebeke. Print run: 40 copies.

Meinen Eltern:

Sabine Wolff-Lenz & Harald Lenz (†)

Contents

Introduction

Imagine that you are invited to listen to a recording of a person speaking English, and to guess, on the basis of their accent, where they are from. One (subconscious) strategy that you might try is to mentally compare the person in the recording with (your memory of) various other people speaking English, and to identify the closest match. If this is your strategy, you are applying nearest neighbour (NN) classification.

The process just described is extremely intuitive, and Pelillo (2014) has claimed that it has already been formulated as early as the 1030s by Ibn al-Haytham. The simplicity of NN classification contrasts with much more complex algorithms like neural networks, which have gained a reputation for being inscrutable: a powerful magical black box that transforms raw data into conclusions that are generally correct, but which may also fail in mysterious ways.

As NN classification is so simple, one could be forgiven for thinking that there is not much left to say about it. But that is not quite true. How many neighbours should we consider, and should we consider them equally? In order to be able to identify neighbours in the first place, we need a definition of distance. It turns out that there are multiple plausible choices, so which distance measure should we select, and how should we scale our data?

What is more, Jensen & Cornelis (2008) have proposed a variant of NN, known as *fuzzy rough nearest neighbours* (FRNN). It is still under active development, and is not yet understood as well as NN classification. For FRNN, we face the same questions as listed in the previous paragraph, and in addition we may wonder whether FRNN offers any advantages over NN.

In the machine learning literature, it is typical to establish results theoretically, with carefully controlled simulations or with individual case studies. This is perhaps not surprising, given that machine learning is an outgrowth of statistics. But it does mean that the real-life consequences of these results are not always clear.

Part of the reason for these approaches may also be practical. Real-life datasets are a limited resource that can take a lot of time and effort to compile. However, this has started to change with the advent of dataset

repositories, which are filled by generous donors (typically authors of specific case studies). Perhaps the most important of these is the UCI machine learning repository (Dua & Graff 2019), established in 1987. Using datasets from this repository still requires a certain amount of preparatory work, since only some of the datasets are suitable for any given task, the datasets come in different formats, and they may require various forms of preprocessing. These challenges also entail that a practitioner must develop a superficial understanding of each dataset that they want to use, from the (often terse) documentation and relevant publications.

Still, the UCI repository represents an invaluable resource. The goal of this thesis is to demonstrate how FRNN and some related algorithms can be applied practically, and throughout we will use a curated collection of real-life classification problems from the UCI repository to empirically evaluate proposals.

In Section 0.1, we will provide a number of preliminary definitions that will be used throughout this thesis. After that, we will sketch a brief outline of the contents of this thesis in Section 0.2.

0.1 Preliminaries

In this section, we will give formal definitions of fuzzy sets, datasets and classification, and explain how we evaluate classification performance throughout this thesis.

Fuzzy sets

Fuzzy rough nearest neighbour classication is based on concepts from fuzzy set theory (Bellman et al 1964; Zadeh 1965). However, we will only require a few elementary definitions.

Fuzzy sets generalise classical ('crisp') subsets. Recall that a subset Y of a set X is equivalent to an indicator function on X, which takes the value 1 for all elements contained in Y, and 0 for all other elements. Fuzzy sets generalise these indicator functions to allow partial membership:

0.1 Definition (Fuzzy set). Let X be a set. A *fuzzy set* in X is a map $X \longrightarrow [0, 1]$. We use $\mathcal{F}(X)$ to denote the set of all fuzzy sets in X.

Recall that a binary relation on a set X is a subset of $X \times X$. Analogously, a binary fuzzy relation on X is a fuzzy subset of $X \times X$. We will use one specific type of binary fuzzy relation:

0.2 Definition (Tolerance relation). A *tolerance relation* on a set X is a fuzzy relation on X that is reflexive and symmetric.

Note that this definition is quite flexible, and there is little conceptual difference with similarity measures that take values in $[0, 1]$.

Fuzzy sets can be used to extend logic to partial truth values in $[0, 1]$. This requires a fuzzification of logical connectives, but there is no single solution for this. Instead, for each classical connective, we define a class of corresponding fuzzy connectives, and the practitioner has to make a choice. In this thesis, we will only encounter t-norms and fuzzy implications, which generalise, respectively, the conjunction $\wedge$ and the implication $\implies$.

0.3 Definition (T-norm). A *t-norm* (or *triangular norm*) is an associative, commutative and monotonically increasing binary operation on $[0, 1]$ for which 1 is an identity element.

0.4 Definition (Fuzzy implication). A *fuzzy implication* is a binary operation I on $[0, 1]$ that is monotonically decreasing in its first argument and monotonically increasing in its second argument, and for which $I(0, 0) = I(0, 1) = I(1, 1) = 1$ and $I(1, 0) = 0$.

The last two conditions ensure that a fuzzy implication reduces to the implication on $\{0, 1\}$. Likewise, because a t-norm T is increasing, $T(0, 0) = T(0, 1) = 0$, so it restricts to the conjunction on $\{0, 1\}$.

Datasets

Nearest neighbour algorithms and other classical machine learning approaches are most easily applied to tabular data. This consists of a varying number of rows, one for each record (or *instance*), and a fixed number of columns, corresponding to the attributes of the records. Attributes are typically either numerical (taking a value in $\mathbb{R}$) or categorical. In the latter case, the set of values is deemed not to have any internal structure, in that no two values are more or less similar than any other two values. Attributes with only two possible values are known as binary attributes, and can be considered as either numerical or categorical attributes.

We can formalise the concept of a dataset with the following definition:

0.5 Definition (Dataset). An *attribute space* $A = \prod A_i$ is a finite product of attributes A_i, which are either copies of $\mathbb{R}$ (*numerical attributes*) or finite sets of values V (*categorical attributes*). A *dataset* is a finite multisubset X of an attribute space A. The elements of X are called *records* or *instances*.

We define X as a multisubset to allow for the possibility of duplicate records, but we will write simply $X \subset A$.

Many algorithms are only defined for numerical data, and one popular solution, perhaps first documented by Suits (1957) (but "not new" even then), is to transform a categorical attribute into a series of binary *dummy variables* that can be represented numerically. Each of these dummy variables indicates whether a different value of the categorical attribute applies. Since each record can only take one value, only one of the dummy variables will take a value of 1, while the rest remain 0. Therefore, this approach is known as *one-hot* encoding:

0.6 Definition (One-hot encoding). Let V be a categorical attribute. For a chosen order $V = (v_1, v_2, \ldots, v_p)$, its *one-hot encoding* is the map $V \longrightarrow \mathbb{R}^p$ that sends v_i to the standard basis vector $\mathbf{e}_i = \langle 0, \ldots, 0, 1, 0, \ldots, 0 \rangle$ for all $i \leq p$.

Thus, if we apply one-hot encoding, we obtain a numerical dataset $X \subset \mathbb{R}^m$ for some $m > 0$.

Classification

One way to characterise the field of machine learning is by the different tasks it is applied to. In many of these tasks, covered by the umbrella term of prediction, we want to predict some unknown property, on the basis of some other properties that are known, as well as a *training set* of examples for which we do know the property to be predicted. In this thesis, we are mostly concerned with the task of classification, for which the unknown property is a categorical attribute.

Formally, classification can be defined as follows:

0.7 Definition (Classification). A *classification dataset* is a dataset $X \subset A$ together with a map $X \longrightarrow C$. The elements $C_i \in C$ are the decision classes of X, and we can identify C_i with its preimage in X, such that we obtain a partition $X = \bigsqcup C$. A *classifier* is an algorithm that takes a classification dataset and returns a function $A \longrightarrow \mathcal{F}(C)$, the *classification model* of X.

Thus, we distinguish between a classifier (the algorithm) and its classification model (the application of the algorithm to a specific problem).

Classification models do not simply predict a single class, but a score in $[0, 1]$ for each class (indicating how likely that class is). The default way of discretising these scores into a single class prediction is to select the class with the highest score.[1] We can also normalise class scores

[1] In the event of a tie, we can choose randomly, select a class according to some order of preference, or adjust the classifier to obtain a new prediction.

such that they sum to 1 (if this isn't already the case), for example by dividing them by their sum.

Classification scores give analysts a complete picture on which to base their decisions. When predicting a single class, we want to avoid errors, but not all errors may carry the same cost in the real world. For example, it may be more harmful to predict that an ill person is healthy than vice-versa. An analyst can take this cost into account, and may want to predict a class even if its score is not the highest. Likewise, they may want to avoid predicting any single class when all of the class scores are low.

Classification performance

To evaluate a classification model $F(X)$ produced by a classifier F, we use a test set $Y \subset A$ for which we also know the class membership $Y \longrightarrow C$ (the *ground truth*). The best-known evaluation measure is *accuracy*, which requires that we discretise $F(X)(Y)$ into a set of predicted values in C; we obtain the accuracy score by counting the share of corresponding matches with the ground truth.

However, because it requires discretising the class scores, accuracy is not a good summary of the full discriminative ability of a classification model. Instead, we will mostly use the *area under the receiver operating characteristic* (AUROC, Bradley 1997; Hanley & McNeil 1982) throughout this thesis. For two decision classes C_1 and C_2, this corresponds to the probability that a classification model assigns a higher normalised score for C_1 to a random test record belonging to C_1 than to a random test record belonging to C_2. For multiclass problems, we will use the extension by Hand & Till (2001), which aggregates the AUROC corresponding to each pair of classes.

To evaluate the performance of a classifier F on a classification dataset X, we will use three procedures, ranked in order of increasing sophistication:

Train/test splitting Randomly split X into a training set X_{train} and a test set X_{test}, and use X_{test} to evaluate $F(X_{\text{train}})$.

Stratified k-fold cross-validation For some $k \geq 2$, select a random partition $X = \bigsqcup X_i$ into k equally-sized test sets with an equal proportion of each decision class, and use every X_i to evaluate the model $F(\bigcup_{j \neq i} X_j)$ of the corresponding training set. We thus obtain k performance measures, which we can summarise by taking the mean.

Leave-one-out validation Predict a score for each $x \in X$ using the model $F(X \setminus \{x\})$, and evaluate the resulting set of scores using the ground truth for X.

Leave-one-out validation is arguably the most elegant of these procedures, but it is only feasible if the score $F(X \setminus \{x\})(x)$ can be calculated efficiently. Throughout this thesis, we will mostly use stratified 5-fold cross-validation to evaluate classifiers, and apply leave-one-out validation in specific cases.

To compare two classifiers (or two variants of the same classifier), we will use a series of datasets and apply cross-validation to obtain a corresponding series of AUROC scores for each classifier. We evaluate the difference in performance by applying a one-sided Wilcoxon signed-rank test (Wilcoxon 1945), which produces a p-value in $[0, 1]$.[2] A value below 0.5 indicates that A performed better than B, and that the probability that B is really better than A (on other datasets) is no more than p. Vice versa, a value above 0.5 indicates that B performed better than A and that the probability that this is coincidence is no more than $1 - p$.

0.2 Outline

We will now give a brief overview of the contents of this thesis. It consists of the following four parts:

Part I We start by reviewing the existing definition of fuzzy rough sets, and proposing a reformulation of FRNN as a proper nearest neighbour algorithm, allowing it to be implemented and applied more efficiently (Chapter 1). We then review the family of Minkowski distance measures and a number of related concepts (Chapter 2), and present a systematic experiment to identify the best scale and distance measure for NN and FRNN classification, compare FRNN against NN, and determine good default choices for the number of neighbours k (Chapter 3).

Part II In this part, we explore how FRNN classification can be scaled to large datasets. We first present an implementation of FRNN that employs distributed computing (Chapter 4), and then argue that the computational complexity of FRNN can be reduced substantially through approximative nearest neighbour searches (Chapter 5).

Part III Next, we widen our view to consider data descriptors, a class of algorithms that model similarity with a decision class. We

[2] When comparing a group of approaches, another option is to apply a Friedman test on the mean ranks (Demšar 2006). However, the appropriateness of this test has been questioned by Benavoli et al (2016), because the resulting p-values may be unduly inflated or deflated by the selection of classifiers. Instead, Benavoli et al (2016) recommend pairwise Wilcoxon signed-rank tests followed by a correction for the family-wise error.

start by introducing the setting of one-class classification and propose our own data descriptor, average localised proximity (Chapter 6). We then evaluate the performance of a number of data descriptors with default hyperparameter values (Chapter 7) and with hyperparameter optimisation (Chapter 8). Finally, we re-analyse FRNN as a one-class classification ensemble and explore whether we can improve its performance by substituting different components (Chapter 9).

Part IV In the last part, we consider missing values, an issue that is frequently encountered with real-life datasets. We evaluate three approaches: missing-indicators, an old proposal from the literature (Chapter 10), interval-valued fuzzy sets, a solution specific to FRNN (Chapter 11), and polar encoding, a new proposal that represents missing values in a neutral manner and which can be used with various algorithms (Chapter 12).

We finish with a Conclusion, in which we summarise the results of this thesis and describe a number of avenues for future research.

Appendices In Appendix A, we describe the software library fuzzy-rough-learn. Appendix B contains an overview of the datasets used in this thesis. Lastly, some of the long tables with complete results have been relegated to Appendix C.

Part I

Numerical datasets

Chapter 1

Fuzzy rough nearest neighbour classification[1]

Fuzzy rough sets were first proposed by Dubois & Prade (1990) as a hybridisation of fuzzy and rough sets. Rough sets (Pawlak 1981, 1982) can be used to model the uncertainty stemming from conflicting information in categorical datasets. Fuzzy rough sets generalise rough sets to numerical datasets using concepts from fuzzy logic (Bellman et al 1964; Zadeh 1965), an extension of classical logic to $[0, 1]$-valued truth.

Fuzzy rough sets can be used to model concepts, analyse datasets and help decision-making. In particular, fuzzy rough sets have been integrated in machine learning algorithms for tasks such as feature selection, instance or prototype selection, classification and regression (Vluymans et al 2015b). Our primary interest in this and the following chapters is *fuzzy rough nearest neighbours* (FRNN) (Jensen & Cornelis 2008), a relatively straightforward classification algorithm that approximates each decision class with a so-called *upper* and *lower approximation* and predicts class membership on the basis of membership degrees in these approximations.

Like other lazy learners, FRNN does not require training and so can be applied directly to classify test instances with a training set. FRNN is also conceptually attractive because its predictions are directly interpretable. Upper approximation membership encodes to what extent a test instance is similar to the training instances of a class, and so possibly belongs to this class. Lower approximation membership encodes to what extent a test instance is not similar to the training instances of other classes, and hence should belong to this class.

In Section 1.1, we review the definitions of rough sets and fuzzy rough sets, and show how the latter have been made more robust through the incorporation of ordered weighted averaging (OWA) aggregation.

[1] This chapter is based on parts of Lenz et al (2019) and Lenz et al (2020b).

This involves the application of weight vectors, and the choice of these weight vectors offers a degree of flexibility. For example, because lower and upper approximations are calculated for each class, it is possible to use different types of weights for different classes. This idea has been applied successfully by Ramentol et al (2015) and subsequent studies (Vluymans et al 2016, 2018b) to imbalanced datasets, where a judicious choice of weights increases the signal of the minority class.

Then, in Section 1.2, we propose a number of changes that make the application of FRNN classification more practical. In particular, this involves the adaptation of OWA operators into weighted maxima and minima, which allows us to calculate upper and lower approximation membership using nearest neighbour queries.

We conclude this chapter in Section 1.3.

1.1 From rough sets to fuzzy rough sets with OWA operators

In this section, we will sketch the historical development of FRNN, starting with rough sets.

Rough sets

Rough sets (Pawlak 1981, 1982) approximate sets of records by way of some attributes defined over those records. Traditionally, rough sets are defined in terms of an information system:

1.1 Definition (Information system). An *information system* is a pair (X, A), where X is a finite set of *records*, and A a finite set of maps $a : X \longrightarrow V_a$, called *attributes*. The *indiscernibility* relation R in (X, A) is the equivalence relation $\{(x, y) \in X \times X | \forall a \in A : a(x) = a(y)\}$.

The equivalence classes of R consist of all records that are indiscernible in terms of their attribute values. Then a rough set is defined as follows:

1.2 Definition (Upper and lower approximations). Let (X, A) be an information system, and $C \subseteq X$ a subset. The *upper approximation* $\overline{C}$ and *lower approximation* $\underline{C}$ of C are defined by:

$$\begin{aligned} \overline{C} &:= \left\{ y \in X | \exists x \in C : y \sim_R x \right\} \\ \underline{C} &:= \left\{ y \in X | \nexists x \in X \setminus C : y \sim_R x \right\} \end{aligned} \tag{1.1}$$

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

The upper approximation of C contains all instances that are indiscernible from instances of C, while the lower approximation of C contains only those instances that are discernible from all instances not contained in C. They are the closure and interior of the quasi-discrete topology generated by R.

The traditional interpretation is that the upper approximation of C contains instances that, based on their attribute values, possibly belong to C, while the lower approximation of C contains instances that, based on their attribute values, necessarily belong to C.

Fuzzy rough sets

Upper and lower approximations were originally proposed for categorical data, and their applicability to numerical data is limited by the fact that they are based on exact equality of attribute values. In order to work with numerical data, we want to be able to use the fact that numerical attribute values can be more or less similar to each other. The solution, proposed by Dubois & Prade (1990), is to work with fuzzy subsets, and to replace R with a tolerance relation (Definition 0.2).

Note that the classical definition of upper and lower approximation can be reformulated as follows, if we consider R and C as indicator functions:

$$\begin{aligned} \overline{C}(y) &= \max_{x \in X}(R(y,x) \wedge C(x)), \\ \underline{C}(y) &= \min_{x \in X}(R(y,x) \implies C(x)). \end{aligned} \tag{1.2}$$

Thus, we can fuzzify upper and lower approximations by replacing the conjunction $\wedge$ and the implication $\implies$ by, respectively, a t-norm (Definition 0.3) and a fuzzy implication (Definition 0.4).

1.3 Definition (Upper and lower approximation). Let (X, A) be an information system. For a choice of a tolerance relation R on X such that $(\forall a \in A : a(x) = a(y)) \implies R(x,y) = 1$, a t-norm T and a fuzzy implication I, the *upper and lower approximation* $\overline{C}$ and $\underline{C}$ of a fuzzy subset C of X are are the fuzzy subsets of X defined by:

$$\begin{aligned} \overline{C}(y) &:= \max_{x \in X}(T(R(y,x), C(x)), \\ \underline{C}(y) &:= \min_{x \in X}(I(R(y,x), C(x)). \end{aligned} \tag{1.3}$$

Analogous to rough sets, a fuzzy rough set in (X, A) is a pair $(\overline{C}, \underline{C})$ for some fuzzy subset $C \subseteq X$.

Table 1.1: Types of weight vectors w^k that can be used for the upper approximation. For the lower approximation, the dual weight vectors can be used.

Name	w_i^k	Example: w^4
Strict	$\begin{cases} 1 & i = 1 \\ 0 & i > 1 \end{cases}$	$\langle 1, 0, 0, 0 \rangle$
Linear	$\frac{2(k+1-i)}{k(k+1)}$	$\left\langle \frac{4}{10}, \frac{3}{10}, \frac{2}{10}, \frac{1}{10} \right\rangle$
Reciprocally linear	$\frac{1}{i \cdot \sum_{i \leq k} \frac{1}{i}}$	$\left\langle \frac{12}{25}, \frac{12}{50}, \frac{12}{75}, \frac{12}{100} \right\rangle$
Exponential	$\frac{2^{k-i}}{2^k - 1}$	$\left\langle \frac{8}{15}, \frac{4}{15}, \frac{2}{15}, \frac{1}{15} \right\rangle$

OWA operators

A practical limitation of Definition 1.3 is that the upper and lower approximation membership of a given y are each completely determined by a single $x \in X$, because of the max and min operators. The solution has been to instead use different operators that approximate max and min but also take into account other records in X. A popular choice has been to use ordered weighted averaging (OWA) operators (Cornelis et al 2010; Yager 1988), this has been shown to generally produce better results than a number of alternative proposals (D'eer et al 2015).

1.4 Definition (OWA operator). Let w be a *weight vector* of n values in $[0, 1]$ that sum to 1. The *OWA operator* owa_w induced by w acts on any collection of n values by ordering these values in descending order and taking the inner product with w. We say that two weight vectors are *dual* if they have inversely ordered matching coefficients.

A number of possible sets of weight vectors are listed in Table 1.1 (Vluymans et al 2019). Strict weights represent the trivial choice, for which we recover max and min. The other weights are displayed visually in Figure 1.1 for $k = 10$.

Thus, we obtain the following revised definition:

1.5 Definition (Upper and lower approximation). Let (X, A) be an information system. For a choice of a tolerance relation R on X such that $(\forall a \in A : a(x) = a(y)) \implies R(x, y) = 1$, a t-norm T, a fuzzy implication I and weight vectors $\overline{w}$ and $\underline{w}$ of length $|X|$, the *upper and lower approximation* $\overline{C}$ and $\underline{C}$ of a fuzzy subset C of X are the fuzzy

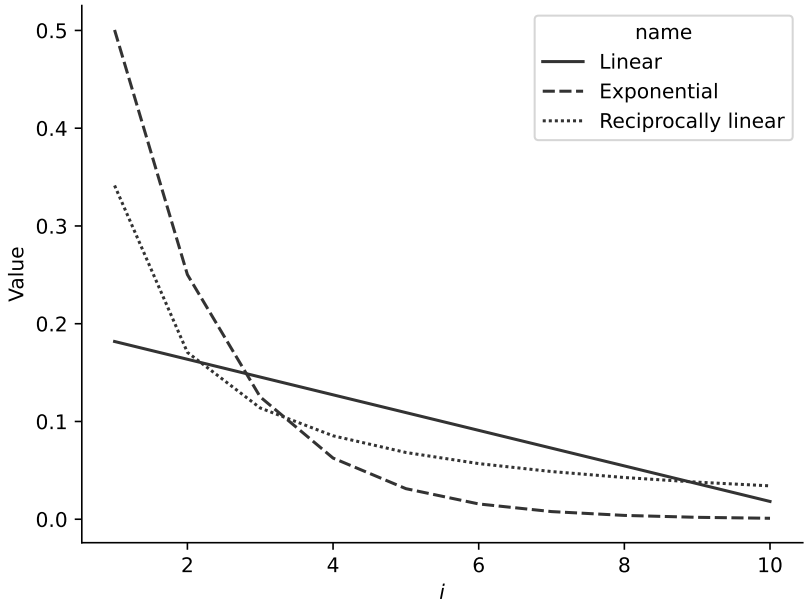


Figure 1.1: Values of the weight vectors from Table 1.1 for $k = 10$.

subsets of X defined by:

$$\begin{aligned} \overline{C}(y) &:= \operatorname*{owa}_{\overline{w}}\big\{T(R(y,x),C(x))|x \in X\big\}; \\ \underline{C}(y) &:= \operatorname*{owa}_{\underline{w}}\big\{I(R(y,x),C(x))|x \in X\big\}. \end{aligned} \tag{1.4}$$

FRNN classification

Fuzzy rough sets can be applied for classification in a straightforward way. This is called fuzzy rough nearest neighbour (FRNN) classification (Jensen & Cornelis 2008):

1.6 Definition (FRNN classification). Let $X \subset A$ be a (training) dataset, and let $X = \bigsqcup C$ be a partition of X into classes. For a test instance $y \in A$, the *upper*, *lower* and *mean approximation classifiers* predict a class membership of, respectively, $\overline{C}(y)$, $\underline{C}(y)$ and $(\overline{C}(y) + \underline{C}(y))/2$ for each class $C \in \mathcal{C}$. When a single prediction is required, select the class with the highest membership.

We have, by definition, that $T(z,0) = 0$ and $T(z,1) = z$ for any $z \in [0,1]$ and any choice T of t-norm. Likewise, for any choice I of fuzzy implication, $I(z,1) = 1$ and $I(z,0) = N_I(z)$, where N_I is the fuzzy negation induced by I. Therefore, in the setting of FRNN classification — where the decision classes C are crisp — the choices of t-norm and fuzzy implication reduce to a choice of fuzzy negation, and it is common to simply use the standard negation $z \longmapsto 1 - z$ (Jensen & Cornelis 2008; Ramentol et al 2015; Vluymans et al 2019).

If we used the strict max and min operators, both the upper and lower approximation classifiers would predict the class containing the record $x \in X$ that is most similar to a test instance y, akin to traditional nearest neighbour classification (with $k = 1$).[2] Thus, using OWA aggregation with non-trivial weights fulfills a similar role to choosing higher values of k.

1.2 FRNN as a nearest neighbour classifier

In this section, we will propose three adaptations that make classification with upper and lower approximations more practical.

Replacing information systems with datasets

To begin with, note that the formulation of upper and lower approximation in the context of an information system rather limits their applicability, as it only allows us to view them as fuzzy sets in X itself. That means that we cannot calculate $\overline{C}(y)$ and $\underline{C}(y)$ unless we already know $C(y)$. In particular, this makes FRNN classification somewhat ill-defined.

Since we require R to respect attribute values, and define upper and lower approximations in terms of the values of R, we obtain a simpler definition that allows generalisation to unseen instances by defining upper and lower approximations in the attribute space. For this, we replace the notion of information systems with that of datasets (Definition 0.5), corresponding to the familiar perspective of records as elements of a vector space.

1.7 Definition (Upper and lower approximation). Let $X \subset A$ be a dataset, let R be a tolerance relation on A, and let $\overline{w}, \underline{w}$ be a choice of weight vectors of length $|X|$. The upper and lower approximation $\overline{C}$ and $\underline{C}$ of a subset C of X are the fuzzy subsets of A defined by:

[2] This was first pointed out explicitly by Verbiest et al (2012).

$$\begin{aligned}\overline{C}(y) &:= \operatorname*{owa}_{\overline{w}}\big\{\min(R(y,x), C(x)) | x \in X\big\};\\ \underline{C}(y) &:= \operatorname*{owa}_{\underline{w}}\big\{\max(1 - R(y,x), C(x)) | x \in X\big\}.\end{aligned} \tag{1.5}$$

Replacing OWA operators with weighted maxima and minima

While the use of OWA operators makes upper and lower approximations more robust against noise and outliers, they also have two significant downsides. Firstly, in order to calculate the upper or lower approximation membership of a record y, we need to order $|X|$ values, which scales poorly. And secondly, this computational effort is at best irrelevant, and at worst detrimental. For strict weights, it is clear that we do not really need to order all values, it suffices to identify the largest or smallest value. Something similar applies to exponential weights. As these reduce exponentially, the contribution of each value quickly becomes insignificant, and, eventually, impossible to compute.

For linear weights, we find that with large datasets, the weights are spread out too much (Ramentol et al 2015). We could solve this problem by choosing a different weight vector, but it is more practical to have a way to directly control how widely the weights are spread out.

The solution to both issues is the following adaptation of OWA operators:[3]

1.8 Definition (Weighted maximum and minimum). Let w be a weight vector of k values in $[0, 1]$ that sum to 1. The *weighted maximum* $w\max$ and *weighted minimum* $w\min$ induced by w of any collection X of $n \geq k$ values are defined as follows:

$$\begin{aligned}w\max\ X &:= \sum_{i \leq k} w_i \cdot X_{(n+1-i)},\\ w\min\ X &:= \sum_{i \leq k} w_i \cdot X_{(i)},\end{aligned} \tag{1.6}$$

where the so-called order statistic $X_{(i)}$ denotes the ith smallest value of X.

When $k = n$, the weighted maximum is equal to the ordered weighted average, while the weighted minimum is equal to the ordered weighted

[3]The idea to limit the application of OWA operators to the k nearest neighbours of a record was perhaps first expressed by Jensen & Mac Parthaláin (2015). It was also noted by Ramentol et al (2015) that when calculating the lower approximation of a decision class C, it makes little sense to assign weight to the subset of values corresponding to records in C, for which $I(R(y,x), C(x)) = 1$.

average induced by the dual weight vector. When $k < n$, this correspondence still holds if we pad the weight vector with zeros. So as an added benefit, this formulation no longer requires us to define two dual variants of each type of weight vector, as the choice of operator determines the orientation.

With this adaptation, we obtain the following simplified definition for upper and lower approximations that can be used for classification:

1.9 Definition (Upper and lower approximation). Let $X \subset A$ be a dataset, let R be a tolerance relation on A, and let $\overline{w}, \underline{w}$ be a choice of weight vectors. The upper and lower approximation $\overline{C}$ and $\underline{C}$ of a subset C of X are defined by:

$$\begin{aligned} \overline{C}(y) &:= \overline{w} \max_{x \in C} R(y, x), \\ \underline{C}(y) &:= \underline{w} \min_{x \in X \setminus C} (1 - R(y, x)). \end{aligned} \tag{1.7}$$

Replacing tolerance with dissimilarity

While replacing OWA operators with weighted maxima and minima prevents us from having to sort all similarity values, a naive implementation would still require calculating $R(y, x)$ for all $x \in C$ or all $x \in X \setminus C$. Therefore, we propose to redefine upper and lower approximations further in terms of a dissimilarity measure:

1.10 Definition (Upper and lower approximation). Let $X \subset \mathbb{R}^m$ be a dataset, let d be a dissimilarity measure on $\mathbb{R}^m$, and let $\overline{w}, \underline{w}$ be a choice of weight vectors of length $\overline{k} \leq |C|$ and $\underline{k} \leq |X \setminus C|$, respectively. The upper and lower approximation $\overline{C}$ and $\underline{C}$ of a subset C of X are the fuzzy subsets of $\mathbb{R}^m$ defined by:

$$\begin{aligned} \overline{C}(y) &:= \max(0, 1 - \overline{w} \min_{x \in C} d(y, x)), \\ \underline{C}(y) &:= \min(\underline{w} \min_{x \in X \setminus C} d(y, x), 1). \end{aligned} \tag{1.8}$$

Now, we can use nearest neighbour queries to obtain the $\overline{k}$ smallest distances from y to records in C and the $\underline{k}$ smallest distances from y to records in $X \setminus C$, and use these to calculate $\overline{C}(y)$ and $\underline{C}(y)$.

A common choice for R (D'eer et al 2015; Ramentol et al 2015; Vluymans et al 2019) is to define it as the mean of per-attribute similarity

relations R_i, corresponding to the attributes A_i that make up the attribute space A. When A_i is categorical, R_i evaluates to 1 for identical values and to 0 otherwise (this is the indiscernibility relation from classical rough set theory). When A_i is numerical, $R_i(y, x) := \max(0, |x-y|)/(2 \cdot r_\infty(X_i))$, where r_∞ is the half-range.

We can reformulate R in terms of the Boscovich distance[4] as follows. Given a dataset $X \subset A$ with m attributes:

1. One-hot encode every categorical attribute;
2. Rescale every numerical attribute by the half-range;
3. Let d be the Boscovich distance divided by $2m$.

The effect of weight types on classification performance

We illustrate the effect of the different weight types with an exploratory experiment (using scaled Boscovich distance, as described above). For this purpose, we use the Python library scikit-learn (Pedregosa et al 2011) to create synthetic datasets with the help of the `make_classification` function. Each dataset consists of two decision classes and twenty numerical attributes, of which eight are informative[5], four are linear combinations of these informative attributes, and the remaining eight are random noise. We repeat this procedure with different random seeds, to obtain three times one hundred training datasets with each, respectively, 1000, 10 000 and 100 000 records, as well as corresponding test datasets containing 1000 records.

The mean accuracy obtained by applying the upper approximation classifier to these classification problems is displayed in Figure 1.2, as a function of k and for three different weight types. Note first that $k = 1$ corresponds to using strict max and min operators, which is clearly suboptimal. Exponential weights improve upon this baseline, but quickly plateau. The difference between linear and reciprocally linear weights is much less substantial — they appear to achieve essentially identical performance, but with different values of k. Linear weights reach their optimum sooner, while the performance of reciprocally linear weights remains stable across a larger range of values for k. We have observed a similar pattern when we repeat this experiment with 10 decision classes, as well as for the lower approximation classifier. Our take-away is that both linear and reciprocally linear weights are good choices, and that the choice of k is more important for classification performance than the choice of either weight type.

[4] The simple 1-distance $x, y \longrightarrow \sum |y_i - x_i|$, see Section 2.1.

[5] When projected onto these eight informative attributes, the records are normally distributed in clusters around the vertices of an eight-dimensional hypercube, with each cluster assigned to one of the decision classes.

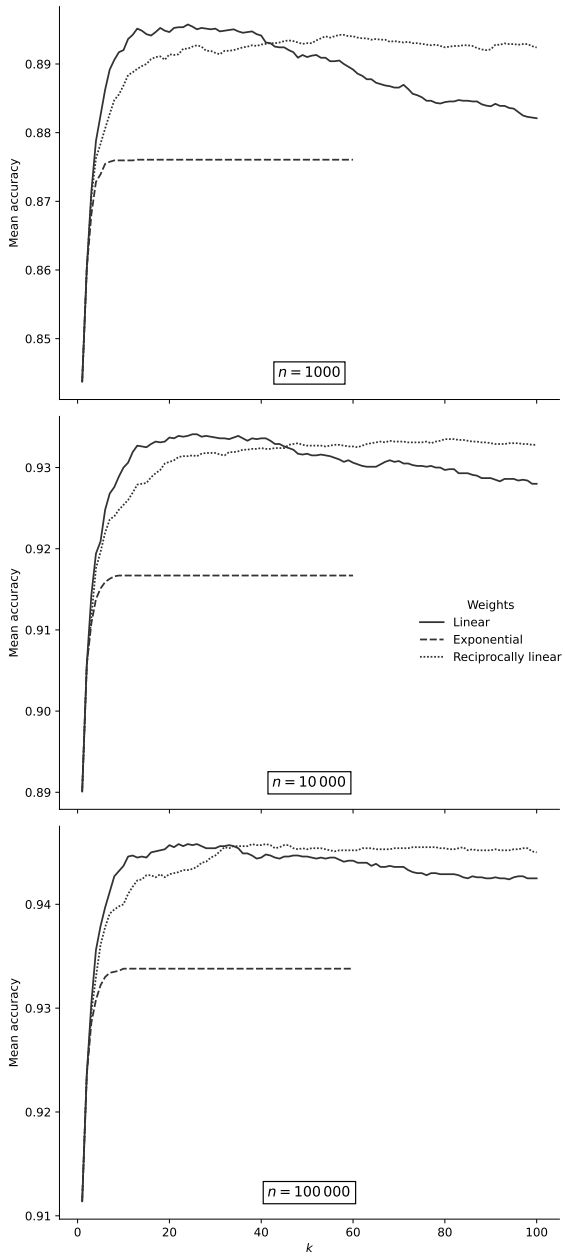


Figure 1.2: Mean accuracy for classification with the upper approximation classifier across 100 artificial training sets of size n and test sets of size 1000.

1.3 Conclusion

In this chapter, we have reviewed the application of upper and lower approximations towards classification with the FRNN algorithm. Upper and lower approximations become more robust through the incorporation of OWA operators, but the latter also have two important downsides. Firstly, they have a large impact on the computational complexity of FRNN, since they require sorting the entire training set. And secondly, they require that practitioners choose a weight vector.

We have shown how the first issue can be addressed by adapting OWA operators into weighted minima and maxima, and that both linear and reciprocally linear weights in principle perform well enough, so that the most important choice to be made by the practitioner is now the number of nearest neighbours on which they should be applied, a choice that is familiar from other nearest neighbour algorithms.

In addition, we have proposed a reformulation of upper and lower approximations that replaces the traditional tolerance relation with a dissimilarity measure. Together with the use of weighted maxima and minima, this makes upper and lower approximation membership directly calculable with nearest neighbour queries.

In the next chapter, we will review a number of measures of distance and scale. Then, in Chapter 3, we will systematically evaluate the choices of distance, scale and the number of nearest neighbours k for FRNN, and compare it with traditional nearest neighbour (NN) classification. Later, in Part II of this thesis, we will build on the more efficient definition of FRNN proposed in this chapter to scale FRNN to very large datasets.

Chapter 2

Minkowski distance

A key ingredient of nearest neighbour algorithms is the dissimilarity measure that determines the distance between records. A handful of distance measures that are commonly used, like Euclidean distance, can be viewed as special cases of a larger family, known as Minkowski distance (Minkowski 1896, also known as l_p distance). In this chapter, we will review its definition, and recall how a number of familiar concepts, like the mean, median, mode and standard deviation of a dataset, and the R^2-score of a regression model, can all be defined in terms of special cases of the Minkowski distance. We will also highlight some other special cases that may be less familiar, and suggest that they may nonetheless be useful.

2.1 Minkowski size and distance

Just as Euclidean distance can be defined in terms of the Euclidean norm, we find it convenient to define general Minkowski distance from the Minkowski size of a vector. We also need two closely related measures: rootless Minkowski size and the Minkowski mean.

2.1 Definition (Minkowski size). For a vector $x \in \mathbb{R}^m$ and some $p \in (0, \infty)$, we define the following measures.

The Minkowski p-size of x:

$$|x|_p := \left(\sum_{i \leq m} |x_i|^p \right)^{\frac{1}{p}} . \tag{2.1}$$

The rootless Minkowski p-size of x:

$$|x|^p := \sum_{i \leq m} |x_i|^p . \tag{2.2}$$

The Minkowski p-mean of x:

$$|x|_p^* := \left(\frac{1}{m} \sum_{i \le m} |x_i|^p \right)^{\frac{1}{p}} . \tag{2.3}$$

The rootless Minkowski p-mean of x:

$$\frac{1}{m} |x|^p := \frac{1}{m} \sum_{i \le m} |x_i|^p . \tag{2.4}$$

Moreover, we also define these measures for $p \in \{0, \infty\}$ through their respective limits. The three most relevant cases are:

$$\begin{aligned} |x|^0 &:= \lim_{p \to 0} |x|^p ; \\ \frac{1}{m} |x|^0 &:= \lim_{p \to 0} \frac{1}{m} |x|^p ; \\ |x|_\infty &:= \lim_{p \to \infty} |x|_p . \end{aligned} \tag{2.5}$$

2.2 Definition (Minkowski distance). The (rootless) Minkowski distance between two vectors $x, y \in \mathbb{R}^m$ is defined as the (rootless) Minkowski size of $y - x$. The (rootless) scaled Minkowski distance between x and y is defined as the (rootless) Minkowski mean of $y - x$.

Because the (rootless) Minkowski size and mean are defined in terms of absolute values, Minkowski distance is symmetric. We can retrieve the Minkowski size of x as the Minkowski distance between x and the zero vector $\mathbf{0}$.

Minkowski (1896) proved what has become known as the Minkowski inequality, namely that for $p \ge 1$ Minkowski size satisfies the triangle inequality:

$$|x + y|_p \le |x|_p + |y|_p, \tag{2.6}$$

for any $x, y \in \mathbb{R}^m$. Together with absolute homogeneity ($|\lambda| \cdot |x|_p = |\lambda \cdot x|_p$ for all $\lambda \in \mathbb{R}$) and positive definiteness ($|x|_p = 0 \implies x = \mathbf{0}$), this means that Minkowski size is a norm for $p \ge 1$.

Rootless Minkowski size does not satisfy absolute homogeneity for any $p \ne 1$, and therefore is not a norm, but for $p \le 1$ it does satisfy the triangle equality, and together with symmetry, positive definiteness, and the fact that $|\mathbf{0}|_p = 0$, this means that rootless Minkowski distance is a metric for $p \le 1$.

Four special cases (and their scaled equivalents) of Minkowski size (and corresponding distance) are of particular interest:

$p = 2$ **(rooted):** Euclidean norm.[1] Arguably the most natural distance in a geometric sense, in particular because it is invariant under rotation. Unrooted Euclidean distance is commonly referred to as *squared Euclidean distance*.

$p = 1$**:** Boscovich norm.[2] In a certain sense the simplest Minkowski size, since it is just the sum of the absolute values in each dimension. In particular, the rooted and rootless Boscovich norm are identical, and the Boscovich mean is the ordinary arithmetic mean of the absolute values in each dimension.

$p = \infty$ **(rooted):** Chebyshev norm.[3] Maximum of the absolute values. The scaled Chebyshev norm and distance are identical.

$p = 0$ **(rootless):** Hamming size.[4] Counts the number of values that are different from 0, while Hamming distance counts the number of values that are different from each other. Because Hamming distance only uses the identity or non-identity of values, it can also be applied to (and is in practice mostly used for) categorical data.

Note that rootless Chebyshev and rooted Hamming size are not particularly interesting, since they are equal to 0 or ∞ for most values. The rooted Hamming mean is the geometric mean, which we have no need for in the present chapter. Thus, while we defined four variants of Minkowski size, only in the Euclidean case all four variants (rootless and/or scaled) are relevant and distinct. Boscovich and Hamming size have scaled variants, Chebyshev size has no relevant variants.

2.2 Minkowski distance between univariate datasets

It is straightforward to extend Minkowski distance to a distance measure between univariate datasets[5] with corresponding indices, because we can represent a univariate real-valued dataset of size n as a vector in $\mathbb{R}^n$.

[1] Corresponding to the geometry described in Euclid's elements; also known as Pythagorean norm, since it corresponds to the Pythagorean theorem.

[2] Perhaps first used implicitly by Boscovich (1757, 1760) to minimise regression residuals (Eisenhart 1961; Stigler 1986; Todhunter 1873); also known as *city-block*, *Manhattan*, *rectilinear*, *right-angle* and *taxicab* norm.

[3] Popularised by Tchebyshev (1854, 1859) in the context of approximating functions; also known as *chessboard*, *king-move*, *maximum*, *square*, *supremum* and *uniform* norm.

[4] Introduced by Hamming (1950) as part of his proposal for error detection and correction.

[5] Datasets $X \subset \mathbb{R}^m$ with $m = 1$.

2.3 Definition. Let $X = (x_1, x_2, \ldots, x_n)$ and $Y = (y_1, y_2, \ldots, y_n)$ be two univariate datasets with corresponding index. Then the (rootless and/or scaled) Minkowski p-distance between X and Y is the (rootless and/or scaled) Minkowski p-size of $\langle y_1 - x_1, y_2 - x_2, \ldots, y_n - x_n \rangle \in \mathbb{R}^n$.

The squared Euclidean distance between datasets is known as the *sum of squares*, the scaled and scaled squared Euclidean distance as, respectively, the *root-mean-square* and *mean squared deviation* or *error*.

For the scaled Boscovich and Chebyshev distance, we obtain, respectively, the *mean* and *maximum absolute deviation* or *error*.

The (scaled) Hamming distance between two datasets is simply the number (share) of corresponding pairs of elements that are different.

2.3 Univariate measures of central tendency and dispersion

We can in turn extend the Minkowski distance between two univariate datasets to the distance between a univariate dataset and a single value:

2.4 Definition. Let $X = (x_1, x_2, \ldots, x_n)$ be a univariate dataset and $z \in \mathbb{R}$ a single value. The (rootless and/or scaled) Minkowski p-distance between X and z is the (rootless and/or scaled) Minkowski p-size of $\langle z - x_1, z - x_2, \ldots, z - x_n \rangle \in \mathbb{R}^n$.

With abuse of notation, we will write $|X - z|_p$ to denote the distance between a dataset and a single value. Note that unlike the distance between two datasets, the index of the dataset is not important here.

An example is the standard deviation of a dataset, which is the scaled Euclidean distance between the dataset and its mean. In turn, the mean of a dataset is the value that minimises this distance. We can generalise this as follows:

2.5 Definition. Let $X = (x_1, x_2, \ldots, x_n)$ be a univariate real-valued dataset. The Minkowski p-radius and rootless Minkowski p-radius of X are defined, respectively as:

$$\begin{aligned} r_p(X) &:= \min_{z \in \mathbb{R}} |X - z|_p^* ; \\ r^p(X) &:= \frac{1}{n} \min_{z \in \mathbb{R}} |X - z|_p . \end{aligned} \tag{2.7}$$

The Minkowski p-centre of X (not necessarily unique) is defined as:

$$c_p(X) := \begin{cases} \arg\min\limits_{z \in \mathbb{R}} |X - z|^p & p \in [0, \infty); \\ \arg\min\limits_{z \in \mathbb{R}} |X - z|_p & p \in (0, \infty]. \end{cases} \tag{2.8}$$

Note that the two cases agree for $p \in (0, \infty)$, and likewise for scaled distance.[6]

The nature of the Minkowski p-centre differs quite strongly depending on whether p is smaller or larger than 1, as demonstrated by the following two results:

2.6 Lemma. For $p \in (1, \infty)$, the Minkowski p-centre of a univariate dataset X is unique.

Proof. Let $f(z) := |X - z|^p$. For $p > 1$ and any $x \in \mathbb{R}$, $|x - z|^p$ as a function in z is twice differentiable. Consequently, f as a sum of twice differentiable functions is itself twice differentiable. Denote $X_{\leq z} := \{x \in X | x \leq z\}$ and $X_{\geq z} := \{x \in X | x \geq z\}$. Then we have:

$$\begin{aligned} f(z) &= \sum_{x \in X_{\leq z}} (z - x)^p + \sum_{x \in X_{\geq z}} (x - z)^p; \\ f'(z) &= p \cdot \left(\sum_{x \in X_{\leq z}} (z - x)^{p-1} - \sum_{x \in X_{\geq z}} (x - z)^{p-1} \right); \\ f''(z) &= p \cdot (p-1) \cdot \left(\sum_{x \in X_{\leq z}} (z - x)^{p-2} + \sum_{x \in X_{\geq z}} (x - z)^{p-2} \right) \\ &= p \cdot (p-1) \cdot |X - z|^{(p-2)}. \end{aligned} \tag{2.9}$$

$f'(z)$ is negative if $z < \min X$ and positive if $z > \max X$, so f has at least one minimum in $[\min X, \max X]$. If X consists of more than one distinct point, f'' is positive everywhere, and therefore f is a strictly convex function with no more than one minimum. □

2.7 Lemma. For $p \in (0, 1)$, the Minkowski p-centre of a univariate dataset X is in X.

Proof. If $p \in (0, 1)$, then $f(z) := |X - z|^p$ is continuous, but not differentiable in the points of X. However, on each segment of the complement of X, f is still twice differentiable, with derivatives as in (2.9). Since $p > 0$ and $p - 1 < 0$, we now have that f'' is negative everywhere, whence f is strictly concave on each segment, and that f' diverges to ∞ as $z \to \max X_{\leq z}$ and to $-\infty$ as $z \to \min X_{\geq z}$. Consequently, f has a local minimum in each point of X and no minima in the complement of X. □

The Hamming, Boscovich, Euclidean and Chebyshev centres and radii admit an explicit characterisation; they correspond to familiar

Table 2.1: Special cases $p \in \{0, 1, 2, \infty\}$ of the Minkowski centre c_p and Minkowski radius r_p.

p	c_p	r_p	r^p
0	Mode	—	Variation ratio
1	Median	Mean absolute deviation around the median ($r_1 = r^1$)	
2	Mean	Standard deviation	Variance
∞	Midrange	Half-range	—

measures of central tendency and dispersion (Table 2.1). The proofs that these centres really minimise (2.8) are relatively straightforward:

2.8 Lemma. A Hamming centre $c_0(X)$ of a univariate dataset X is a mode of X.

Proof.

$$|X - z|^0 = |\{x \in X | x \neq z\}| \tag{2.10}$$

is minimal if z is a mode of X. □

2.9 Lemma. A Boscovich centre $c_1(X)$ of a univariate dataset X is a median of X.

Proof. For any $z \in \mathbb{R}$, write $X_{\leq z} := \{x \in X | x \leq z\}$ and $X_{\geq z} := \{x \in X | x \geq z\}$. Then

$$\begin{aligned} |X - z|^1 &= \sum_{x \in X} |x - z| \\ &= \sum_{x \in X_{\leq z}} (z - x) + \sum_{x \in X_{\geq z}} (x - z), \end{aligned} \tag{2.11}$$

which is piecewise linear and continuous as a function in z. It is descending for all z such that $|X_{\leq z}| < |X_{\geq z}|$ and ascending for all z such that $|X_{\leq z}| > |X_{\geq z}|$. It is constant, and consequently minimal, for all z such that $|X_{\leq z}| = |X_{\geq z}|$, i.e. if z is a median. □

2.10 Lemma. The Euclidean centre $c_2(X)$ of a univariate dataset X is the mean of X.

Proof. We want to find the value $z \in \mathbb{R}$ for which $f(z) = |X - z|^2 = \sum_{x \in X} (x - z)^2$ is minimal. By Lemma 2.6, its minimum corresponds to the unique zero of f'. We have that:

$$\begin{aligned} f'(z) &= \sum_{x \in X} -2 \cdot (x - z) \\ &= 2nz - 2 \cdot \sum_{x \in X} x, \end{aligned} \tag{2.12}$$

which is equal to 0 if:

$$z = \frac{1}{n} \cdot \sum_{x \in X} x. \tag{2.13}$$

□

2.11 Lemma. The Chebyshev centre $c_\infty(X)$ of a univariate dataset X is the midrange of X.

Proof. We have that

$$\begin{aligned} |X - z|_\infty &= \max_{x \in X} |x - z| \\ &= \max(\max X - z, z - \min X). \end{aligned} \tag{2.14}$$

This is minimal if z is the midrange of X. □

2.4 Multivariate measures of central tendency

We can generalise the Minkowski p-centres to multivariate datasets[7] by combining the per-attribute Minkowski p-centres.

2.12 Definition. Let X be a multivariate dataset. The Minkowski p-centre $c_p(X)$ of X is defined as $(c_p(X_1), c_p(X_2), \ldots, c_p(X_m))$, where X_i is the projection of X onto its ith component, for $i \leq m$.

For $p = 2$, this is the *centroid* (or *centre of gravity*) of a dataset. The centroid satisfies the property that it minimises the squared Euclidean distance to all points in the dataset. In general, for $p \in [0, \infty)$ and any $z \in \mathbb{R}^m$, we have that:

$$\begin{aligned} \sum_{x \in X} |x - z|^p &= \sum_{x \in X} \sum_{i \leq m} |x_i - z_i|^p \\ &= \sum_{i \leq m} \sum_{x \in X} |x_i - z_i|^p \\ &= \sum_{i \leq m} |X_i - z_i|^p , \end{aligned} \tag{2.15}$$

[7] Datasets $X \subset \mathbb{R}^m$ with $m > 1$.

which is minimised by $z = c_p(X) = (c_p(X_1), c_p(X_2), \ldots, c_p(X_m))$, so the Minkowski p-centre minimises the sum of the rootless p-distances to records in X.

In addition, we have that the per-attribute midrange $z = c_\infty(X)$ minimises:[8]

$$\begin{aligned}\lim_{p\to\infty}\left(\sum_{x\in X}|x-z|^p\right)^{\frac{1}{p}} &= \lim_{p\to\infty}\left(\sum_{i\le m}\sum_{x\in X}|x_i-z_i|^p\right)^{\frac{1}{p}} \\ &= \max_{x\in X; i\le m}|x_i-z_i|\,.\end{aligned} \tag{2.16}$$

2.5 Minkowski score

One application of the distance between two univariate datasets is to evaluate a series of predicted values by comparing it against a series of reference values (e.g. the ground truth). In practice, it is often desirable to normalise the resulting error score by dividing it by some baseline expectation, and subtracting it from 1 to obtain a similarity value. The result is called a *skill score* (Heidke 1926; Muller 1944a,b,c). If we do this with Minkowski p-distance, we obtain the Minkowski p-score:

2.13 Definition. Let X and Y be two univariate real-valued datasets with corresponding index. Then the Minkowski p-score and rootless Minkowski p-score of X with respect to Y are defined as, respectively:

$$\begin{aligned}R_p &:= 1 - \frac{|Y-X|_p}{r_p(Y)}, \\ R^p &:= 1 - \frac{|Y-X|^p}{r^p(Y)},\end{aligned} \tag{2.17}$$

with r_p and r^p the Minkowski p-radius and rootless Minkowski p-radius (Definition 2.5).

The squared Euclidean score is the R^2 score that is a popular measure of regression performance. R_1 and R_∞ are alternative measures that put different amounts of weight on smaller or larger deviations. Note that R_2 is simply a transformation of R^2, namely $1 - \sqrt{1-R^2}$.

R^0 is a modified accuracy score. It expresses the number of matching values, divided not by the total number of values, but by the number of values in the target set that are different from the mode. For binary

[8] Note, however, that $z = c_\infty(X)$ is in general not a unique minimum. Let $r_\infty^{\max}$ be the maximum per-attribute half-range of X. Then on the remaining attributes with smaller half-ranges, there are more values than just the midrange that lie within $r_\infty^{\max}$ of all points, and any combination of these values will still minimise (2.16).

datasets that are perfectly balanced, this linearly scales the ordinary accuracy score from $[0, 1]$ to $[-1, 1]$. For imbalanced datasets, it corrects for the strategy of always predicting the mode, without otherwise giving more weight to minority classes. Thus, it represents a potentially useful alternative to balanced accuracy.

2.6 Conclusion

In this chapter, we have reviewed the family of Minkowski p-distances, and highlighted the special cases of Hamming, Boscovich, Euclidean and Chebyshev distance. We then showed how Minkowski distance can be extended to a distance measure between datasets and how this underpins a number of very familiar measures from everyday data science.

One open question is whether the case $p = \frac{1}{2}$ is also of interest, which one might expect on the basis of symmetry, given that it mirrors the Euclidean case. In particular, we may ask whether the centre $c_{\frac{1}{2}}$ of a dataset admits an explicit characterisation.

In the next chapter, we will evaluate the performance of NN and FRNN classification with Boscovich, Euclidean and Chebyshev distance, as well as with r_1, r_2 and r_∞ scaling.

Chapter 3

Classification performance

In Chapter 1, we have argued that if we use linear weights with FRNN classification, the choice of a weight vector can effectively be reduced to the choice of its length k. In the present chapter, we will evaluate the performance of FRNN when we optimise k on validation data, and establish a good default value for k. We will do the same for classical nearest neighbour (NN) classification with and without weights, which will allow us to compare FRNN and NN.

The nearest neighbour queries at the heart of NN and FRNN classification are sensitive both to the distance measure used and the relative scale of each attribute. A typical default for NN is to rescale by the standard deviation and to use Euclidean distance, while we saw in Chapter 1 that the traditional approach for FRNN is to scale by the range and to use scaled Boscovich distance.

In the previous chapter, we have identified three popular distance measures for numerical data: Boscovich, Euclidean and Chebyshev distance, corresponding to Minkowski p-distance with p equal to, respectively, 1, 2, and ∞. In addition, we have identified three measures of dispersion that can be used to rescale datasets: the mean absolute deviation around the median r_1, the standard deviation r_2 and the half-range r_∞.

Because the performance of NN and FRNN is inextricably linked to the distance measure and scale of the data, we will also establish the best choices for these two measures.

We start with a brief overview of the literature (Section 3.1). We then describe our experiment (Section 3.2) and present the results (Section 3.3), before concluding (Section 3.4).

3.1 Background

In this section, we briefly review a number of previous experiments with NN and FRNN classification.

NN classification

Nearest neighbour classification was first formally proposed by Fix & Hodges (1951). Research into the properties of nearest neighbour classification started in earnest following the influential paper by Cover & Hart (1967).

Perhaps the most significant innovation was the proposal by Dudani (1973, 1976) to weigh the contribution of neighbours on the basis of their distance. His main proposal was to weigh the ith neighbour by:

$$w_i = \begin{cases} \dfrac{d_k - d_i}{d_k - d_1} & k > 1; \\ 1 & k = 1, \end{cases} \tag{3.1}$$

where d_i is the distance to the ith neighbour. This establishes a linear correspondence between the distance of each neighbour and its weight. A curious aspect of this weighting scheme is that the kth weight is always 0 (if $k > 1$). In addition, Dudani (1973, 1976) also suggested an alternative weighting scheme, where weights correspond reciprocally to distance:[1]

$$w_i = \frac{1}{d_i}. \tag{3.2}$$

For the first type of weights, Dudani (1973, 1976) demonstrated lower classification error than unweighted NN on a synthetic dataset. However, Bailey & A Jain (1978) subsequently showed that this was due to the fact that Dudani (1973, 1976) had counted all ties as errors, and that when these are resolved instead (e.g. by choosing randomly), the performance of weighted and unweighted NN was similar on the synthetic dataset. Moreover, Bailey & A Jain (1978) also proved that the asymptotic classification error of unweighted NN is minimal among all possible weighted variants of NN. This in turn elicited a response by Macleod et al (1987), who argued that there exist finite classification problems where some distance-weighted variants of NN do have lower error.

Despite the extensive literature on NN classification, there have only been a small number of experimental studies of distance-weighted NN.

[1] If $d_i = 0$ for some i, we set $w_i = 1$ for all such i and $w_i = 0$ for all other i. Similarly, if $d_k - d_1 = 0$ in (3.1), we set all weights equal to 1. In either eventuality, this essentially means that we perform unweighted NN with the neighbours that have identical attribute values to the test record.

Working with 18 synthetic and real-life datasets, Wettschereck (1994) found that reciprocally weighted NN clearly outperforms unweighted NN for Euclidean distance, and that there is no clear difference between Euclidean and Boscovich distance. Zavrel (1997) additionally considered linear weights, with slightly better results than reciprocal weights in a comparison based on 13 datasets and cosine distance, while both weighted variants clearly outperformed unweighted NN.

Moreover, we are not aware of any systematic evaluations as to what a good default choice for k might be. The default used by the popular machine learning library scikit-learn (Pedregosa et al 2011) is $k = 5$, but there are theoretical arguments that k should increase with the number of records n to limit classification error (Chaudhuri & Dasgupta 2014, and references therein). One recommendation that appears in several places (Jirina & Jirina 2011; Lantz 2013; Nadkarni 2016), with no clear origin, is to set $k = \sqrt{n}$.

FRNN classification

Vluymans et al (2019) has to date been the only systematic study into how FRNN should be configured. In addition to strict (no weights), linear, reciprocally linear and exponential weights (Table 1.1), Vluymans et al (2019) also consider a *multiplicative* weighting type, defined as follows:

$$w_i = \prod_{j \leq i} m_j, \tag{3.3}$$

with

$$m_i = \begin{cases} 1, & i = 1 \vee v_i = v_{i-1}; \\ 1 - |v_i - v_1|, & v_i \neq v_{i-1}, \end{cases} \tag{3.4}$$

where v_i is the ith value to be aggregated, and the data is scaled such that all values v_i lie in $[0, 1]$. Thus, unlike the weight types in Table 1.1, these weights are dependent on the values to be aggregated. They are designed in particular such that successive values that are equal are also weighted equally.

For the lower approximation, Vluymans et al (2019) recommend choosing the weights for a dataset according to eight rules, to be resolved in order (Table 3.1). A major limitation of this scheme is that it assumes full-length weight vectors (not limited by k).

3.2 Experimental setup

To evaluate NN and FRNN classification, we will use 50 numerical real-life datasets (these are described in Section B.1). We perform 5-fold

Table 3.1: Weighting scheme recommended by Vluymans et al (2019) for lower approximation classification. n: number of records; m: number of attributes; num: numerical attributes; c: number of classes; IR: imbalance ratio; F1: Maximum Fisher's discriminant ratio (corresponding inversely to complexity) (Ho et al 2006).

#	n	m	num	c	IR	F1	Weights
1			$= 0$				Multiplicative
2					$= 1$	≥ 2	Strict
3	≤ 1000	≥ 30					Multiplicative
4				> 5	≤ 10		Exponential
5				> 5	> 10		Strict
6	≤ 4000			≤ 5	≤ 2		Reciprocally linear
7	≤ 4000			≤ 5	> 2		Linear
8	> 4000			≤ 5			Exponential

cross-validation, and calculate the mean AUROC as a measure of the discriminative ability of each classifier. To compare two approaches, we calculate the p-value from a one-sided Wilcoxon signed-rank test. When we compare classifiers against each other, we apply the Holm-Bonferroni method (Holm 1979) to correct for family-wise error.

We compare four classifiers. Three variants of nearest neighbours: unweighted (NN), with linear weights (3.1) (NN-L) and with reciprocal weights (3.2) (NN-R). In addition, we include FRNN with linear weights (except when evaluating the weighting scheme recommended by Vluymans et al (2019)).

Our evaluation consists of three parts. Firstly, we optimise k for all classifiers, and evaluate their performance for three distance measures and for scaling by four measures of dispersion, before comparing the classifiers against each other. The distance measures are Boscovich, Euclidean and Chebyshev distance (Section 2.1); the measures of dispersion are r_1, r_2, r_∞ (Section 2.3), as well as the interquartile half-range (*iqhr*), i.e. half the distance between the first and third quartile, which is a robust measure of dispersion that is less sensitive to outliers and suited for asymmetric distributions (Rousseeuw & Croux 1993).

To optimise k, we apply a form of leave-one-out validation that is efficient in the sense that we do not have to create n models, where n is the training set size, but can take a short cut. To obtain the k nearest neighbours of a training record, we perform a $k + 1$-nearest neighbour query on the training set, and eliminate the 1st nearest neighbour, which is the training record itself. For FRNN, we select for each classification problem either the upper, lower or mean approximation based on validation AUROC. For the mean approximation, we optimise two values for k (for the upper and lower approximation).

Secondly, we identify optimal default values for k by considering

Table 3.2: One-sided p-values, comparing optimised AUROC with Boscovich distance against Euclidean and Chebyshev distance.

Classifier	Scale	Euclidean	Chebyshev
NN	r_1	< 0.0001	< 0.0001
	r_2	< 0.0001	< 0.0001
	r_∞	0.0022	< 0.0001
	iqhr	0.00018	< 0.0001
NN-L	r_1	0.00027	< 0.0001
	r_2	0.00098	< 0.0001
	r_∞	0.00037	< 0.0001
	iqhr	0.00037	< 0.0001
NN-R	r_1	0.00011	< 0.0001
	r_2	0.0011	< 0.0001
	r_∞	0.0039	< 0.0001
	iqhr	0.015	< 0.0001
FRNN	r_1	0.00023	< 0.0001
	r_2	0.0011	< 0.0001
	r_∞	0.00051	< 0.0001
	iqhr	0.00016	< 0.0001

the mean test AUROC across datasets. If a value of k is too large for a dataset, we instead substitute the largest possible value for that dataset.

And lastly, we compare the classifiers with these new default values against each other and against the values suggested in Section 3.1. For the NN variants, these are $k = 5$ and $k = \sqrt{n}$, while for FRNN we will compare against the weighting scheme recommended by Vluymans et al (2019). In order not to overestimate the performance of our recommended default values, we compare them using a leave-one-dataset-out scheme, recalculating the optimal values for each dataset on the basis of all other datasets.

3.3 Results

In this section, we will present the results of our experiments for optimised and default values of k.

Optimised performance

We will first consider classification performance when k is optimised through leave-one-out validation.

Table 3.2 lists the result of comparing Boscovich distance against Euclidean and Chebyshev distance. As can be seen, Boscovich distance produces results that are clearly better for all scales and all classifiers. Therefore, we restrict the rest of our experiments to Boscovich distance.

Table 3.3 lists the results of comparing rescaling by r_1 against r_2, r_∞ and iqhr. Generally speaking, r_1 scaling outperforms r_∞ and iqhr

Table 3.3: One-sided p-values, comparing optimised AUROC with r_1 scaling against r_2, r_∞ and iqhr scaling (Boscovich distance).

Classifier	r_2	r_∞	iqhr
NN	0.31	0.025	0.079
NN-L	0.16	0.0021	0.0084
NN-R	0.15	0.0022	0.033
FRNN	0.17	0.011	< 0.0001

Table 3.4: One-sided p-values, testing that the classifier in the row has higher AUROC than the classifier in the column, with Boscovich distance, r_1 scaling and optimised k, with Holm-Bonferroni family-wise error correction applied to each row.

	NN-L	NN-R	NN
FRNN	0.00020	0.0014	< 0.0001
NN-L		0.21	0.00045
NN-R			0.00012

scaling, although less significantly so for NN (without weights). The difference between r_1 and r_2 scaling is only weakly significant for all classifiers. Nonetheless, we will adopt r_1 scaling going forward, as it produces the best results on our selection of datasets.

When we compare the classifiers against each other for Boscovich distance and r_1 scaling (Table 3.4), we find that NN classification with distance weights performs better than without, and that weights that correspond linearly with distance give somewhat better results than reciprocal-distance weights, but that this is only weakly significant. However, FRNN classification clearly outperforms NN classification with either weight type.

Optimal default values for k

Having established that all variants of NN, as well as FRNN, work best with Boscovich distance and r_1 scaling, we consider what might be good default values for k. For this, we will look at the mean AUROC obtained across datasets, as a function of k.[2]

While it makes intuitive sense that k should grow with the dataset size n, and the value of $\sqrt{n}$ has been suggested, we have not been able to maximise mean AUROC by reparametrising k in terms of $\log n$ or $\sqrt{n}$. That is, we have not been able to obtain a higher mean AUROC by

[2]We adopt the rule that when k is larger than the size of a dataset (NN) or class (FRNN) we substitute that value instead (for that dataset or class).

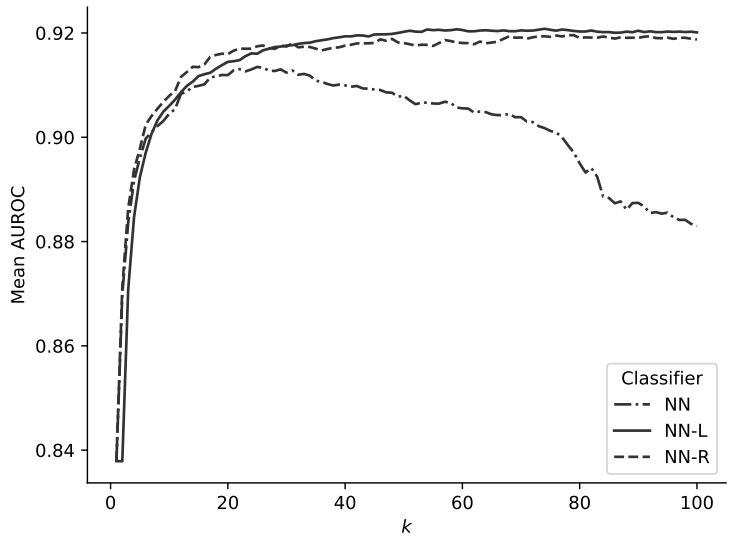


Figure 3.1: Mean test AUROC of NN without weights, and with weights that correspond linearly (NN-L) or reciprocally (NN-R) with distance.

setting $k = a \log n$ or $k = a\sqrt{n}$ and optimising a than by optimising k directly.

For NN with and without distance-weights, the result is displayed in Figure 3.1. The curves for the weighted variants of NN differ from unweighted NN in two significant ways. Firstly, they clearly achieve a higher mean AUROC. And secondly, while unweighted NN has a peak, indicating that performance deteriorates when k is too high, the weighted versions of NN are much more stable, making it much easier to select a value of k that is large enough without having to worry that it is too large. The difference between NN-L and NN-R is quite small, although NN-L does achieve a slightly higher mean AUROC.

For FRNN, we obtain the highest mean AUROC by only using the lower approximation. The resulting curve as a function of k is displayed in Figure 3.2. As with unweighted NN, this curve has a clear peak — performance deteriorates when k is too high — but it is less steep.

The resulting best values for k are listed in Table 3.5. For NN-L and NN-R, these values are somewhat arbitary — any sufficiently large k appears to provide similar performance. For FRNN, this value (21) is remarkably close to our first approximation (20) in Chapter 1.

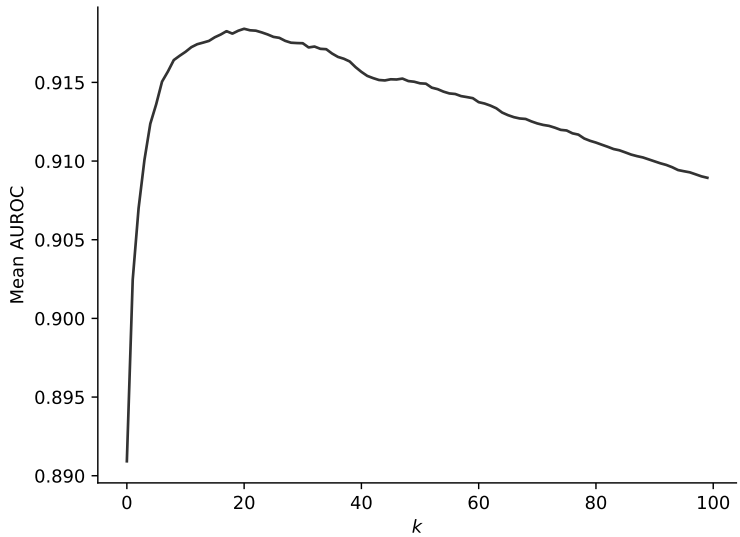


Figure 3.2: Mean test AUROC of classification with the lower approximation (FRNN).

Table 3.5: Optimal default values for k.

Classifier	k
NN	25
NN-L	74
NN-R	79
FRNN	21

Performance with default k

Lastly, we consider how well the NN variants and FRNN perform with the default values for k that we have just established, by recalculating k for each dataset using our leave-one-dataset-out scheme. When we compare the classifiers against each other (Table 3.6), we find that FRNN still performs better than the NN variants on our selection of datasets, but the difference with respect to distance-weighted NN is no longer significant. Moreover, in this setting, only NN-L is quite certain to outperform unweighted NN, while for NN-R and FRNN, the difference is only weakly significant.

For the NN variants, our new default values for k result in significantly higher AUROC than the scikit-learn default of $k = 5$ (Table 3.7).

Table 3.6: One-sided p-values, testing that the classifier in the row has higher AUROC than the classifier in the column, with Boscovich distance, r_1 scaling and default k, with Holm-Bonferroni family-wise error correction applied to each row.

	NN-L	NN-R	NN
FRNN	0.37	0.28	0.060
NN-L		0.27	< 0.0001
NN-R			0.083

Table 3.7: One-sided p-values, comparing AUROC with Boscovich distance, r_1 scaling and optimal default k against $k = 5$ and $k = \sqrt{n}$.

Classifier	$k = 5$	$k = \sqrt{n}$
NN	< 0.0001	0.15
NN-L	< 0.0001	0.052
NN-R	< 0.0001	0.19

The difference with respect to $k = \sqrt{n}$ on the other hand is only weakly significant, indicating that this was quite a reasonable guess.

For the lower approximation (FRNN), we find that simply using linear weigths with $k = 21$ leads to substantially better results than the weighting scheme recommended by Vluymans et al (2019), regardless of whether the latter is combined with r_1 scaling ($p = 0.035$), or, as originally proposed, r_∞ scaling ($p = 0.020$).

3.4 Conclusion

In this chapter, we have conducted a large-scale experiment on real-life datasets for different nearest neighbour classifiers, resulting in a number of clear findings. We found that, in general, Boscovich distance produces markedly better results than both Euclidean and Chebyshev distance. This confirms the default for FRNN, but is surprising for NN classification, where Euclidean distance is often the implicit or explicit default choice. As for scaling, we obtained the best results by scaling with r_1, the mean deviation from the median, although the difference with respect to the standard deviation r_2 is only weakly significant. When we optimise the number of nearest neighbours k, we find that FRNN outperforms all forms of NN, and that NN with distance-weights outperforms unweighted NN. Between the two different weight types for NN, there is weak evidence that Dudani (1973)'s original linear weights outperform his reciprocal weights, confirming the earlier finding by Zavrel (1997).

We have also been able to establish optimal default values for k. For FRNN, we found that it is best to only use the lower approximation, and set $k = 21$, which is remarkably close to our previous working assumption of $k = 20$. Using a leave-one-dataset-out scheme, we showed that this gives better performance than the weight vectors recommended by Vluymans et al (2019). For NN, NN-L and NN-R, the optima are located at $k = 25$, $k = 74$ and $k = 79$, respectively, but in the latter two cases, any value between 50 and 100 should give similar results. These values are significantly better than a small value like $k = 5$, the current default in scikit-learn, and there is some evidence that they may also be better than setting $k = \sqrt{n}$.

Having established FRNN as a strong nearest neighbour classifier, we will investigate in Chapter 9 whether it can be improved even further by viewing FRNN as a one-class ensemble and optimising k for each decision class.

Part II

Large datasets

Chapter 4

Distributed FRNN classification[1]

Over the course of the past few decades, ever larger quantities of data have become available as potential inputs for machine learning algorithms, to the point where the performance of machine learning algorithms is often no longer constrained by the availability of training data, but by the capability of the algorithms to handle this training data. One popular tactic to increase data processing capacity is to break down the work of an algorithm into a series of parallel tasks, and to execute these tasks on a cluster of computing cores. A number of frameworks exist that automate many of the aspects of parallel cluster computing, including Apache Spark (Karau et al 2015), which we use in this chapter.

Handling large amounts of data is a particular challenge for lazy learners like FRNN, which have to process the entire training set when they receive a test instance. Since the application of fuzzy rough sets in machine learning problems is a relatively recent, ongoing endeavour, it is not surprising that while there exist distributed implementations of nearest neigbour (Maillo et al 2017b) and fuzzy nearest neighbour (Maillo et al 2017a) classification, no big data implementation exists of a fuzzy rough set classifier. The few implementations that do try to extend the use of fuzzy rough sets to a big data context focus on preprocessing algorithms, and only Q Hu et al (2018) apply their implementation to a real dataset with more than 1 million instances.

In this chapter, we address this lacuna by presenting a big data implementation of FRNN classification. By effectively parallelising the algorithm, our implementation can be scaled to arbitrarily large datasets by adding additional computing cores. We demonstrate this through a series of systematic tests on synthetic datasets of up to 2^{24} instances. In

[1] This chapter is based on Lenz et al (2019).

Table 4.1: Existing work using fuzzy rough sets with large datasets — Use with Prototype Selection (PS) or Feature Selection (FS) and largest synthetic and real-life dataset sizes

Source	Use	Synthetic	Real-life
Asfoor et al (2014)	—	10 000 000	—
Vluymans et al (2015a)	PS	10 000 000	320 395
Asfoor (2015)	PS	10 000 000	320 395
Jensen & Mac Parthaláin (2015)	FS	—	832
Qian et al (2015)	FS	—	2310
Zeng et al (2015)	FS	—	2800
Zeng et al (2017)	FS	—	2800
Q Hu et al (2018)	FS	—	4 898 431

addition, we use our implementation to classify test instances with real datasets containing over 10 million instances.

4.1 Existing big data implementations of fuzzy rough sets

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

The first publication to explicitly adapt a fuzzy rough set algorithm for large datasets was by Asfoor et al (2014). The authors point out that for a given information system (X, A) and fuzzy set C in X, the time complexity of calculating the membership of each instance of X in the lower and upper approximations of C is $O(mn^2)$, where n is the number of records and m the number of attributes of X. In addition, the resulting indiscernibility matrix has size $O(mn^2)$, and storing it in memory becomes highly problematic as n grows. They solve these challenges with a distributed implementation in Message Passing Interface (MPI) that avoids calculating and storing the whole matrix. This work was continued by Vluymans et al (2015a), who present a distributed implementation in Apache Spark of Fuzzy Rough Prototype Selection (FRPS), a preprocessing algorithm for nearest neighbour classification developed by Verbiest et al (2013) and adapted by Vluymans et al (2015a) for nearest neighbour regression. Asfoor (2015) also adapts OWA-FRPS, a more robust version of FRPS with OWA operators, into a distributed implementation (POWA-FPRS) that approximates the ordered weighted average by partitioning the data and calculating the ordered weighted

average of the ordered weighted averages within these partitions.

Jensen & Mac Parthaláin (2015) point out that the calculation of fuzzy rough sets scales badly to large numbers of instances, and that this is further compounded if the feature space is also large. They propose three variants of Fuzzy Rough Feature Selection (FRFS). In nnFRFS and nnFDM (based on FRFS with Fuzzy Discernibility Matrices), the indiscernibility relation is modified to only consider the k nearest neighbours of each instance. Fuzzy Rough Feature Grouping (FRFG) introduces a preliminary step in which overlapping groups of correlated features are defined. For each pass, only the most decisive feature from each group is considered, and other features in the same group are then skipped, thus reducing the number of candidates that have to be evaluated.

A number of other authors have presented big data implementations of FRFS. Qian et al (2015) propose to reduce the computational cost of FRFS by relaxing the calculations of the lower and upper approximations, potentially reducing the specificity of the resulting feature selection. Zeng et al (2015, 2017) present a mechanism to incrementally update fuzzy rough approximations in a hybrid information system (HIS) (in which a hybrid metric combines different types of attributes) and apply this to feature selection. Finally, Q Hu et al (2018) present a distributed implementation of multi-kernel attribute reduction using kernelised fuzzy rough sets, and evaluate the results for Support Vector Machines (SVM) and Classification and Regression Trees (CART).

As can be seen in Table 4.1, half of these works only use datasets with up to a few thousand instances. The connected studies of Asfoor et al (2014), Asfoor (2015) and Vluymans et al (2015a) work with generated datasets of up to 10 000 000 instances and only Q Hu et al (2018) test on real datasets with more than one million instances.

The studies mentioned above have demonstrated the usefulness of scalable implementations of fuzzy rough prototype and feature selection. However, the application of FRNN classification to large datasets has so far remained unexplored.

4.2 Implementing FRNN classification on Spark

We propose a parallelised implementation of FRNN (using the mean approximation) that can classify test instances with arbitrary large datasets in a fixed amount of time if we add sufficient parallel computing power.

There exist several different frameworks for parallel computing that provide different trade-offs between ease of use, automated performance optimisation and user control. Since our main objective is to demonstrate the conceptual viability of our approach, rather than to obtain the

absolutely fastest run times possible, we have chosen to implement our algorithm in Spark, which is well-established, widely used, and which offers a relatively straightforward path to parallelisation. We implement FRNN through the Python API of Spark, using high-level dataframe operations that allow us to express operations as SQL instructions which are automatically distributed across the nodes in the cluster.

Our implementation is structured as follows:

0. Initialise Spark.
1. Read the training set, combine all attributes into a feature vector. If the attributes are numerical, scale the features to $[0, 1]$.
2. Read the test set, combine all attributes into a feature vector. If the attributes are numerical, apply the same scaling as in Step 1.
3. Optional: divide the training set from Step 1 into a large number of small partitions.
4. Fill a dataframe of length k with linear weights.
5. Broadcast the test set from Step 2 to all partitions, cross join with the training set from Step 1, calculate the distance between each pair of test and training instances and select the k closest distances per class per test instance.
6. Cache the dataframe from Step 5.
7. Join the weights from Step 4 with the distances from Step 5, multiply, and sum per class and test instance to get the upper approximations.
8. For every test instance and class, join the weights from Step 5 with the k closest training instances from Step 5 that do not belong to that class, multiply, and sum to get the lower approximations.
9. Join the upper and lower approximations from Steps 7 and 8 and for every test instance, select the class for which the sum of the approximations is highest.
10. Divide the number of test instances from Step 9 for which the predicted class matches the actual class by the total number of test instances and report the accuracy.

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

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

4.3 Experimental setup

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

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

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

The algorithm was also tested on four large real-life datasets, described in Section B.2. *susy*, *hepmass* and *higgs* are three datasets of Monte Carlo simulations of particle physics collisions. The attributes are all real-valued and we use Boscovich distance, with both attributes and distance scaled to $[0, 1]$. *poker-hand* is a slightly smaller dataset of possible hands of cards in the game of poker. It was included here because its attributes are categorical, allowing us to experiment with a different dissimilarity measure, namely Hamming distance scaled to $[0, 1]$ (see Chapter 2). We set $k = 20$.

Our primary performance measure is $T_{p,n}$, the time it takes using p cores to classify one test instance with n training instances. Time

Table 4.2: Run times in seconds per test instance of FRNN classification applied to generated training sets of different sizes, for different numbers of cores

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

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

Table 4.3: Speedups of FRNN classification applied to generated training sets of different sizes, for different numbers of cores

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

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

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

4.4 Results

Table 4.2 summarises the run times of our distributed implementation of FRNN classification for various generated training set sizes and various numbers of cores, and Table 4.3 shows the resultant speedups with

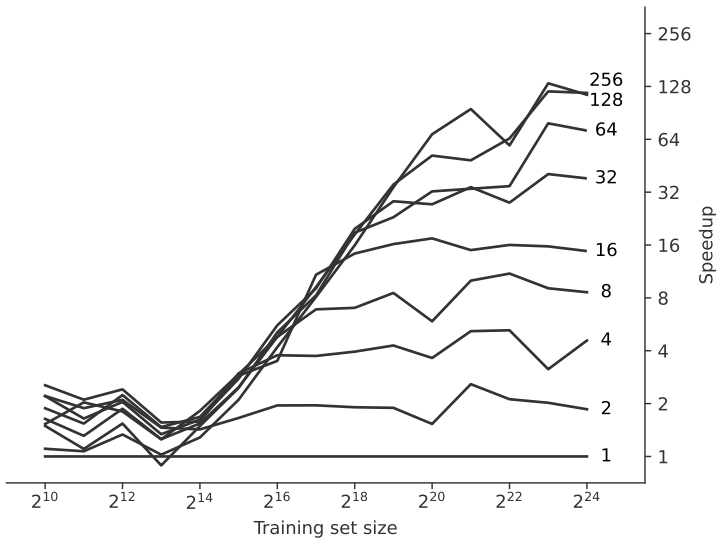


Figure 4.1: Speedups for different numbers of cores, with FRNN classification applied to generated training sets of different sizes

respect to the baseline of using only one core. The speedups are also plotted in Figure 4.1.

The results show first of all that there is a certain amount of random fluctuation, which is to be expected on shared infrastructure. For training sets with fewer than 2^{11} instances, the overhead of the implementation is the dominating factor, and run time is effectively constant. For training sets with fewer than 2^{13} instances, overhead is still large enough that it negates the effect of adding more cores: speedup is constant. As training set size grows beyond 2^{13} instances, the speedup with p cores starts to climb more or less linearly until it reaches its theoretical maximum, p. This is reflected in the distinct diagonal cluster of lines in Figure 4.1. Only the maximal configuration with 256 cores does not reach its full potential speedup within the space of these dataset sizes.

Table 4.4 shows the run times of our implementation of FRNN applied to the real datasets, which demonstrate that our implementation can be used to classify instances using FRNN with very large training sets.

Table 4.4: Run times per test instance of FRNN classification applied to real datasets, with 256 cores

Name	Time (s)
poker-hand	1.2
susy	4.3
hepmass	27
higgs	30

4.5 Conclusion

In this chapter we have demonstrated that through parallelisation on a computing cluster, FRNN classification can also be applied to very large datasets. We have showed that with sufficiently large datasets, the execution time of our implementation is effectively reduced by a factor equal to the number of computing cores. We note that a key component of our implementation was the adaptation of OWA operators into weighted minima and maxima that was proposed in Chapter 1, restricting the calculation of the weighted sum to the k nearest neighbours of each test instance, rather than the full training set.

While distributed computing is useful in principle, enabling us to use training sets of more than 10 million instances, it requires a considerable amount of computational infrastructure. Therefore, we will consider in the next chapter a different, more pragmatic approach towards FRNN classification with large datasets: reducing computational complexity by approximating the nearest neighbour search.

Chapter 5

Approximate FRNN classification[1]

In the previous chapter, we considered what could be called the brute force approach towards large datasets: simply using more computing resources. But the availability of ever larger amounts of data for machine learning also compels us to scrutinise the scalability of existing algorithms, and where necessary to develop variants or alternatives that scale better. For machine learning problems where run time is a greater impediment to performance than the amount of available data, approximative algorithms with a slightly lower accuracy but a higher capacity may offer a worthwhile trade-off. If the computational complexity of an algorithm is superlinear, computation time may blow up as dataset size grows. But even if it is strictly linear, processing orders of magnitude more data requires orders of magnitude more computing time. Ideally, then, we would like to have access to algorithms with logarithmic computational complexity.

Computing FRNN classification requires a nearest neighbour search in each decision class for those training instances that are closest to a given test instance. Consequently, a straightforward implementation without preprocessing has a time complexity that is linear with respect to the number of training instances. As discussed above, this restricts the applicability of FRNN classification to large datasets. Parallelisation, as in the previous chapter, cannot speed up FRNN by a factor larger than the number of processor cores available. Thus, while we were able to use real-life training sets with as many as 10 000 000 records to classify individual test records, the results that we obtained also indicate that performing cross-validation on the whole dataset would still require a run time of about ten years. Therefore, it would be more effective if we could reduce the time complexity of FRNN classification itself.

[1] This chapter is based on Lenz et al (2020b).

There is a rich literature of nearest neighbour search algorithms (Indyk & Motwani 1998; Jégou et al 2011; J Johnson et al 2021; Muja & Lowe 2014; Yu et al 2015). One of the most popular exact approaches uses a so-called KD-tree (Bentley 1975). It has a theoretical average query time complexity that is logarithmic, but in practice this is hard to achieve, and its query time complexity for real datasets with more than a handful of attributes is much closer to linear (Andoni & Indyk 2017). A recent approximate nearest neighbour proposal, Hierarchical Navigable Small World (HNSW) (Malkov & Yashunin 2020), promises to achieve actual logarithmic query time complexity at a very low constant error rate.

In this chapter, we propose approximate FRNN classification, which we obtain by incorporating HNSW nearest neighbour identification into FRNN. We hypothesise that this is a particularly fortuitous combination because the use of OWA operators has been shown to make FRNN robust against noise (D'eer et al 2015), and this should translate into a certain amount of tolerance for the nearest neighbour misidentifications introduced by HNSW.

To test the performance of approximate FRNN classification, we explore four different parameter settings of HNSW, and show that it is possible to achieve logarithmic query time complexity while sacrificing a minimal amount of accuracy with respect to exact FRNN classification. This means that approximate FRNN classification can be applied to very large datasets, and we demonstrate this by performing cross-validation on three of the largest datasets of the UCI Machine Learning Repository (Dua & Graff 2019).

5.1 Hierarchical Navigable Small World graphs

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

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

A classical example of an exact representation is the binary KD-tree (Bentley 1975), which iteratively divides the training set in two with a series of hyperplanes. The theoretical asymptotic average query time complexity of KD-trees is $O(\log n)$ (Friedman et al 1977), but KD-trees suffer from the "curse of dimensionality", in the sense that with datasets with more than a handful of attributes, the time complexity in practice is much closer to linear (Andoni & Indyk 2017).

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

Both query time and the correctness of the result depend on the edges between the nodes. Everything else being equal, query time is reduced if there are fewer neighbours per node and if the graph can be traversed in fewer steps, whereas more edges generally improve the reachability of the actual k nearest neighbours of y. The Navigable Small Worlds (NSW) model (Malkov et al 2014) strives to strike a good balance between these tendencies through a mix of short and long distance connections. It adds training instances to the graph in random order, and inserts edges to the M nearest instances already present, for some value of M. As a consequence, connections that are established early generally cover larger distances than connections that are established later. By starting the nearest neighbour search at an instance that was inserted early, we gain immediate access to these long-distance connections and can traverse the dataset with large steps until we reach the general neighbourhood of our query instance. The search stops when the list of nearest neighbours is no longer updated between steps. It is possible to increase the accuracy to any desired level by repeating the search from different training instances. By keeping a record of instances that have been inspected across searches and excluding them from future consideration, unnecessary repetition is avoided.

The query time complexity of NSW is $O(\log^2 n)$ for a constant accuracy of 0.999. In order to improve query time complexity further,

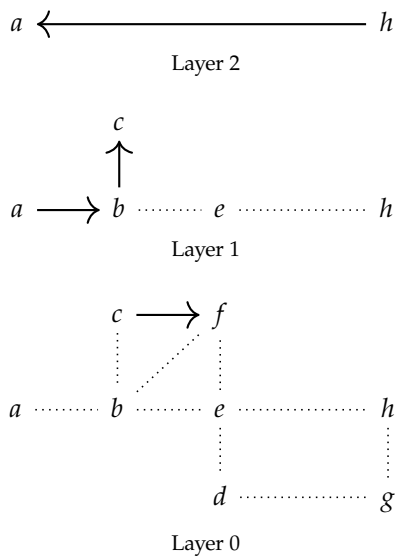


Figure 5.1: Schematic depiction of a nearest neighbour search in HNSW. The search begins in the highest layer, and continues in each lower layer from the element that was previously found. The size of the layers follows an exponential distribution.

the authors recently introduced a modified version of NSW called Hierarchical Navigable Small Worlds (HNSW) (Malkov & Yashunin 2020). The main conceit of HNSW is that it employs a hierarchy of m_L search layers that can be viewed as a vertical stack. The bottom layer consists of a search graph of the whole training set, and each subsequent layer consists of a search graph of a subset of the subset below it.

To identify the k nearest neighbours of a query instance y, we start in the top layer and run a greedy search to select one nearest neighbour, like in NSW. This process is repeated in the lower layers, in each case using the previously selected neighbour as the new entry point. In the bottom layer, the greedy search is expanded to identify ef_q candidate neighbours. From these, the k instances nearest to y are returned. This process is depicted in Figure 5.1.

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

Table 5.1: Parameters of HNSW

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

[1] The parameters ef_c and ef_q are called $efConstruction$ and ef respectively by Malkov & Yashunin (2020), they have been relabelled here for the sake of readability.
[2] These are the default values in the Non-Metric Space Library (NMSLIB, Boytsov & Naidan 2013), the implementation used in this chapter.

and connect it to up to M neighbours by repeatedly adding an edge to the closest remaining candidate neighbour that is closer to x than to any of the already selected neighbours. We then use the selected neighbours as multiple starting points in the next layer.

For a given configuration of the HNSW model, accuracy decreases with dataset size. If accuracy is kept constant, the HNSW model has a query time complexity of $O(\log n)$ and a construction time complexity of $O(n \log n)$. The principal parameters that can be used to tune performance are summarised in Table 5.1. Higher values for ef_c, M and ef_q increase accuracy in return for longer run times. However, the authors recommend to experimentally identify values for ef_c and M that produce reasonable results and then increase ef_q to attain the desired level of accuracy. In a recent empirical comparison of approximate nearest neighbour search algorithms, HNSW achieved the highest speed at all accuracy levels on all four datasets that were considered (Bernhardsson 2018). While the scope of this experiment was limited and its results are only indicative, it allows us to select HNSW as representative of the state of the art in approximate nearest neighbour search algorithms.

5.2 Approximate FRNN classification

To adapt FRNN classification for use with large datasets, we propose to calculate weighted minima on the basis of the approximate nearest neighbours of a test instance as returned by the HNSW model. Since this is the only step of the query phase of FRNN classification that is dependent on the training set size, our proposal should result in query times that scale logarithmically.

Table 5.2: Approximate FRNN classification, parameter configurations of HNSW

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

5.1 Definition (Approximate upper and lower approximation). Let $X^m_{\mathbb{R}}$ be a dataset, d a dissimilarity measure, $\overline{w}$ and $\underline{w}$ a choice of weight vectors of length $\overline{k}$ and $\underline{k}$ respectively, and N a process that takes an integer k, a multiset C in A, and an element y of A and returns a submultiset of C of size k (the *neighbour selector*). Then for any submultiset C of X, the *upper* and *lower approximation* $\overline{C}$ and $\underline{C}$ are the fuzzy subsets of A defined by:

$$\begin{aligned} \overline{C}(y) &= 1 - \overline{w} \min N(\overline{k}, C, y) \\ \underline{C}(y) &= \underline{w} \min N(\underline{k}, X \setminus C, y) \end{aligned} \tag{5.1}$$

As has been pointed out by Ramentol et al (2015), if there are just two decision classes C_1 and C_2 and $\overline{w} = \underline{w}$, then $\underline{C_1}(y) = 1 - \overline{C_2}(y)$ (up to consistency of N), and so the upper and lower approximation classifiers are identical. Therefore, we proceed with the upper approximation classifier. To obtain a clear comparison of our run times, we fix a choice of linear weights of length $k = 40$.

Definition 5.1 allows us to equip FRNN classification with alternative nearest neighbour search algorithms N. In this chapter we will compare an exact KD-tree search (*exact* FRNN) with four different parameter configurations of HNSW (Table 5.2). The goal is to identify a combination of ef_c and M that produces a good baseline in terms of accuracy. For this purpose, we fix $ef_q = k$. Combination A1 represents the default values $ef_c = 200$ and $M = 16$ (see Table 5.1). For combinations A2–4, we lower ef_c to $ef_q = 40$ and to its minimum value $M = 16$, and M to its minimum value of 5.

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

in particular due to the high noise tolerance of FRNN with weighted minima.

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

5.3 Experimental setup

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

All experiments are carried out on a single laptop computer equipped with a 4-core i7-8550U (Kaby Lake Refresh @ 1.8 GHz) processor and 16 GB of memory. We use our own Python implementation of FRNN, which incorporates the Cython (compiled to C) implementation of the scikit-learn library (Pedregosa et al 2011) for KD-Tree nearest neighbour searches and the implementation in C++ provided by the Non-Metric Space Library (NMSLIB) (Boytsov & Naidan 2013) for HNSW nearest neighbour searches. To ensure a fair comparison, all experiments are single-threaded.

As in the previous chapter, we work with three of the largest datasets from the UCI Machine Learning Repository (Dua & Graff 2019): *susy*, *higgs* and *hepmass* (described in Section B.2). These are sufficiently large that performing cross-validation with exact FRNN classification is not practical. To measure time complexity and to detect the effect of dataset size on accuracy, we take random samples of different sizes. This setup allows us to evaluate run time and accuracy on real data, while ensuring that the datasets of different sizes are in all other respects comparable to each other.

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

Table 5.3: Accuracy deficit of approximate FRNN for 5-fold cross-validation on samples of 2^{20} instances.

Method	susy	higgs	hepmass
A1	-0.0001	-0.0009	-0.0020
A2	-0.0009	-0.0014	-0.0155
A3	-0.0048	-0.0051	-0.0387
A4	-0.0072	-0.0206	-0.0654

Table 5.4: 5-fold cross-validation accuracy of approximate FRNN on full datasets.

Method	susy	higgs	hepmass
A1	0.785	0.679	0.844
A2	0.783	0.676	0.834
A3	0.776	0.654	0.813
A4	0.775	0.637	0.784

5.4 Results

Figure 5.2 shows the accuracy obtained with 5-fold cross-validation on samples of the susy, higgs and hepmass datasets. As discussed in Section 5.1, the accuracy of HNSW decreases with sample size for a given parameter configuration. This is reflected in the approximate FRNN implementations — which incorporate HNSW — in the form of an accuracy deficit with respect to exact FRNN that eventually opens up as sample size grows large enough. The final accuracy deficits at sample sizes of 2^{20} instances — the largest sample size for which cross-validation with exact FRNN proved feasible — are listed in Table 5.3. The final accuracy figures for approximate FRNN over the whole datasets are listed in Table 5.4.

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

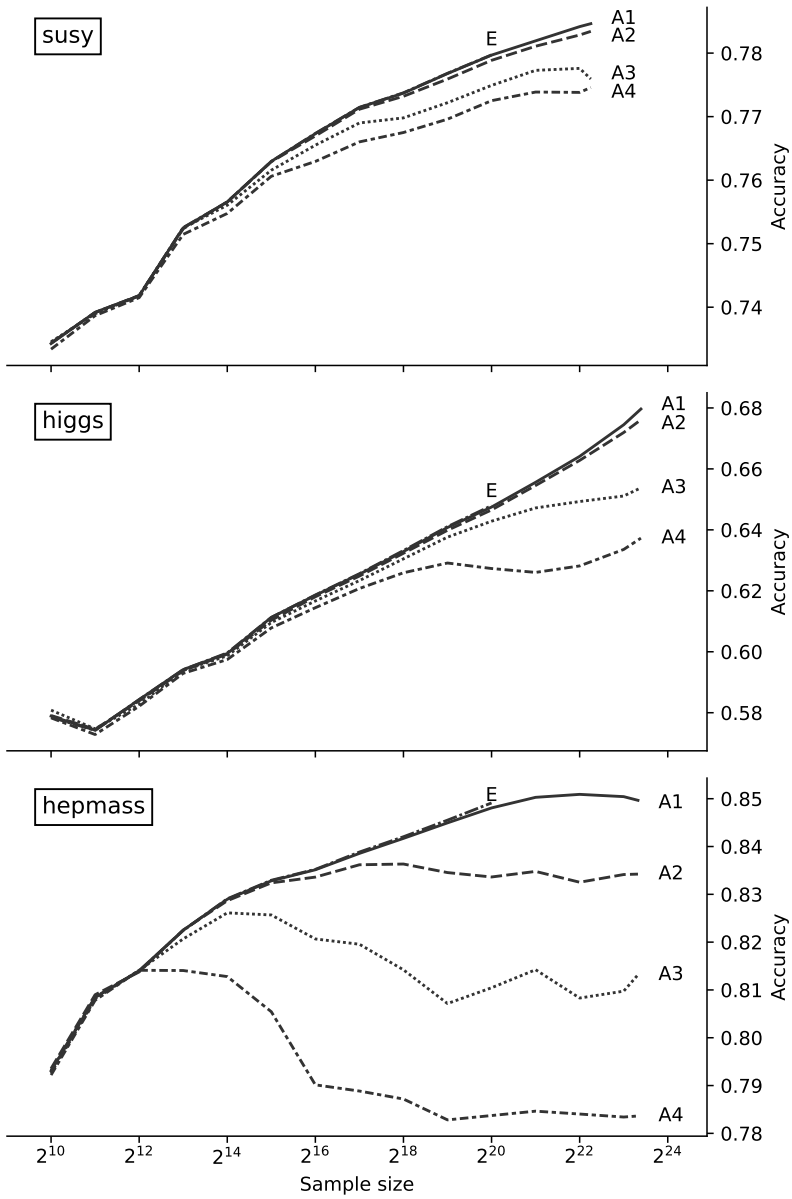


Figure 5.2: 5-fold cross-validation accuracy of exact (E) and approximate (A1–4) FRNN classification on samples of susy, higgs and hepmass.

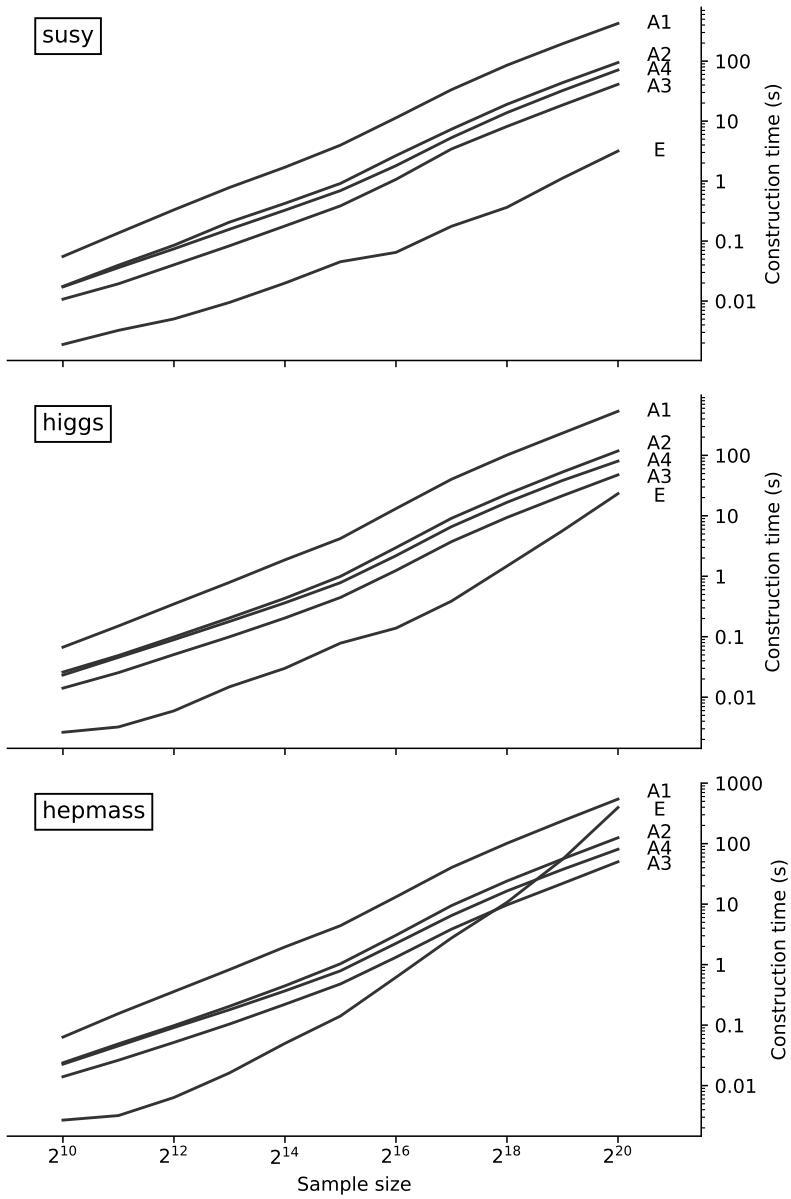


Figure 5.3: Construction times of exact (E) and approximate (A1–4) FRNN on samples of susy, higgs and hepmass.

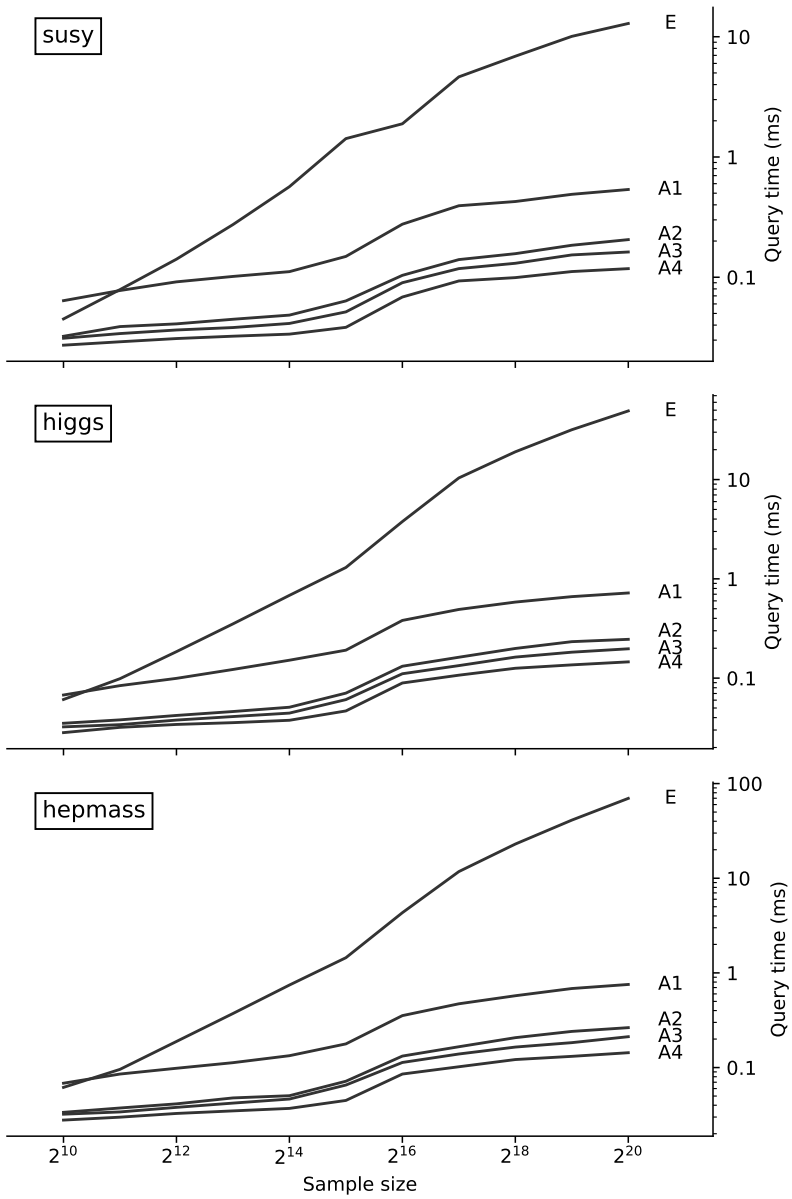


Figure 5.4: Query times per test instance of exact (E) and approximate (A1–4) FRNN on samples of susy, higgs and hepmass.

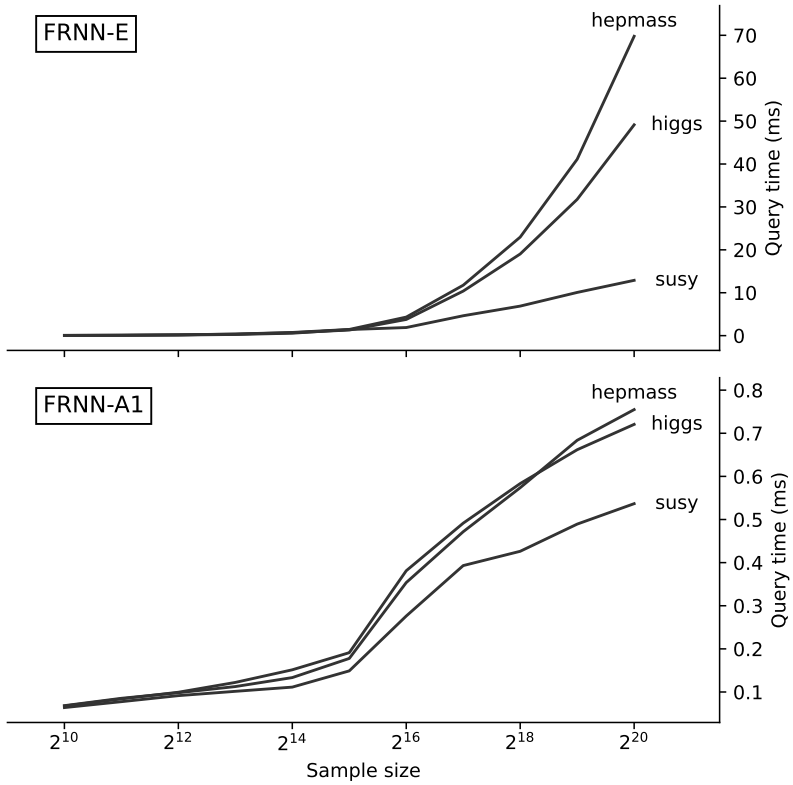


Figure 5.5: Query times per test instance of FRNN-E and -A1 on samples of susy, higgs and hepmass.

this is not the case, it should be possible to match the accuracy of exact FRNN by increasing ef_q.

Figure 5.3 shows the construction times of FRNN with various training set sizes. The construction time of approximate FRNN starts out by being longer than the construction time of exact FRNN, but with higgs and hepmass the difference becomes less pronounced as training set size grows. The observed time complexity of approximate FRNN is linear, supporting the claimed $O(n \log n)$ construction time complexity of HNSW.

The log-log graphs in Figure 5.4 show that the query time per test instance of approximate FRNN scales much better with training set size than the query time of exact FRNN. Figure 5.5 displays the same query times in lin-log graphs for exact FRNN and FRNN-A1 (the graphs for FRNN-A2–4 are very similar). It appears that with susy, exact

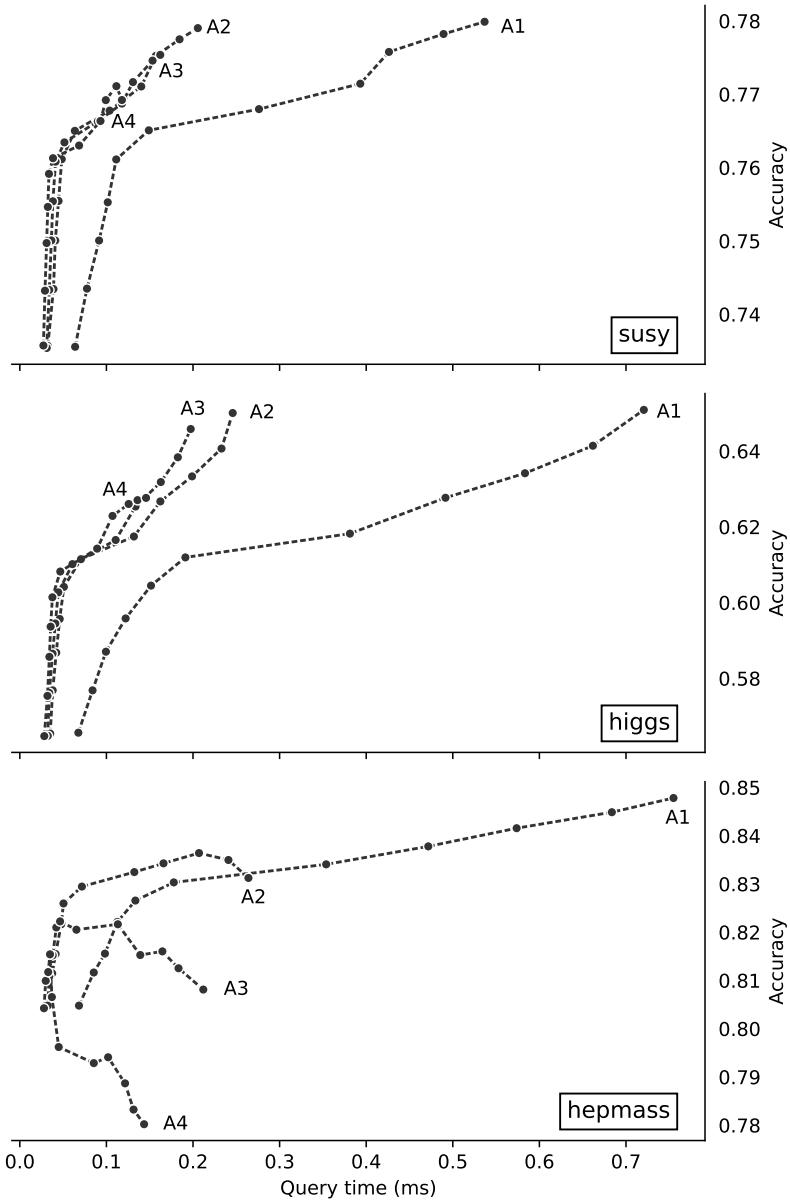


Figure 5.6: Holdout accuracy as a function of query time per test instance of approximate (A1–4) FRNN with training sets of various sizes, on samples of susy, higgs and hepmass.

FRNN possibly achieves its theoretically predicted logarithmic time complexity as training set size grows beyond 2^{16}, but that with the higher dimensional higgs and hepmass, this does not happen within the range of the tested training set sizes. The query times of FRNN-A1 scale more unevenly, but seemingly better than logarithmically as sample size grows beyond 2^{15}.

Finally, the query times of approximate FRNN are repeated in Figure 5.6 together with the corresponding holdout test accuracy. These graphs illustrate the trade-off between accuracy and query time which FRNN-A1–4 offer when the quantity of available data is not a significant limiting factor. It can be seen that for the datasets tested, most levels of accuracy can be reached with A2 in less time than A1, while A3 and A4 offer no clear advantage. However, to achieve the highest levels of accuracy with hepmass we do need A1.

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

5.5 Conclusion

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

To compare the performance of exact and approximate FRNN, we defined four parameter configurations (A1–4) of HNSW and selected three very large datasets with up to 11 million instances (susy, higgs and hepmass). These datasets were then used to draw a series of

random samples of different sizes. To evaluate classification accuracy, we performed cross-validation on these samples as well as the full datasets. Finally, we measured construction and query times through simple hold-out testing on the samples.

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

Now that the usefulness of using HNSW as part of FRNN has been demonstrated, we propose that it might be possible to achieve further improvements through deeper integration. For instance, when performing imbalanced binary classification, the computational cost of performing the nearest neighbour search required for calculating upper approximation membership of the minority class is relatively low. Therefore, we could perform this first, and terminate the nearest neighbour search for the upper approximation of the majority class early as soon as we know that a test instance has a higher membership therein. We would only have to run the full search if we end up predicting minority class membership, which should only occur in a corresponding minority of cases.

Part III

One-class datasets

Chapter 6

Average localised proximity[1]

One-class classification (Tax 2001), also known as *novelty*, *semi-supervised outlier* or *semi-supervised anomaly detection*, is an asymmetric type of binary classification between a *target* or *positive* class and the *other* or *negative* class. One-class classifiers, known as *data descriptors*, learn a model of the target class that can later be used to predict whether unseen instances belong to that target class. The difference with ordinary binary classification lies in the fact that a data descriptor only uses training data belonging to the target class. It is this restriction that makes one-class classification a challenging problem.

One-class classification is also closely related to, but subtly different from *learning from positive and unlabelled data* (Bekker & Davis 2020), in which the training data contains both positive and unlabelled records, as well as unsupervised outlier detection (Domingues et al 2018), in which the training data is an unlabelled mixture of mostly positive and some negative records and the goal is to find out which are which. Algorithms originally defined for one of these settings are frequently repurposed, but their performance should be evaluated separately in each setting.

In recent years, one-class classification has been applied to a wide range of problems, including the detection of tweets promoting hate or extremism (Agarwal & Sureka 2015), or generated by bots (Rodríguez-Ruiz et al 2020), user authentication based on keystroke dynamics (M Antal & Szabó 2015), writer identification (Hadjadji & Chibani 2018), detecting abnormal train door operations (Ribeiro et al 2016), and the identification of different tumor cell subtypes (Sokolov et al 2016).

In the present chapter, we will review a number of existing data descriptors, and introduce our own proposal, Average Localised Proximity (ALP), which can be seen as an improved version of the existing Localised Nearest Neighbour Distance (LNND) and Local Outlier Factor

[1]This chapter is based on a part of Lenz et al (2021b).

(LOF) data descriptors. We will then compare their performance in the next two chapters.

We proceed by providing formal definitions of one-class classification in general (Section 6.1) and the eight existing data descriptors covered in this thesis in particular (Section 6.2). We then introduce Average Localised Proximity (Section 6.3) and offer some concluding thoughts (Section 6.4).

6.1 One-class classification

Formally, we may view one-class classification as a generalisation from a finite set of instances to a function from the entire attribute space to the unit interval (Definition 6.1). Recall that a dataset $X \subset \mathbb{R}^m$ is a finite multisubset (Definition 0.5).

6.1 Definition. A data descriptor is a function D that takes a dataset $X \subset \mathbb{R}^m$ for some $m > 0$ and returns a function $\mathbb{R}^m \longrightarrow [0, 1]$ (the model by D of X).

In practical applications, X is a sample drawn from some statistical population, the *target class*, and the goal of one-class classification is to predict, on the basis of their attribute values, whether new instances originated from this target class. For this purpose, the values in $[0, 1]$ assigned by a data descriptor model can be interpreted as confidence scores. Accordingly, a data descriptor model should ideally assign high scores to the instances in X, although this is not a strict requirement.

Most of the data descriptors in this chapter are initially formulated as functions to a larger subset $S \subseteq \mathbb{R}$, expressing some form of similarity or distance. Such functions can easily be adapted to fit Definition 6.1 through composition with a monotonic or anti-monotonic map $S \longrightarrow [0, 1]$. In particular, we will transform distance functions $\mathbb{R}^m \longrightarrow [0, \infty)$ with the map (6.1).

$$z \longmapsto \frac{1}{1+z} \tag{6.1}$$

It may be desirable to make a more definite statement about target class membership by using a threshold α, and predicting that all instances with a score greater than or equal to α belong to the target class. The choice of α determines a trade-off between false positive and false negative predictions, and should therefore be informed by the context in which the data descriptor model is deployed.

We say that two data descriptor models are equivalent if they can be transformed into each other through a strictly order-preserving map on $[0, 1]$. Such a map is also a map between thresholds.

We can evaluate the performance of a data descriptor model using a test set $Y \subseteq \mathbb{R}^m$ containing both elements drawn from the target class (but not contained in X), as well as other instances. Evaluation in terms of accuracy is dependent on a choice of threshold. AUROC is a better evaluation metric of the model as such, since it represents the ability of the model to separate target instances from other instances. It directly corresponds to the chance that a random target instance receives a higher score than a random other instance.

As mentioned in the Introduction, one-class classification differs from unsupervised outlier detection, where the assumption is that most but not all of the instances in X are drawn from the target class, and the task is not to generalise X to $\mathbb{R}^m$, but to identify the elements in X that do not belong to the target class. Data descriptors can be used for this task by considering the scores they assign to records of X itself. Conversely, some unsupervised outlier algorithms can also be used to assign scores to instances outside of X, and we will consider some data descriptors that have been repurposed in this way.

6.2 Existing data descriptors

We now discuss a number of data descriptors that have been evaluated favourably in the comparisons by Janssens et al (2009) and Swersky et al (2016), or that have been claimed to perform even better. Throughout, we assume a dataset $X \subset \mathbb{R}^m$ and a generic instance $y \in \mathbb{R}^m$, and define a descriptor through the action of its model of X on y. For the descriptors based on nearest neighbour distances, we use $\mathrm{NN}_k(z)$ to denote the kth nearest neighbour in X of any $z \in \mathbb{R}^m$, excluding z itself if z is explicitly drawn from X. We will define the respective hyperparameter spaces informally, as a choice of hyperparameters.

Fig. 6.1 illustrates the models obtained from applying the data descriptors in this section and the next to the same toy dataset.

Nearest Neighbour Distance

Nearest Neighbour Distance (NND) is conceptually very simple, and goes back to at least Knorr & RT Ng (1997). In its general form, it requires a choice of a dissimilarity measure d and a positive integer k. It is based on the score $d_k(y) := d(y, \mathrm{NN}_k(y))$, from which we obtain a model through composition with (6.1). While this is not scale-invariant, different scalings lead to equivalent models of X.

Localised Nearest Neighbour Distance

The argument for Localised Nearest Neighbour Distance (LNND) (Ridder et al 1998; Tax & Duin 1998) is that distance in the attribute space

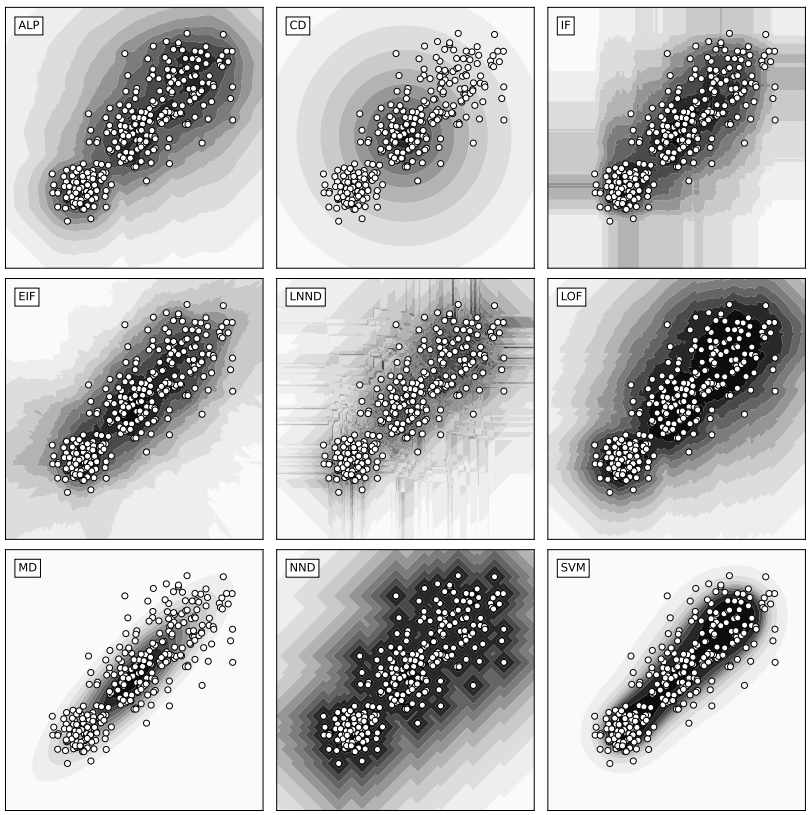


Figure 6.1: Contour lines of data descriptor models, constructed on a randomly generated toy dataset. ALP, LNND, LOF, NND and SVM have been initialised with the default hyperparameter values from Table 7.2. CD: centroid distance. Note that for each subplot, the range of contours corresponds to the range of scores for that data descriptor (in the plotted area), so the different shades of grey do not encode the same scores across data descriptors.

should not be valued equally everywhere, but that it should instead be compared to the local distance between nearby training instances. Thus, for a choice of dissimilarity d and a positive integer k, we can define the local distance $d_k^2(y) = d_k(\mathrm{NN}_k(y))$ relative to a point $y \in \mathbb{R}^m$ and the localised distance $ld_k(y) = d_k(y)/d_k^2(y)$, which we compose with (6.1) to obtain a model of X.

Local Outlier Factor

Local Outlier Factor (LOF), originally proposed for unsupervised outlier detection (Breunig et al 2000), is also based on nearest neighbour distances, and also involves a form of localisation. For a choice of dissimilarity d and positive integer k, it derives from d the (non-symmetric) *reachability* distance rd_k (6.2). The goal is to "significantly reduce" the "statistical fluctuations" that occur when query instances lie very close to target instances. It achieves this by effectively cancelling out all distances that are smaller than a certain threshold, determined by local nearest neighbour distances.

$$rd_k(y, x) = \max(d(y, x), d_k(x)), \tag{6.2}$$

LOF then aggregates and inverts a range of reachability distance values to obtain the local reachability density lrd_k (6.3), and localises and aggregates a range of local reachability density values to end up with the local outlier factor lof_k (6.4), which we compose with (6.1) to obtain a model of X.

$$\mathrm{lrd}_k(y) = \frac{1}{\frac{1}{k}\sum_{i \le k} rd_k(y, \mathrm{NN}_i(y))}, \tag{6.3}$$

$$\mathrm{lof}_k(y) = \frac{1}{k}\sum_{j \le k} \frac{\mathrm{lrd}_k(\mathrm{NN}_j(y))}{\mathrm{lrd}_k(y)}. \tag{6.4}$$

Mahalanobis Distance

One of the oldest approaches to novelty detection is to assume that the target class samples are drawn from a multivariate Gaussian Distribution. The atypicality of an instance under this distribution is given by the Mahalanobis Distance (MD) (Mahalanobis 1936) to this distribution (6.5), where μ and S are the mean and the covariance matrix of the training instances.

$$D(y) = \sqrt{(y - \mu)^T S^{-1} (y - \mu)}, \tag{6.5}$$

Mahalanobis distance generalises distance from the mean in terms of standard deviations in a univariate Gaussian distribution. Squared Mahalanobis distance follows a χ^2-distribution with m degrees of freedom, and we obtain a p-value (6.6) by applying its cumulative distribution function F and subtracting from 1.

$$p(y) = 1 - F(D(y)^2). \tag{6.6}$$

This p-value is a natural choice for a model of X, but it approaches 0 very quickly, making it computationally difficult to distinguish between p-values of large Mahalanobis distances (they are all rounded to 0). Therefore, in order not to unduly limit the discriminative power of MD, we instead compose D with (6.1) in the experiment that we perform in the next chapter.

Support Vector Machine

There exist two, practically equivalent, adaptations of the soft-margin Support Vector Machine (SVM) (Cortes & Vapnik 1995) to one-class classification. Both allow the use of a kernel $k : \mathbb{R}^m \times \mathbb{R}^m \longrightarrow \mathbb{R}$ to transform the feature space in which we obtain the solution via an implicit map $\phi : \mathbb{R}^m \longrightarrow Z$ to some inner product space Z.

The Tax variant (Tax & Duin 1999a,b) fits a hypersphere of minimal volume around the training instances by solving the optimisation problem (6.7), with dual (6.8), for a choice of $C \in [0, \infty)$ and a choice of kernel k. The instances x_i with corresponding non-zero values of α_i are the support vectors — these span a hypersphere centred at some point $a \in Z$ with radius R^2. The parameter C determines how many training instances remain outside the hypersphere. The decision function is d_T (6.9), the distance to a.

$$\min_{a \in Z, R \in \mathbb{R}, \xi \in \mathbb{R}^n_{\geq 0}} R^2 + C \sum_i \xi_i, \qquad \text{with } \forall i : \left\| \phi(x_i) - a \right\|^2 \leq R^2 + \xi_i \tag{6.7}$$

$$\min_{\alpha \in [0,C]^n} \sum_i \alpha_i k(x_i, x_i) - \sum_{i,j} \alpha_i \alpha_j k(x_i, x_j), \qquad \text{with } \sum_i \alpha_i = 1 \tag{6.8}$$

$$d_T(y) = k(y, y) - 2 \sum_i \alpha_i k(y, x_i) + \sum_{i,j} \alpha_i \alpha_j k(x_i, x_j) \tag{6.9}$$

The Schölkopf variant (Schölkopf et al 1999, 2001) fits a hyperplane to separate the training instances from the origin, at a maximum distance from the origin, by solving the optimisation problem (6.10), with dual (6.11), for a choice of $\nu \in (0, 1]$ and a choice of kernel k. The instances x_i with corresponding non-zero values of α_i are the support vectors —

these span a hyperplane with distance ρ to the origin. The parameter ν determines how many training instances remain on the wrong side of the hyperplane. The decision function is the signed distance d_S to this hyperplane (6.12), with negative values on the side of the origin.

$$\min_{w \in Z, \rho \in \mathbb{R}, \xi \in \mathbb{R}^n_{\geq 0}} \frac{1}{2} \|w\|^2 + \frac{1}{\nu n} \sum_i \xi_i - \rho, \qquad \text{with } \forall i : w \cdot \phi(x_i) \geq \rho - \xi_i \tag{6.10}$$

$$\min_{\alpha \in [0, \frac{1}{\nu n}]^n} \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j k(x_i, x_j), \qquad \text{with } \sum_i \alpha_i = 1 \tag{6.11}$$

$$d_S(y) = \sum_i \alpha_i k(x_i, y) - \rho. \tag{6.12}$$

Both SVM variants have been shown to produce the best overall results with the Gaussian kernel (6.13), which requires a choice for the *kernel width* c.

$$k(x, y) = e^{-\frac{\|x-y\|^2}{c}} \tag{6.13}$$

For kernels k for which $k(y, y)$ is constant, like the Gaussian kernel, the respective optimisation problems become equivalent (with reparametrisation $C = \frac{1}{\nu n}$), resulting in the same set of support vectors (Schölkopf et al 2001; Tax & Duin 2004), and the decision functions d_T and d_S are monotonic linear transformations of each other, leading to equivalent data descriptor models. In this thesis we use a data descriptor model based on the Schölkopf variant (6.14), for which we have access to an implementation. Instances on the hyperplane receive a score of 0.5.

$$y \longmapsto \frac{1}{2} \left(\frac{d_S(y)}{\left| d_S(y) \right| + 1} + 1 \right). \tag{6.14}$$

With the Gaussian kernel, the Schölkopf variant of SVM has two hyperparameters that need to be tuned, ν and c.

Isolation Forest

Isolation Forest (IF) (FT Liu et al 2008) is an adaptation of the Random Forest classifier for one-class classification. Its central idea is that instances that are more isolated from the target class should be easier

to separate from the training instances. This idea is modelled by constructing randomised search trees on the target data, and measuring the average number of steps required to pass through these trees.

For positive integers t and ψ, IF creates t binary incomplete search trees of height at most $\lceil \log_2(\psi) \rceil$ using t random subsamples of ψ instances of X, in which each node splits the remaining instances by randomly selecting an attribute and a corresponding value within the range of remaining values, until the maximum tree height is reached.

The expected average path length in a tree of i instances is expressed by c (6.15), where H_{i-1} is the $(i-1)$th harmonic number.

$$c(i) = 2H_{i-1} - \frac{2(i-1)}{i} \tag{6.15}$$

For each tree T, define $h_T(y)$ as the sum of the path length of y in T and $c(j)$, where j is the subsample size remaining in the final node of y in T. The rationale behind limiting tree height and estimating the remaining path length is that this limits the number of search steps, while it mostly affects target class instances.

From this we obtain an anomaly score s in $[0, 1]$ (6.16), which we subtract from 1 to obtain a data descriptor model.

$$s(y) = 2^{-\frac{1}{t}\sum_T h_T(y)/c(\psi)}. \tag{6.16}$$

The performance of IF initially increases with the number of trees t and the subsample size ψ, but eventually converges, and FT Liu et al (2008) experimentally finds that these hyperparameters can safely be set to 100 and $\min(256, n)$ respectively. Hence, a significant advantage of IF is that it has no tuning parameters.

Extended Isolation Forest

Extended Isolation Forest (EIF) (Hariri et al 2021) is a slight modification of IF, which only changes the construction of the trees. It is motivated by the observation that IF seems to produce counter-intuitive results with simple distributions. In particular, IF assigns much lower anomaly scores to instances that are far removed from the training set along a single feature dimension (but not the others) than to instances that are slightly removed from the training set in several feature dimensions. This is due to the fact that the splits in the binary trees correspond to hyperplanes in the attribute space that always lie parallel to $m-1$ feature dimensions.

EIF solves this bias by splitting its trees along hyperplanes with a randomly chosen slope and an intersect randomly drawn from the range determined by the remaining instances.

Shrink Autoencoder

The problem of one-class classification can also be approached with autoencoders. These neural networks learn a latent representation of a dataset by forcing the data through a bottleneck layer with fewer features than the input. After this encoding step, the data is decoded again by applying the same weights in reverse, and the whole model is trained by minimising the resulting reconstruction error. After training is completed, the reconstruction error of new instances can be used to define a data descriptor. However, Cao et al (2019) argue that better results can be obtained by working directly with the latent representation of an autoencoder, if we force this representation to conform to a compact distribution. For this purpose, Cao et al (2019) propose the Shrink Autoencoder (SAE) and the Dirac Delta Variational Autoencoder (DVAE). We only consider SAE here, as its reported results are slightly better and it has proven easier to implement.

SAE is an autoencoder with five hidden layers, in which the number of features is linearly reduced from m to $\lfloor\sqrt{m}\rceil+1$ — the number of latent features of the central layer[2] — and increased back to m. In addition to the reconstruction error, the loss function also incorporates Euclidean regularisation on the central layer, thereby directing the learning process towards latent representations that are distributed closely around the origin.

By substituting the latent for the original representation of the data, SAE can be used as a preprocessing step in combination with any other data descriptor. Cao et al (2019) show that this is beneficial, especially for sparse datasets. Moreover, due to the regular shape of the latent representation, the choice of hyperparameters and even the choice of data descriptor becomes largely irrelevant. Therefore, Cao et al (2019) propose that one may use SAE in combination with a simple centroid data descriptor, which measures the distance to the mean after rescaling by the respective standard deviations in each dimension. It is this combination that we will test in the next chapter.

6.3 Average Localised Proximity

The performance of NND and LNND has been compared by Swersky et al (2016), with the remarkable result that NND outperforms LNND. Yet the principle behind LNND seems sensible: if a dataset is more densely distributed in one part of the attribute space than in another, we should adjust our expectations regarding nearest neighbour distance accordingly. So we may ask what causes LNND to perform badly.

[2] By $\lfloor\sqrt{m}\rceil$, we mean $\sqrt{m}$ rounded to the nearest integer.

The most obvious problem with LNND is that its measure for local nearest neighbour distance is not very robust, since it is determined by a single distance in the training set. In every natural dataset, there is random variation in the distances between instances, and this directly translates into random variation of the LNND scores of test instances. Secondly, while it seems elegant to localise against the kth nearest neighbour distance of the kth nearest neighbour of a test instance, it is not clear why the kth nearest neighbour distance of the closest training instance isn't more relevant. And finally, LNND only considers neighbour distances for a single value of k, whereas it would be more robust to aggregate over different values for k.

These problems are addressed by LOF, but LOF has issues of its own. The conceptual motivation for its slightly convoluted amalgam of localisation and reachability is not entirely clear. LOF seems to take localisation one step too far, with three rounds of averaging that require calculating the distance to the kth neighbour of the ith neighbour of the jth neighbour of a test instance y. There is also a degree of arbitrariness to its application of reachability, as it uses kth neighbour distances as a threshold for ith neighbour distances, with k generally larger than i. More fundamentally, it is not clear that reachability is a good idea, since it may discard useful information by cancelling out small distances.

As an alternative, we propose Average Localised Proximity (ALP), a new data descriptor that is more robust than LNND, yet conceptually simpler than LOF (Definition 6.3). It uses an Ordered Weighted Averaging (OWA) operator (Yager 1988) (Definition 6.2) to obtain a soft maximum.

6.2 Definition. Let w be a weight vector of length k, with monotonically decreasing values in $[0, 1]$ that sum to 1. The *Ordered Weighted Averaging* operator owa_w induced by w transforms a collection $Y = \{y_i\}_{i \leq k}$ of values in $\mathbb{R}$ into the weighted sum $\text{owa}_w y_{i \leq k} = \sum_{i \leq k} w_i \cdot y_{o(i)}$, where $o(i)$ is the index value of the ith largest element in Y.

6.3 Definition. Let $(\mathbb{R}^m, X)$ be a dataset, let d be a choice of dissimilarity function on $\mathbb{R}^m$, let $k, l \in \mathbb{N}$ be choices of positive integers, and let w^k, w^l be choices of weight vectors of length k and l respectively. For each $i \leq k$, define $D_i(y)$ (6.17), the local ith neighbour distance relative to y, and from this, $\text{lp}_i(y)$ (6.18), the localised proximity of y. Then the *average localised proximity* $\text{alp}(y)$ of y is the ordered weighted average of these values (6.19).

$$D_i(y) = \sum_{j \leq l} w^l_j \cdot d_i(\text{NN}_j(y)). \tag{6.17}$$

$$\mathrm{lp}_i(y) = \frac{D_i(y)}{D_i(y) + d_i(y)}. \tag{6.18}$$

$$\mathrm{alp}(y) = \mathrm{owa}\, w^k{}_{i \le k}\, \mathrm{lp}_i(y). \tag{6.19}$$

As illustrated in Figure 6.2, ALP features aggregation on two levels. Local nearest neighbour distance is determined with a weighted sum over a section of the training set. By choosing monotonically decreasing weights, we let the contribution of training instances to this weighted sum decrease with distance to y. The choice of weight vector determines the amount of localisation, which can be seen as a trade-off between variance and bias. For $k = l = 1$, we recover LNND with $k = 1$, whereas if $l = n$ and all weights are equal to $\frac{1}{n}$, all distances are localised against the training set mean and we obtain models that are equivalent to those produced by NND.

We thus obtain k localised distance values, which we can interpret as representing different scales. Inspired by the definition of fuzzy rough sets (Chapter 1), we first transform these into proximity values in $[0, 1]$, and then apply a weighted maximum. With suitable weights, the weighted maximum offers a combination of flexibility and robustness, emphasising the scales at which test instances have the greatest proximity to the target class, without being completely determined by any single scale. The impact of scales with small proximities is also reduced by our choice to aggregate after transforming values from $[0, \infty]$ to $[0, 1]$, rather than vice-versa.

The choice of weight vector offers further opportunity for optimisation. However, the experiments in Chapter 1 suggest that the effect of this choice may in fact be limited, and that linearly decreasing weights constitute a good default. Similarly, a good enough default choice for the dissimilarity measure is Boscovich distance. Therefore, the only other hyperparameters to be specified by the user are k and l.

We illustrate the application of ALP with a toy example (Figure 6.3), with two instances to be scored (y_1 and y_2) and a number of training instances ($x_1, x_2, \ldots$). We choose $k = 3$ and $l = 2$. Accordingly, we obtain $w^k = \left\langle \frac{3}{6}, \frac{2}{6}, \frac{1}{6} \right\rangle$ and $w^l = \left\langle \frac{2}{3}, \frac{1}{3} \right\rangle$.

A relevant selection of nearest neighbours and nearest neighbour distances is listed in Table 6.1a, while Table 6.1b contains the resulting local distances relative to y_1 and y_2, their localised proximities and their average localised proximity scores. The local distances are calculated as the weighted sum of the nearest neighbour distances of x_1 and x_2 for y_1, and x_9 and x_{10} for y_2, weighted with $\frac{2}{3}$ and $\frac{1}{3}$ respectively. The final scores are obtained by sorting the localised proximities of y_1 and y_2 and from large to small, and taking their sum with weights $\frac{3}{6}$, $\frac{2}{6}$ and $\frac{1}{6}$.

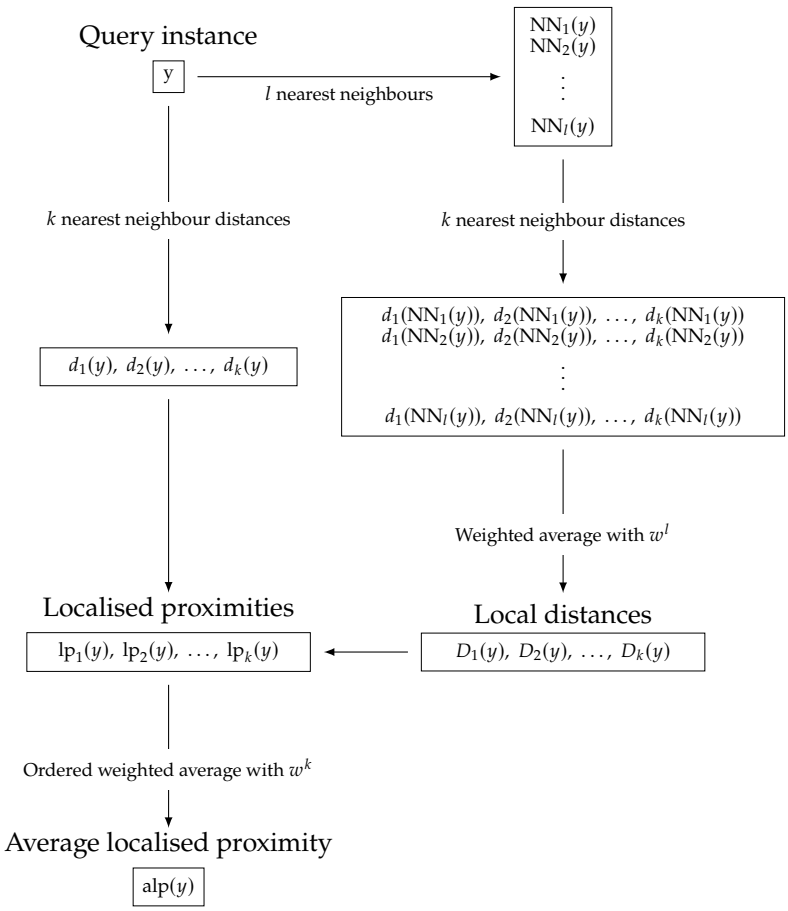


Figure 6.2: Schematic illustration of the calculation of the average localised proximity of a query instance y.

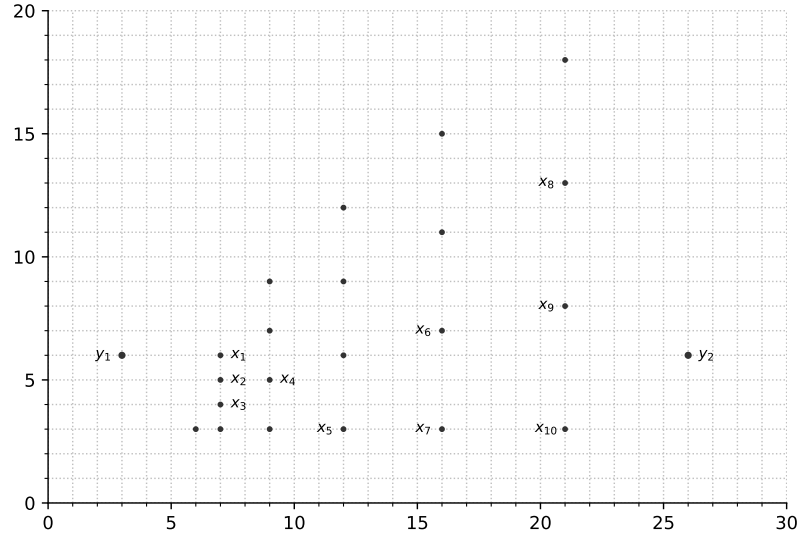


Figure 6.3: One-class classification example, with training instances ($x_1, x_2, \ldots$) and query instances (y_1 and y_2).

Table 6.1: Selected values used in the application of ALP to the one-class classification example in Figure 6.3.

(a)

	NN_1	NN_2	NN_3	d_1	d_2	d_3
y_1	x_1	x_2	x_3	4	5	6
y_2	x_9	x_{10}	x_6	7	8	11
x_1	x_2	x_3	x_4	1	2	3
x_2	x_1	x_3	x_4	1	1	2
x_9	x_8	x_{10}	x_6	5	5	6
x_{10}	x_7	x_9	x_5	5	5	9

(b)

	D_1	D_2	D_3	lp_1	lp_2	lp_3	alp
y_1	1.00	1.67	2.67	0.20	0.25	0.31	**0.27**
y_2	5.00	5.00	7.00	0.42	0.38	0.39	**0.40**

6.4 Conclusion

In this chapter, we have reviewed the challenge of one-class classification, and presented a number of data descriptors that are representative of the state of the art. We then presented a new proposal, Average Localised Proximity (ALP), motivated by a number of perceived shortcomings in the existing data descriptors LNND and LOF.

While data descriptors are intended to be applied to training data that only consists of positive data, a number of the data descriptors in this chapters, including ALP, have one or two hyperparameters without an obvious default value. In the next chapter, we will use a large selection of real-life datasets to establish good default values for these hyperparameters. Then, in Chapter 8, we will investigate how they can best be optimised experimentally if a sample of negative records is available for this purpose. In both settings, we will also compare the empirical performance of the data descriptors presented in this chapter.

Chapter 7

One-class classification with default hyperparameter values[1]

There have been two large evaluations of data descriptors for one-class classification in the literature, which identified, respectively, SVM, LOF and LNND (Janssens et al 2009) and SVM, LOF, NND and MD (Swersky et al 2016) as the best-performing algorithms. A common property of these data descriptors, with the exception of MD, is that they require setting one or two 'magic' hyperparameters by the user, which typically control a trade-off between variance and bias. Both Janssens et al (2009) and Swersky et al (2016) optimise these hyperparameters for each one-class classification task through cross-validation on the training set. While this approach can be expected to yield the best possible result for that particular task, it does not fully answer the challenge raised by the authors of the Schölkopf variant of SVM to "turn the algorithm into an easy-to-use black-box method for practitioners" (Schölkopf et al 1999). For that, we need a set of sensible default values for these hyperparameters, in particular for applications where no substantial amount of training material from the other class is available for tuning.

In contrast, IF, EIF and SAE, which were not included in these evaluations, do not require users to specify any hyperparameters. Strictly speaking, these data descriptors do have hyperparameters, but the authors have established default values that can be taken for granted. In a small-scale evaluation of unsupervised outlier detection by FT Liu et al (2008), IF was shown to achieve better results than a number of other approaches, including LOF with a (likely sub-optimal) fixed hyperparameter choice of $k = 10$. The proponents of EIF in turn showed in another small-scale experiment that it achieved better unsupervised outlier detection results than IF (Hariri et al 2021). However, it remains unclear how well IF and EIF perform one-class classification. Finally, Cao

[1] This chapter is based on a part of Lenz et al (2021b).

et al (2019) report that SAE (paired with centroid distance) outperforms LOF and SVM for one-class classification, especially with sparse datasets, but in their experiment too the choice of hyperparameters for LOF and SVM may not have been optimal, and the selection of datasets was limited.

In this chapter, we address the state of affairs sketched above in two steps. Firstly, we determine optimal default hyperparameter values for NND, LNND, LOF, SVM and ALP, which allows them to be used as "black-box methods" like MD, IF, EIF and SAE. And secondly, we compare the performance of these nine data descriptors across 246 one-class classification tasks derived from 50 datasets.

We will first describe the experiments that we perform (Section 7.1), and then present our analysis of the results (Section 7.2) and our general conclusion (Section 7.3).

7.1 Experimental setup

A large part of the one-class classification problems used by the two previous comparison studies (Janssens et al 2009; Swersky et al 2016) were created by David Tax[2] on the basis of binary or multiclass classification datasets in the UCI machine learning repository (Dua & Graff 2019), by selecting one decision class as the target class, and combining the rest to form the other class. We apply the same procedure to a selection of 50 numerical datasets that represent real-life data (described in Section B.1). By matching the variation contained in the UCI machine learning repository, we aim to achieve a certain degree of representativeness. In total, the 50 datasets contain 246 individual decision classes, with up to 194 198 target class instances and up to 649 attributes (Table B.1)[3]. We will also consider the sparsity of the target datasets, which we define as the rate of attribute values equal to the respective mode.[4]

For each descriptor, dataset and class, we perform one-class classification with 5-fold stratified cross validation. We evaluate the performance of one-class classification with the mean AUROC across folds.

Some of the data descriptors in our selection are sensitive to the relative scale of the attributes. To ensure that the potential contribution of each attribute is approximately equal, we rescale the instances within each fold, using only information from the target class instances in the training set. Since we are dealing with a large variety of target class

[2] `http://homepage.tudelft.nl/n9d04/occ/index.html`

[3] A few of the datasets have a handful of records with missing attribute values. These records were omitted.

[4] Our definition differs slightly from that of Cao et al (2019), who take the rate of zeros. The motivation for our choice is that it makes sparsity independent of the encoding of the data.

Table 7.1: Data descriptors with hyperparameters and reparametrisations, with n the size of the target class and m the number of attributes. Hyperparameters k and l rounded to the nearest integer in the range $[1, n-1]$.

Data descriptor	Hyperparameter	Reparametrisation	Resolution	Window size
NND	k		1	3
LNND	k	$a \log n$	0.01	101
LOF	k	$a \log n$	0.01	101
SVM	ν		0.1	11
	c	$c'm$	0.1	$\times 11$
ALP	k	$a \log n$	0.1	11
	l	$b \log n$	0.1	$\times 11$

distributions, including some that may be heavily skewed, we have chosen a robust measure of scale: the interquartile range (Rousseeuw & Croux 1993). By rescaling attribute values to match the interquartile range of the target class, the central half of all attribute values are brought to the same scale, which is not influenced by extreme values and incidental outliers.

For EIF, we use the implementation provided by the authors[5], while for SVM and IF, as well as for nearest neighbour searches, we use implementations provided by scikit-learn. We have implemented SAE in Keras and TensorFlow based on the code provided by the authors[6], and we use our own Python wrapper fuzzy-rough-learn (Appendix A) for MD, NND, LNND, LOF and ALP.

We evaluate the performance of a data descriptor with a specific combination of hyperparameter values by first calculating the mean AUROC for each dataset, and then the mean of these mean values across datasets. This ensures that datasets with a large number of classes do not dominate the final result.

During initial experimentation, we found that we obtain better overall results by reparametrising some hyperparameters in terms of the number of instances or features. We also found that for small hyperparameter differences, the response in average AUROC becomes noisy. We can interpret this noisiness as a natural limit on the resolution with which it makes sense to determine optimal default hyperparameter values, at least on the basis of our current selection of datasets.[7] To make the results more robust, we apply a rolling mean with a centered window

[5] https://github.com/sahandha/eif

[6] https://github.com/vanloicao/SAEDVAE

[7] In the case of SVM, another more practical limit is computation time. This is less of an issue for the other data descriptors because we can reuse nearest neighbour searches, and because the number of possible values for k and l is finite.

(two-dimensional in the case of SVM and ALP). The reparametrisations, the chosen resolution and the window size are listed in Table 7.1.

In addition to the number of nearest neighbours k, NND, LNND and LOF also require a choice of dissimilarity measure. We will see in Section 7.2 that NND, LNND and LOF obtain near-uniformly better results with the Boscovich than with the Euclidean metric, so we decided to simplify the rest of the experiments by limiting them to the Boscovich metric.

We will first use this setup to identify and report recommended default hyperparameter values for each data descriptor.

Next, we compare the performance of all data descriptors as black-box classifiers with predetermined hyperparameter values. As in Chapter 3, to ensure that the comparison is fair and the results generalise to other datasets, we use a leave-one-dataset-out scheme, where for each dataset we use those hyperparameter values that maximise AUROC across the other datasets. Our first question is whether our proposal, ALP, introduced in the previous chapter, performs better than the existing data descriptors. But we will also have the opportunity to test whether any of the existing data descriptors can be said to outperform each other.

We would like to compare data descriptors by applying Wilcoxon signed-rank tests. However, our observations contain one-class classification problems derived from the same dataset. Even though data descriptors only use training data from the target class, which differs for each observation, they are still based on the same type of data, and we cannot assume that these observations are completely independent. To address this, we choose to apply clustered Wilcoxon signed-rank tests (Rosner et al 2006), as implemented in the R package clusrank (Jiang et al 2020).

For each data descriptor, we test whether it performs better than the eight other data descriptors with a series of eight one-sided clustered Wilcoxon signed-rank tests. We apply the Holm-Bonferroni method (Holm 1979) to correct for family-wise error. The Holm-Bonferroni method indexes the eight uncorrected p-values from small to large, and then derives the corrected values as $\tilde{p}_i = \max_{j \leq i}(9 - j) \cdot p_j$.

Finally, we will also have a look at the ability of the data descriptors to scale to large datasets, by measuring the construction and query times of one-class classification with a series of training sets of increasing size. For this purpose, we draw random samples from the large *higgs* dataset (Section B.2). At each size, we report the average single-threaded computation time based on five such samples. We query with a test set of 1024 instances, and report the average query time per instance.

Table 7.2: Optimal default hyperparameter values of data descriptors, with n the size of the target class and m the number of attributes. Hyperparameters k and l rounded to the nearest integer in the range $[1, n-1]$.

Data descriptor	Hyperparameter	Optimal default value
NND	k	1
LNND	k	$3.4 \log n$
LOF	k	$2.5 \log n$
SVM	ν	0.20
	c	$0.25m$
ALP	k	$5.5 \log n$
	l	$6.0 \log n$

Table 7.3: Weighted mean rank, AUROC and standard deviation of AUROC across cross-validation folds (CVSD) of data descriptors.

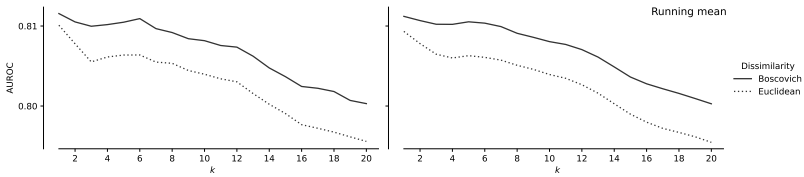
Data descriptor	Rank	AUROC	CVSD
ALP	3.51	0.817	0.0378
SVM	3.90	0.814	0.0404
NND	4.33	0.805	0.0414
LOF	4.67	0.803	0.0417
MD	4.68	0.806	0.0370
IF	5.15	0.800	0.0479
EIF	5.59	0.790	0.0504
LNND	6.55	0.781	0.0438
SAE	6.63	0.757	0.0539

7.2 Results and analysis

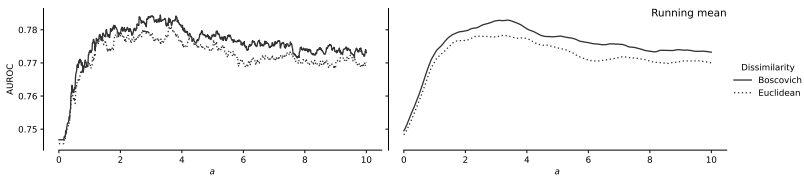
Figure 7.1 displays the average AUROC of the data descriptors with hyperparameters. It is clear that NND, LNND and LOF produce better overall results with Boscovich distance than with Euclidean distance. After applying the running mean, the AUROC graphs approximate 1- or 2-dimensional unimodal curves with a prominent global maximum. For SVM and ALP, the mean weighted AUROC decreases quite slowly away from the global maximum, which means that these data descriptors are quite robust to small small changes in their hyperparameter values. On the basis of these graphs, we can recommend the default hyperparameter values listed in Table 7.2. To avoid unwarranted precision, we have chosen to only determine these values up to multiples of 0.1 (LNND and LOF), 0.05 (SVM), and 0.5 (ALP).

The full leave-one-dataset-out AUROC values of the data descriptors are listed in Table C.1. The weighted mean rank and AUROC[8] of the

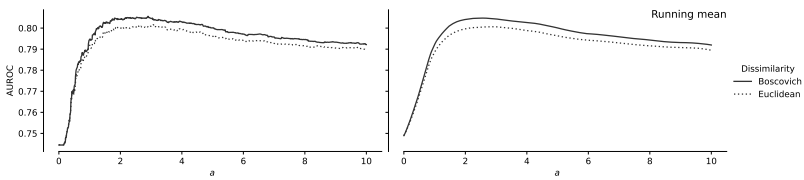
[8]Note that for ALP, SVM, NND, LOF and LNND, the weighted mean AUROC is lower than the highest scores in Figure 7.1, since, in general, the leave-one-dataset-out



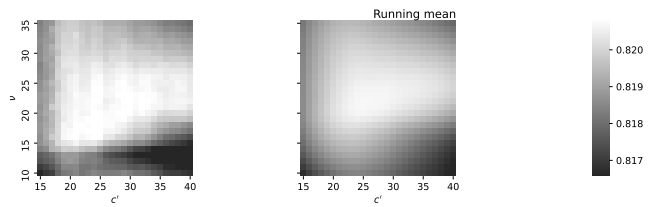
(a) NND



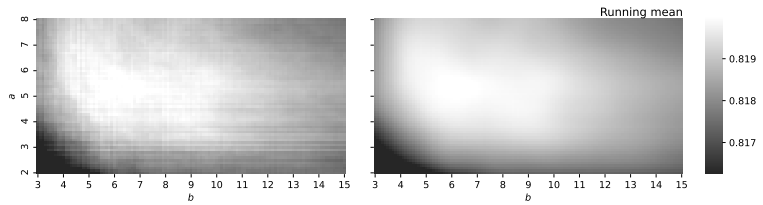
(b) LNND



(c) LOF



(d) SVM



(e) ALP

Figure 7.1: Weighted mean AUROC of data descriptors across their respective hyperparameter spaces.

Table 7.4: One-sided p-values of clustered Wilcoxon signed-rank tests of the hypotheses that the data descriptor in the row is ranked higher than the data descriptor in the column, with Holm-Bonferroni family-wise error correction applied to each row.

	SVM	NND	LOF	MD	IF	EIF	LNND	SAE
ALP	0.14	0.018	< 0.0001	0.035	0.035	0.00076	< 0.0001	< 0.0001
SVM		0.51	0.029	0.050	0.042	< 0.0001	< 0.0001	< 0.0001
NND			≥ 1	0.47	0.38	0.017	< 0.0001	< 0.0001
LOF				≥ 1	0.83	0.036	< 0.0001	0.00028
MD					0.35	0.11	0.0066	< 0.0001
IF						≥ 1	0.20	0.0068
EIF							0.78	0.025
LNND								0.27

data descriptors are summarised in Table 7.3. Because the AUROC for each dataset and target class is calculated through cross-validation, we can also measure the standard deviation across folds to gain an idea to what extend the performance is influenced by random factors. Generally speaking, we find that the lower-ranked data descriptors are more susceptible to statistical variation. We might reasonably expect to be able to improve their performance by making them more robust to statistical variation. The principal exceptions to this trend are MD and LNND, which show relatively low statistical variance, and this suggests a more limited potential for improvement.

Next, we test which data descriptors can be said with certainty to perform better than others on the type of one-class classification problems represented by our sample. Table 7.4 contains the p-values of the one-sided clustered Wilcoxon signed-rank tests, corrected for family-wise error with the Holm-Bonferroni method.

We find that there is strong evidence that ALP outperforms the other data descriptors, except SVM, for which there is only weak evidence ($p = 0.14$). In particular, these results confirm ($p < 0.0001$) our hypothesis that ALP improves upon LOF and LNND. Among the other data descriptors, there is good evidence that SVM performs better than the others, except NND and MD (for which the evidence is weaker: $p = 0.050$), as well as strong evidence that LOF and NND outperform EIF, LNND and SAE, that MD outperforms LNND and SAE, and that IF and EIF outperform SAE, and weak evidence that MD performs better than EIF ($p = 0.11$).

We note that the performance of IF is the hardest to pin down: in fact we can only say with limited certainty that ALP ($p = 0.035$) and SVM ($p = 0.042$) perform better. In particular, we find no evidence that

hyperparameter values are not equal to the optimal values calculated on the basis of all datasets.

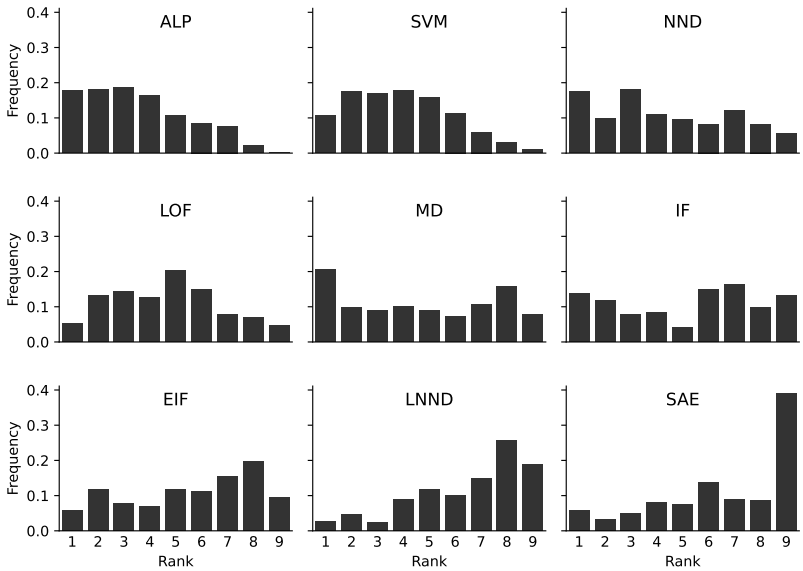


Figure 7.2: Average frequency (y-axis) of ranks (x-axis) of data descriptors, weighted by dataset, with ties distributed evenly among the respective ranks.

EIF performs better than IF. This may be because the type of dataset that motivated the formulation of EIF is not represented in our sample.

To gain more insight into the typical performance of the data descriptors, we have summarised the frequency with which a data descriptor achieves a certain rank in Figure 7.2. For ALP, SVM, LOF and LNND, the distributions of these frequencies are close to unimodal. In contrast, NND, MD, IF and EIF are less consistent — their performance is good or bad as or more often than it is mediocre. The poor overall performance of SAE is reflected in the fact that it is the worst-ranked data descriptor for nearly 40% of one-class classification problems. Yet it is also the best-ranked data descriptor in around 5% of them.

Figure 7.3 shows the extent to which the performances of the data descriptors correlate, as reflected by the weighted Kendall's τ (Vigna 2015). The strongest correlation can be found between the nearest neighbour based data descriptors and SVM, and between IF and EIF. This information is potentially useful when constructing ensembles of data descriptors. In addition to selecting data descriptors with good individual performance, it may be desirable to select pairs of data descriptors with low correlation that may be expected to complement (e.g. ALP and IF) rather than reinforce (e.g. ALP and LOF) each other.

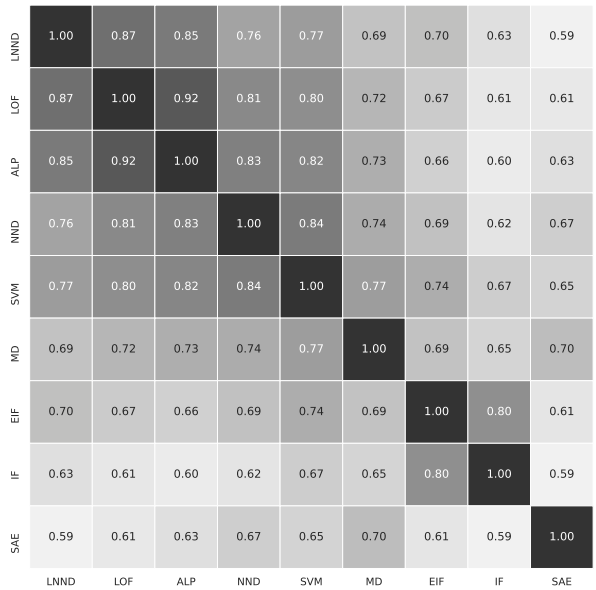


Figure 7.3: Weighted Kendall's τ between data descriptors.

Indeed, when comparing all pairs of data descriptors on the basis of the maximum of their respective AUROC scores for each one-class classification problem, we find that the five best performing pairs are (ALP, MD), (NND, MD), (ALP, NND), (ALP, IF) and (NND, IF).

There are a number of factors that potentially explain the performance of the data descriptors: the number of instances n, the dimensionality m and the sparsity of the target data. In addition, we can use the median AUROC obtained by the data descriptors in this chapter as an indication of the 'simplicity' or 'separability' of the evaluation data.

When we fit weighted linear regression models on the ranks of the data descriptors in terms of $\log n$, $\log m$, sparsity and median AUROC, we find that median AUROC is a positive significant factor for NND, LOF, ALP and MD, and a negative significant factor for LNND, IF and SAE. Surprisingly, there is no strong evidence that sparsity ($p = 0.400$) or dimensionality ($p = 0.802$) are positive factors for SAE. Unsurprisingly, given that SAE is a neural network, size *is* a significant positive factor ($p = 0.002$), with each order of magnitude (base e) increasing the rank of SAE by 0.31 places (± 0.19). Among the other data descriptors, sparsity may be a negative factor for LOF ($p = 0.058$).

The effect of median AUROC is illustrated in Figure 7.4. We see that the performance of ALP and SVM is broadly similar, except for the evaluation data that is the easiest to separate. This is the largest

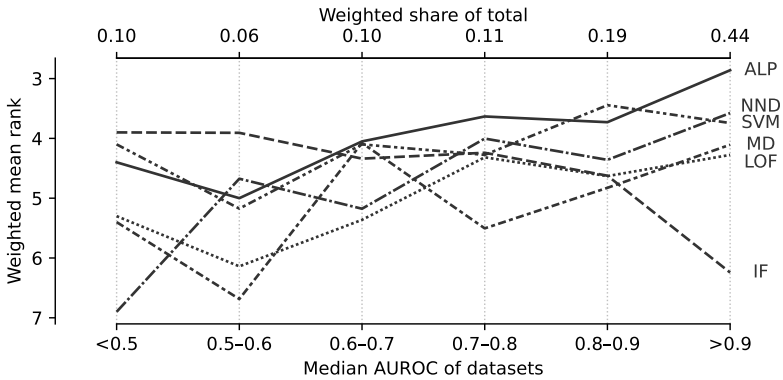
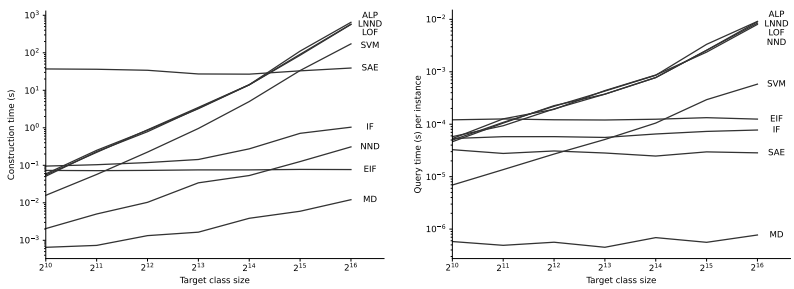


Figure 7.4: Weighted mean rank (y-axis) of selected data descriptors, stratified by the median AUROC of datasets across all data descriptors (x-axis).



(a) Construction times

(b) Query times

Figure 7.5: Mean construction and query times of data descriptors across five runs. Query times calculated on the basis of 2^{10} instances.

stratum in our sample, and it is the stratum for which ALP has the biggest relative advantage. ALP also has a fairly constant advantage over LOF across all strata. Interestingly, while IF performs suboptimally on evaluation data that is easily separable, it is competitive on harder problems.

Figure 7.5 contains the construction and query times of the data descriptors. The absolute times are implementation-dependent, but the way these times scale is nevertheless informative. There is a clear division between the nearest neighbour based data descriptors and SVM on the one hand, for which query times grow polynomially with training set size, and the other data descriptors, which keep query

times near-constant. The query times of NND, LNND, LOF and ALP are near-identical, since they all revolve around a nearest neighbour query. The faster construction times of NND reflect the fact we only pre-calculate local distances for LNND, LOF and ALP. The construction times for SAE start high, but remain effectively constant, although this is dependent on the precise choice of early-stopping criteria. Similarly, the construction times of IF and EIF are constrained by the constant number of trees and number of samples per tree.

When comparing classification accuracy and computational performance, we see that the data descriptors present different possible compromises. ALP and SVM offer the highest accuracy, but slow run times with large datasets, whereas IF, MD and SAE offer much better computational performance with large datasets but lower accuracy. We note that in the case of ALP and SVM, this trade-off can be further tweaked by incorporating approximative nearest neighbour search and support vector machine algorithms.

7.3 Conclusion

This chapter has produced two main results. On the basis of a large collection of one-class classification problems derived from real-world datasets, we have determined optimal default hyperparameter values for ALP, as well as the existing data descriptors NND, LNND, LOF and SVM. And secondly, we have evaluated the performance of the data descriptors through a leave-one-dataset-out procedure, and found weak evidence that ALP outperforms SVM, and strong evidence that it outperforms all other data descriptors, in particular LNND and LOF.

At the beginning of this chapter, we pointed out that the data descriptors MD, IF, EIF and SAE have no user-tunable hyperparameters and may therefore appear simpler to apply, and even preferable when no data is available for tuning. The optimal default hyperparameter values identified in this chapter allow all of the data descriptors discussed here to also be used as "easy-to-use black-box method for practitioners", as called for by Schölkopf et al (1999). Moreover, we have shown that with their default hyperparameter values, ALP and SVM generally perform better than MD, IF, EIF and SAE. The ability to further optimise these hyperparameters for specific datasets should therefore be seen as an additional advantage, not a necessary requirement.

We see a number of avenues for further research. Firstly, it may be possible to further reparametrise some hyperparameters in a way that would allow the identification of even better performing default values. Secondly, we set out to evaluate general data descriptor performance using a variety of different one-class classification problems, but it may also be worthwhile to investigate whether certain data descriptors are

better suited to specific types of problems. Thirdly, we have evaluated novelty detection, and it remains to be seen whether the results in this chapter can be reproduced for unsupervised outlier detection. And fourthly, given the better computational performance of MD, IF, EIF and SAE, one could explore the trade-off between accuracy and run time — on the one hand, ALP and SVM could be sped up by incorporating approximative algorithms, while on the other hand, it may be possible to modify the IF, EIF and SAE algorithms to improve their accuracy, at some computational cost.

In the next chapter, we will again compare these data descriptors, but this time in a setting with per-dataset hyperparameter tuning.

Chapter 8

One-class classification with hyperparameter optimisation[1]

Having looked in the previous chapter at one-class classification performance with default hyperparameter values, we now consider what happens if we are allowed to optimise, or 'tune', hyperparameter values using negative data. Notably, for all of the concrete applications of one-class classification listed in the Introduction of Chapter 6, there was negative data that could have been used for hyperparameter optimisation. In addition, negative data is always available when data descriptors are used as building blocks in a multi-class classification ensemble, which we consider in the next chapter.

As mentioned in the previous chapter, there have been two previous evaluations of data descriptors with hyperparameter optimisation. These evaluations had partially contradictory results. Janssens et al (2009) compared five data descriptors, and found that SVM, LOF and LNND significantly outperform the Parzen Window (PW) and Local Correlation Integral (LOCI) data descriptors, as well as weak evidence that SVM and LOF outperform LNND. Swersky et al (2016) replicated this experiment, and found instead that LNND performs no better than PW and LOCI, plus weak evidence that SVM outperforms LOF. In addition, Swersky et al (2016) tested 6 more descriptors, of which NND also performed very well.

Both previous experiments had several limitations that make a re-evaluation worthwhile. The first question that we will try to answer is: "What is the best way to optimise the hyperparameter values of a data descriptor?". Whereas both previous works use a naive grid search to optimise hyperparameter values, we will evaluate a representative selection of optimisation algorithms, and identify the most effective strategy. We will also explain how data descriptors based on near-

[1]This chapter is based on Lenz et al (2022c).

est neighbour searches can be optimised using efficient leave-one-out validation instead of ordinary cross-validation.

A second question that we address is: "How long should hyperparameter optimisation run for?" Besides being inefficient, grid search also requires fixing the total number of evaluations before optimisation begins. In contrast, the optimisation algorithms that we consider sequentially select points in the hyperparameter space to evaluate. This allows us to present our results in terms of the number of evaluations, giving practitioners more insight into the effect of this choice.

We compare the data descriptors that have one or more optimisable hyperparameters: ALP, SVM, NND, LNND and LOF. In the experiments performed by Janssens et al (2009), SVM and LOF were tied, ahead of LNND, but the difference was not statistically significant. Swersky et al (2016) ranked SVM, NND, LOF and LNND from high to low in that order, but only the difference between SVM and LNND was statistically significant. The former study did not include NND, and neither study included ALP, which had not yet been proposed at the time. Thus, ALP remains untested in the context of hyperparameter optimisation, whereas we found in the previous chapter that it is the best-performing data descriptor with default hyperparameter values (although the difference with SVM was only weakly significant).

Both previous studies evaluated performance with a Nemenyi test on mean ranks (Demšar 2006). This forced them to amalgamate results from one-class classification problems derived from the same dataset, of which there were 24 in Janssens et al (2009) and 30 in Swersky et al (2016). In order to get more statistical power, we draw from a larger number of datasets (50), and we compare pairs of data descriptors using a clustered Wilcoxon signed rank test (Rosner et al 2006) that allows us to use the full results from all 246 one-class classification problems. This choice is additionally motivated by the criticism that the p-value generated by the Nemenyi test for a pair of machine learning algorithms is too strongly dependent on the inclusion of other algorithms in the comparison (Benavoli et al 2016).

Together, these improvements over the previous studies — using a suitable optimisation procedure, more datasets and a more precise statistical test, and including ALP — allow us to provide a stronger answer to our third and final question: "What is the best data descriptor for one-class classification with hyperparameter optimisation?"

We proceed by discussing our selection of data descriptors (Section 8.1) and optimisation algorithms (Section 8.2), explaining how our experiments are structured (Section 8.3), discussing the results of these experiments (Section 8.4) and presenting our conclusions (Section 8.5).

8.1 Data descriptors

In this section, we discuss how we will optimise the hyperparameters of our data descriptors. The data descriptors were defined in detail in Chapter 6. Here we focus on how the hyperparameters of these data descriptors can be optimised. The goal in each case is to maximise AUROC.

Support Vector Machine

Recall that the Schölkopf variant of SVM fits a hyperplane between most of the target data and the origin, at a maximal distance to the origin, and that we apply a Gaussian kernel to transform the problem space. Accordingly, SVM has two hyperparameters that can be optimised. $\nu \in (0, 1]$ controls the relative weight placed respectively on maximising distance to the origin, and not leaving training instances on the same side as the origin. $c \in (0, \infty)$ parametrises the 'width' of the Gaussian kernel. Because many of the optimisation methods that we consider require compact hyperparameter domains, we reparametrise c as $\frac{c'}{1-c'}$, and optimise c' in $[10^{-6}, 1 - 10^{-6}]$, and restrict the domain of ν to $[10^{-6}, 1]$.

In order to optimise ν and c', we apply stratified five-fold cross-validation to obtain five splits of the available training data into a smaller training set and a validation set. For each training set, we select the target class instances to obtain a target set. We evaluate a pair of hyperparameter values by constructing a model on the target set, querying with the respective validation sets, and calculating the mean of the resulting AUROC scores. This means that optimising the hyperparameters of SVM requires constructing five models for each pair of values to be evaluated.

Nearest Neighbour Distance

In principle, the distance measure to be used with NND can be chosen freely. However, in order to allow the efficient form of optimisation discussed below, we fix this choice to Boscovich distance, which, as we saw in the previous chapter, generally gives better results than Euclidean distance. This leaves the nearest neighbour number k as the only hyperparameter to be optimised. Since it encodes a magnitude, we optimise k logarithmically. To avoid having to work with extremely large arrays, and knowing that its optimal default value is simply 1 (Table 7.2), we limit k to $\min(n, 100 \log n)$.

Because NND is so simple, k can be optimised more efficiently than the hyperparameters of SVM. Firstly, it is not necessary to completely recalculate a new NND model for each value of k. if $k_{\max}$ is the maximum value for k that we want to consider, we only require a single sorted $k_{\max}$

nearest neighbour query, since this contains all kth nearest neighbours for $k \leq k_{\max}$.

Secondly, instead of five-fold cross-validation, we use an efficient form of leave-one-out cross-validation, where each validation set contains a single instance. For five-fold cross-validation, we would have to create five nearest neighbour search models, one for each target set corresponding to a fold. To perform leave-one-out cross-validation, we create a single nearest neighbour search model on the basis of all records from the target class. For each target class record, we must ensure that it is not also part of the target set corresponding to its fold, so we query to obtain its $k_{\max} + 1$ nearest neighbours, and remove the first nearest neighbour distance (with value 0). For negative instances, there is nothing to correct and we can simply perform a $k_{\max}$ nearest neighbour query. We collect the resulting scores and calculate a single validation AUROC.

Localised Nearest Neighbour Distance

As with NND, we adopt Boscovich distance and optimise k logarithmically, up to $\min(n, 100 \log n)$. We can also perform efficient leave-one-out cross-validation as with NND, but we have to do some additional work to obtain a correct result. For each target class record, we have to check whether it is among the k nearest neighbours of its kth nearest target class neighbour, and if so, substitute the $k + 1$th nearest neighbour.

Local Outlier Factor

Again as with NND and LNND, we adopt Boscovich distance and optimise k logarithmically, up to $\min(n, 100 \log n)$. The calculation of LOF requires determining the kth nearest neighbour distance of the ith nearest neighbour of the jth nearest neighbour of a test instance (for all $i, j \leq k$). For this reason, it is no longer feasible to apply the efficient form of leave-one-out cross-validation described for NND and LNND, and we resort to performing stratified five-fold cross-validation as with SVM. However, we still retain the efficiency that for each fold, we only need one query to obtain the $k_{\max}$ nearest neighbours of a test instance.

Average Localised Proximity

ALP has two nearest neighbour hyperparameters to be optimised. k and l respectively determine the scale at which nearest neighbour distances are considered and the amount of localisation. As with NND, we optimise k and l logarithmically. Similarly to NND and LNND, we only need a single $\max(k_{\max}, l_{\max})$ nearest neighbour query if we want to evaluate values of k and l up to $k_{\max}$ and $l_{\max}$. Since ALP is less

complex than LOF, we can also apply efficient leave-one-out validation, by performing essentially the same correction as for LNND. In particular, we correct the local distances of a neighbour x of y by considering its $k+1$ nearest neighbours, and removing either the distance to y or the $k+1$th nearest neighbour distance.

Increasing k and l has two effects: it draws in more distant neighbours of y, and it decreases the slope of the weight vectors, making the contribution of successive neighbours more equal. The asymptotic limit of this process is a weight vector with equal weights everywhere. A more practical limit is that k and l cannot grow beyond the number of target instances n. However, we can simulate higher values for k and l by truncating the weight vector and multiplying by a constant to ensure that its sum still equals 1.

This also allows us to address a computational issue with large datasets, that evaluating a pair of values for k and l on the whole training set involves $(k+1) \cdot l \cdot n$ distance values. If all three values are large, processing a single array with all distance values requires a very large amount of memory. For these reasons, we let k and l range up to $5n$, but truncate distance values and weight vectors after $\min(n, 20 \log n)$. This is informed by the knowledge that the optimal default values for k and l are $5.5 \log n$ and $6 \log n$ respectively.

8.2 Optimisation algorithms

Optimisation problems are typically formulated in terms of a problem function $f : P \longrightarrow \mathbb{R}$, where the *problem space* P is a subspace of $\mathbb{R}^m$ for some m that is often required to be compact. Depending on the context, the goal of optimisation is to find points in P that minimise or maximise f. In the present instance, we wish to maximise the validation AUROC of our data descriptors, and the problem space is determined by the hyperparameters that we optimise.

There is an important distinction between algorithms that aim to find a local optimum of f in P, and those that aim to identify the global optimum of f in P. An essential characteristic of global optimisation algorithms is that they have to balance the exploration of areas of the problem space with large uncertainty and the exploitation of areas where function performance is known to be good.

An intrinsic disadvantage of local optimisation algorithms is that they require a choice of starting point, and may get stuck in a local optimum if this starting point is chosen poorly. However, the optimisation problems of finding the best hyperparameter values for our data descriptors may be close to unimodal/quasiconvex (Stephenson et al 2021), in which case this risk could be relatively small. For this reason, we include two classical local optimisation algorithms that are easy to implement.

Random search

Purely random search is a surprisingly potent global optimisation strategy. By continuing to evaluate arbitrary points in the problem space, we can expect to eventually get arbitrarily close to the global optimum. In particular, random search is a more efficient algorithm than grid search (Bergstra & Bengio 2012). Because this strategy uses no information from previous evaluations, it serves as a good baseline.

Hooke-Jeeves

The local search algorithm proposed by Hooke & Jeeves (1961) passes through the problem space in steps. Each step follows a pattern, which is the vector sum of a number of substeps along each coordinate axis. The pattern is adjusted with each step, by optionally adding or subtracting a substep along each coordinate axis, depending on which option results in the greatest improvement in the objective function value. If the pattern cannot be adjusted to produce a step with any improvement, a new pattern is created from scratch. If no such pattern can be found either, the substep size is decreased.

We use the implementation provided by pymoo (Blank & Deb 2020) and its default values. As starting values we use the optimal default values identified in the previous chapter.

Nelder-Mead

The local search algorithm proposed by Nelder & Mead (1965) is based on an earlier proposal by Spendley et al (1962). It iterates on $m + 1$ points that can be viewed as the vertices of an m-dimensional simplex. The algorithm lets this simplex 'walk' through the problem space by replacing its worst vertex in each iteration. Each step is directed towards the mid-point of the remaining vertices. The new vertex is either placed between the worst vertex and this mid-point (shrinking the simplex), or beyond the midpoint (reflecting and optionally extending it). If none of these options improve upon the worst vertex, the entire simplex is shrunk, with only the best vertex remaining in place.

The theoretical performance of Nelder-Mead optimisation had long been unclear, until Torczon (1989) demonstrated with a concrete example that Nelder-Mead can converge on points that are not local optima, even with problems that are twice differentiable. Nevertheless, Nelder-Mead optimisation has been very popular due to its relative simplicity, and because it seems to converge very quickly in many simple practical applications (Wright 1995).

We use the implementation provided by SciPy (Virtanen et al 2020), with starting simplices centred around the optimal default values identified in the previous chapter.

Kushner-Sittler

A global optimisation method was first proposed by Kushner (1962, 1964), based in part on unpublished work by Robert A Sittler (see Betrò (1991), DR Jones (2001) and Brochu et al (2009) for overviews of later developments). This has been referred to as simply *global optimisation*, or *Bayesian optimisation*, because its central idea is to iteratively use the Bayesian information criterion to select the next point in the problem space to evaluate. We assume that the unknown problem function is drawn from a random distribution of functions, which is traditionally modelled as a Gaussian process. In each step, we can calculate the conditional probability $p(y|x)$ that the problem function will obtain certain values at a given point x in the problem space, in light of the function values that have already been calculated. These conditional probabilities are then reduced to a single score for each x with an activation function, and the point with the maximal such score is selected as the next point to evaluate. The activation function most often used today, already hinted at in Kushner (1962), is *expected improvement*, the expected value of $\max(y - y^*, 0)$ under the model for some $y^* \in \mathbb{R}$, typically the largest evaluated function value so far.

Kushner-Sittler optimisation transforms the original optimisation problem into a series of new optimisation problems for each iteration. These new optimisations incur a certain cost themselves, but this cost is only dependent on the dimensionality of the original problem, so for problems that are costly to evaluate, the trade-off is worthwhile. Note also that as with the original problem, these optima can generally only be approximated, but it is (often tacitly) assumed that this is not problematic.

We use the implementation provided by Emukit (Paleyes et al 2019), with the first point chosen randomly.

Bergstra-Bardenet

A more recent variant of Kushner-Sittler optimisation, motivated in particular by high-dimensional and conditional problem spaces, is the Tree-structured Parzen Estimator (TPE) proposal by Bergstra et al (2011). It lets the target value y^* correspond to a quantile of the evaluated function values. This induces a split between small and large values, and the corresponding distributions $p(y < y^*)$ and $p(y > y^*)$, which can be modelled with two Parzen Estimators $l(x)$ and $g(x)$. By reformulating $p(y|x)$ in terms of $p(x|y)$, $p(y)$ and $p(x)$, the authors then show that the

expected improvement of the original model is maximal when $\frac{g(x)}{l(x)}$ is maximal.

We use the Adaptive TPE implementation of hyperopt (Bergstra et al 2011).

Malherbe-Powell

Global optimisation algorithms often proceed from the assumption that the problem function satisfies certain smoothness conditions. In particular, for any $k > 0$, we can define the class of k-Lipschitz functions as those functions f that satisfy:

$$\forall x_1, x_2 \in A : \left|f(x_1) - f(x_2)\right| \leq k \cdot |x_1 - x_2| \tag{8.1}$$

Malherbe & Vayatis (2017) propose that we can use this assumption to restrict the search to certain parts of the problem space. They propose the LIPO algorithm, in which random candidates are drawn from the problem space, but only those candidates are evaluated that potentially improve upon the current optimum, in view of the candidates evaluated so far and (8.1).

In general, k may be difficult to estimate, and we simply want to assume that some such k exists for a problem function. For this purpose, Malherbe & Vayatis (2017) also propose the AdaLIPO algorithm. This alternates LIPO with purely random search, and increases k whenever (8.1) is no longer satisfied.

AdaLIPO was further modified into MaxLIPO and implemented into the dlib library by King (2009, 2017). MaxLIPO presents three improvements. Firstly, it incorporates a noise term that prevents k from approaching infinity if the problem function is not in fact k-Lipschitz because it has small discontinuities. Secondly, it employs separate values of k for each dimension. And thirdly, instead of selecting new candidates at random, it identifies the candidate with the largest potential improvement in light of (8.1).

A more fundamental problem of AdaLIPO that carries over to MaxLIPO is that while it is seemingly able to quickly locate the neighbourhood of the global optimum, it then takes much longer to approach the optimum itself. This can be understood, since by considering the maximal potential improvement, rather than some form of expected improvement, these algorithms place a greater emphasis on exploration than exploitation. To address this, King (2017) lets MaxLIPO alternate with the trust region approach of the local optimiser BOBYQA (Powell 2009), a bounded version of the earlier NEWUOA proposal (Powell 2004).

8.3 Experimental setup

In order to enable comparison with using default hyperparameter values, we closely follow the experimental setup of the previous chapter. In particular, we use the same collection of 246 one-class classification problems.

For each one-class classification problem, we apply stratified five-fold cross-validation. We measure the performance of a data descriptor with specific hyperparameter values in terms of AUROC. For each division, we measure validation AUROC using nested cross- or leave-one-out validation as explained in Section 8.1, as well as test AUROC by retraining the data descriptor on all of the training data and evaluating its performance on the test data.

We maximise validation AUROC by applying the optimisation algorithms from Section 8.2. Each optimiser is allowed a maximum budget of 50 evaluations of hyperparameter values. Although the NND, LNND, LOF and ALP hyperparameters k and l are discrete, in order to be able to apply the selected optimisation algorithms, we optimise them as if they were continuous.[2] As a result, subsequent steps of the optimisation search may target different points in the problem space that are discretised back to the same concrete value(s), which are not evaluated again. Thus, with local optimisers that have reached a local optimum, as well as with small datasets in general, the optimisation search may never reach 50 evaluations. Therefore, we terminate all optimisation searches after 100 steps.

We structure our analysis in terms of the number of evaluations. For each cross-validation division and number of evaluations, we use the hyperparameter values that maximise validation AUROC up to that point. We start our analysis by identifying the most suitable optimiser for each data descriptor, and adopt this for the rest of our analysis. We compare data descriptors both by summarising performance with the mean test AUROC, and by looking at individual results at the level of cross-validation divisions. In both cases, we apply a weighting scheme such that the 50 datasets from which the one-classification problems are drawn all contribute equally. In scatter plots, this is reflected in the size of the markers. We measure rank correlation with the weighted Kendall's τ (Vigna 2015).

In order to determine statistical significance, we apply clustered Wilcoxon signed-rank tests (Rosner et al 2006) on the mean AUROC scores across folds for each one-class classification problem. When we compare data descriptors to each other, we use the Holm-Bonferroni method (Holm 1979) to correct for family-wise error.

[2] We note that more generally, for any classification algorithm, performance metrics like AUROC, accuracy, precision and recall are non-continuous because a finite number of instances can only be subjected to a finite number of rankings.

8.4 Results and analysis

The results of our experiments allow us to answer the three questions raised in the Introduction.

What is the best way to optimise the hyperparameter values of a data descriptor?

Figure 8.1 shows the performance of the different optimisers with the data descriptors. We can get an idea of the difficulty of these optimisation problems by looking at the baseline random strategy. For NND, our maximum budget of fifty evaluations is essentially enough for random search to find the global maxima, while for LOF, it comes reasonably close. LNND and ALP are more difficult, and random search clearly lags behind the other optimisation strategies with SVM. Taking into account the dimensionality of the respective problem spaces, it appears that the problem curves of LNND and SVM are relatively hard to optimise.

It is clear that the two local optimisation methods, Nelder-Mead and Hooke-Jeeves, generally fail to find the global optima because they get stuck in a local optimum. Neither method performs clearly better than the other. Nevertheless, if ease of implementation is a larger priority than performance, they may be an acceptable option for ALP and SVM. For the data descriptors with one hyperparameter, simple random search is to be preferred.

Of the global algorithms, the performance of Kushner-Sittler is surprisingly poor, in particular with NND and LOF, where it appears to stall below the level reached by the best local algorithm. Closer inspection reveals that it is too strongly focused on exploitation over exploration, and will often evaluate long series of points in the problem space that are very close together. This may be due to the chosen implementation, or the fact that the problem functions induced by the hyperparameters k and l are locally constant.

The overall best-performing method is Malherbe-Powell, with Bergstra-Bardenet in clear second place. Malherbe-Powell finds the highest AUROC for all data descriptors, except NND, where the difference with Bergstra-Bardenet is minimal ($1.4 \cdot 10^{-4}$). Therefore, we will use the results of Malherbe-Powell for the rest of this section. However, we also note that some of the differences are very small, and practitioners may want to prioritise ease of implementation when selecting an optimiser.

Figure 8.2 shows the distribution of the hyperparameter values after 50 evaluations. These distributions are relatively uniform, which suggests that the chosen parametrisations are efficient, in the sense that the optimisation algorithm doesn't have to spend unnecessary evaluations

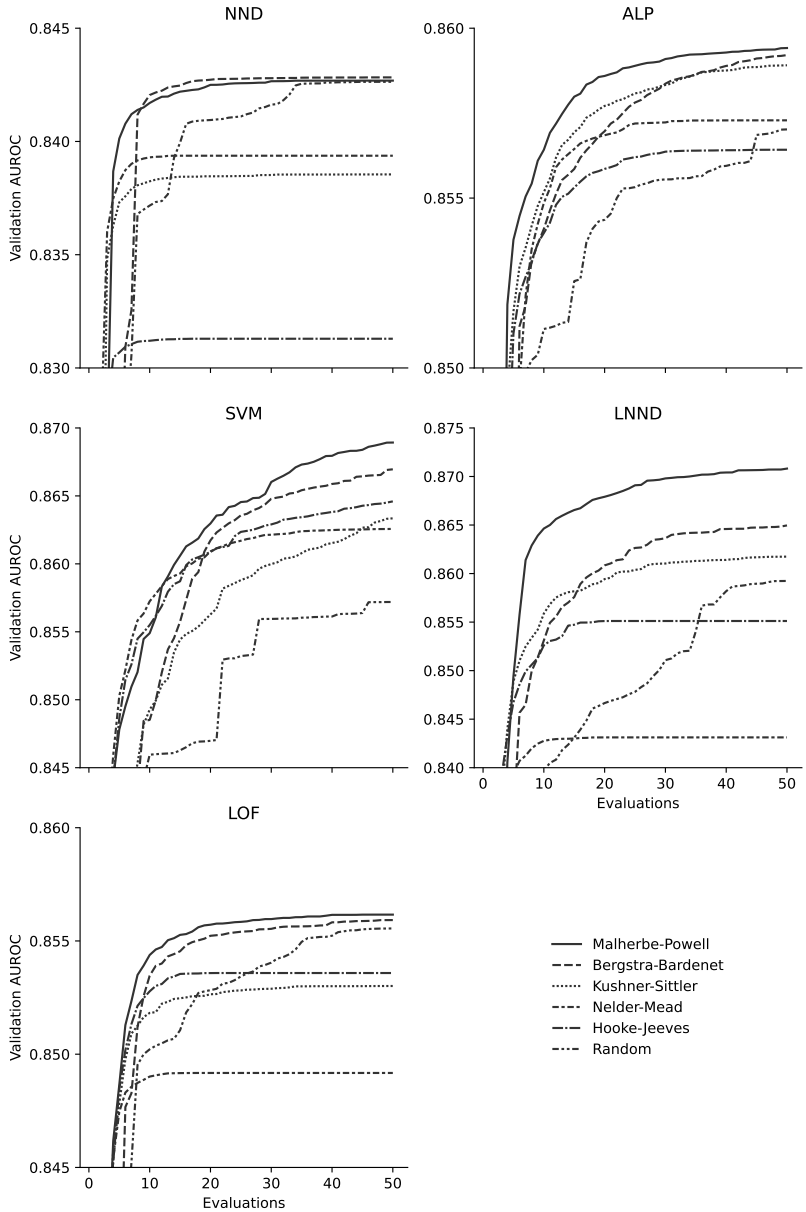


Figure 8.1: Weighted mean validation AUROC of data descriptors with hyperparameters optimised by a number of different algorithms.

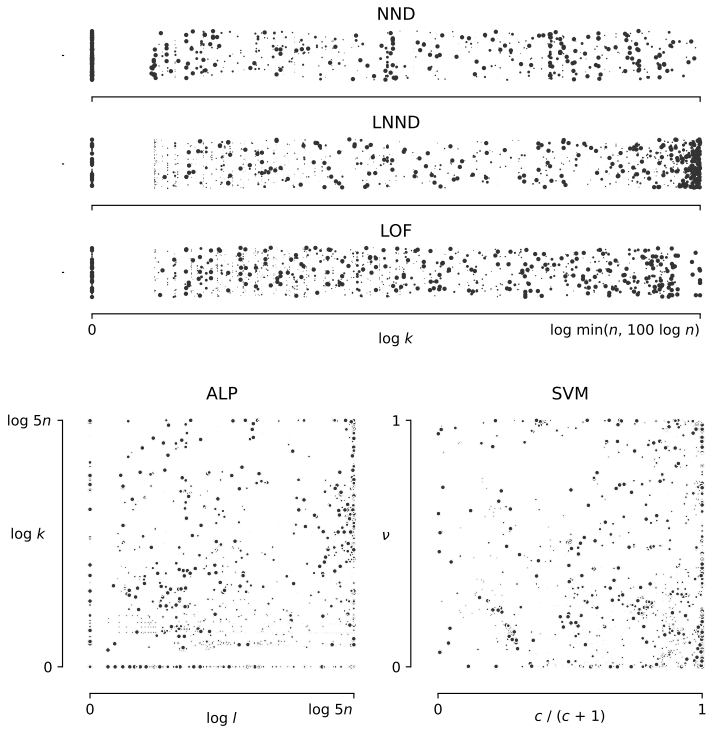


Figure 8.2: Distribution of selected hyperparameter values after 50 evaluations with Malherbe-Powell optimisation, for each of the 246 one-class classification problems and for each five-fold cross-validation division. Point size corresponds to the weight of a problem, which corresponds inversely to the number of problems derived from the same original dataset.

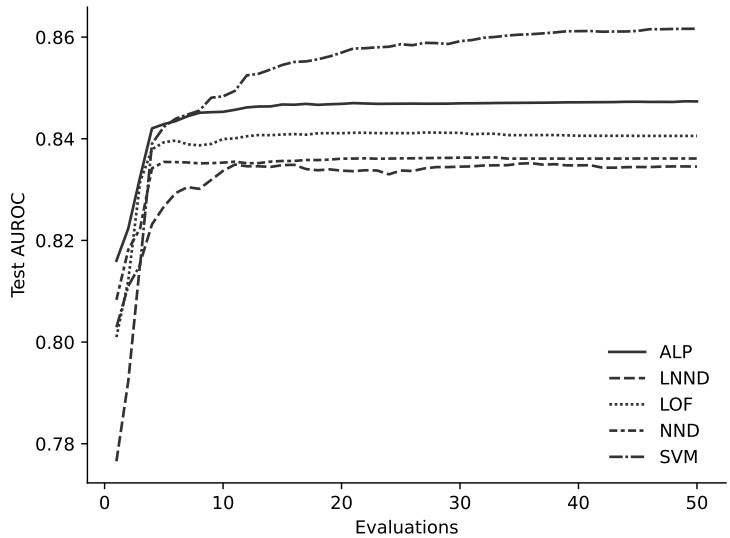


Figure 8.3: Weighted mean test AUROC of data descriptors, with hyperparameters optimised by the Malherbe-Powell algorithm.

exploring sparse areas of the hyperparameter space.[3] However, for LOF (7.3%), LNND (8.6%) and especially NND (39%), there is a substantial minority of problems for which the optimal value is simply 1. In the case of NND, for which 1 is the default value, this indicates that k often doesn't need to be optimised. For LNND, a large number of optimal values (23%) are within 1% of the maximum, but this is in most cases due to the fact that k cannot increase beyond n, rather than our imposed limit of $100 \log n$.

How long should hyperparameter optimisation run for?

Figure 8.3 shows the weighted mean test AUROC of the data descriptors. The data descriptors display varying sensitivity to hyperparameter optimisation. All test AUROC curves increase steeply for 4 evaluations and then flatten out. However, for SVM, the initial rise is steeper and its curve continues to increase for much longer, allowing it to surpass LOF, NND and ALP even though it starts quite low. LNND improves even more steeply, but because it starts out very poorly, remains the

[3] We also find for many problems that different cross-validation divisions optimise to different values, which may indicate that the response curves are somewhat flat and noisy.

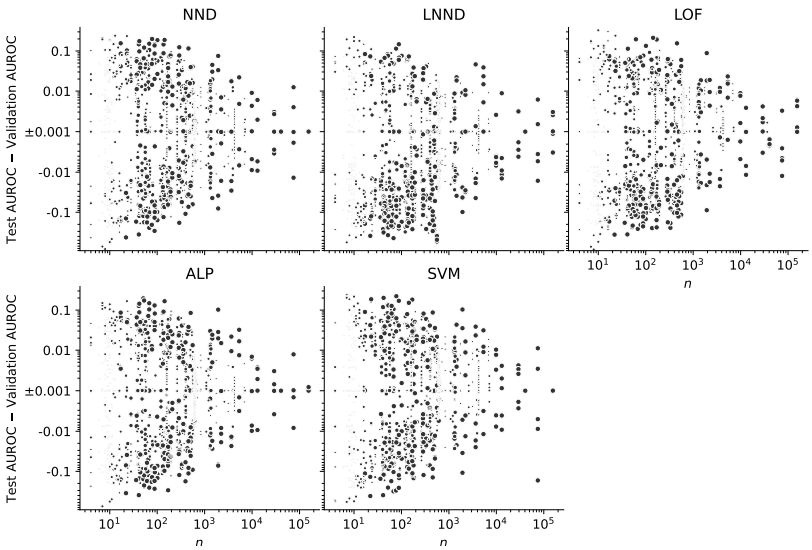


Figure 8.4: Difference between test and validation AUROC as a function of target class size n. Point size corresponds to the weight of a problem, which corresponds inversely to the number of problems derived from the same original dataset.

worst-performing data descriptor. The data descriptors approach their final scores (after 50 evaluations) to within 0.001 points after respectively 5 (NND), 10 (LNND and LOF), 13 (ALP) and 37 (SVM) evaluations.

The test AUROC curves in Figure 8.3 don't achieve the same scores as the validation AUROC curves for Malherbe-Powell optimisation in Figure 8.1. This difference can be interpreted as overfitting. Among the data descriptors optimised with leave-one-out validation, it is largest for LNND (0.036 after fifty evaluations), followed by ALP (0.012) and NND (0.0066), while for those optimised with five-fold cross-validation, it is larger for LOF (0.016) than for SVM (0.0073). Note that LNND and LOF, with one hyperparameter, show more overfitting than, respectively, ALP and SVM, with two hyperparameters. The weighted Kendall's τ for the amount of overfitting after 50 evaluations ranges from 0.31 for SVM and LNND to 0.59 for SVM and NND. The amount of overfitting should be taken into account when estimating test AUROC or selecting data descriptors on the basis of validation AUROC. However, the numbers cited above are dependent on the mix of datasets that we use: for all data descriptors, validation AUROC becomes increasingly accurate as the target set size grows (Figure 8.4).

When we apply a clustered Wilcoxon signed rank test to compare the test AUROC obtained with hyperparameter optimisation and the

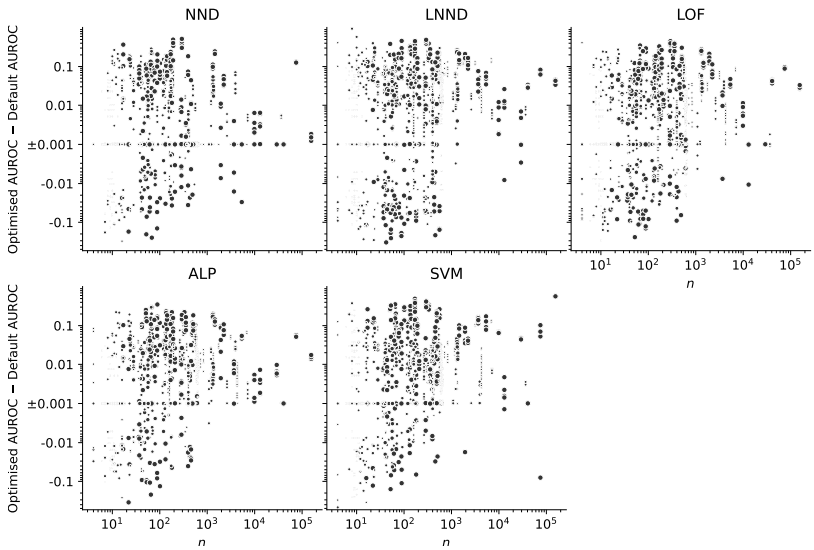


Figure 8.5: Increased or decreased AUROC due to hyperparameter optimisation over default hyperparameter values, as a function of target class size n. Point size corresponds to the weight of a problem, which corresponds inversely to the number of problems derived from the same original dataset.

test AUROC obtained with default hyperparameter values from the previous chapter, we find that optimised values start to outperform default values with great certainty ($p < 0.01$) within 2 (LNND, ALP), 3 (LOF, SVM) and 4 (NND) evaluations. With optimised values, SVM, LOF and NND also perform significantly better ($p < 0.01$) than ALP, the best data descriptor with default values, after 4 (SVM), 5 (LOF) and 20 (NND) evaluations. Even after fifty evaluations, LNND with optimised values still performs worse than ALP with default values. However, note that hyperparameter optimisation is not guaranteed to increase AUROC for any of the data descriptors, especially with smaller datasets (Figure 8.5).

What is the best data descriptor for one-class classification with hyperparameter optimisation?

The test AUROC scores after 50 evaluations are highly rank-correlated: the weighted Kendall's τ ranges from 0.74 (SVM and LNND) to 0.86 (SVM and NND). To determine whether the differences in performance are statistically significant, we perform one-sided clustered Wilcoxon signed rank tests. The resulting p-values after each evaluation are

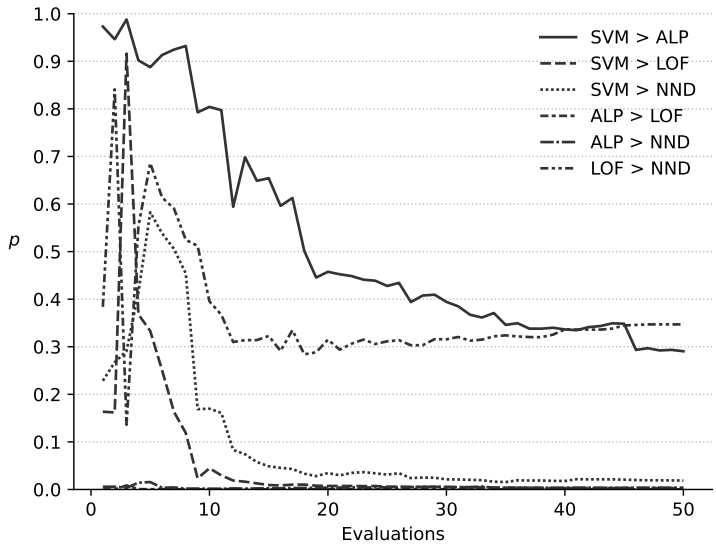


Figure 8.6: One-sided p-values of clustered Wilcoxon signed rank tests that one data descriptor is better than another (uncorrected for multiple testing).

Table 8.1: One-sided p-values of clustered Wilcoxon signed-rank tests of AUROC after 50 evaluations, testing row data descriptor > column data descriptor, with Holm-Bonferroni family-wise error correction applied to each row.

	ALP	LOF	NND	LNND
SVM	0.29	0.0079	0.037	< 0.0001
ALP		0.00091	0.0077	< 0.0001
LOF			≥ 1	< 0.0001
NND				0.00037

Table 8.2: Fraction of one-class classification problems with higher validation and test AUROC by ALP or SVM.

Validation AUROC	Test AUROC			
	ALP < SVM	ALP = SVM	ALP > SVM	Total
ALP < SVM	0.37	0.031	0.088	0.49
ALP = SVM	0.00080	0.033	0.0052	0.039
ALP > SVM	0.12	0.038	0.31	0.47
Total	0.49	0.10	0.40	

displayed in Figure 8.6. The p-value for the opposite test can be obtained by subtracting the respective value from 1. Tests with LNND are omitted from Figure 8.6 since the corresponding p-values don't rise above 0.01. Table 8.1 lists the p-values after 50 evaluations, corrected for multiple testing.

Based on these experiments, we can confidently say that with sufficient evaluations, ALP and SVM perform better than NND, LOF and LNND, and that NND and LOF also perform better than LNND. We have far less certainty about the relative performance of ALP and SVM, and of NND and LOF. Figure 8.6 suggests that LOF generally outperforms NND, and that when the number of evaluations is small, ALP outperforms SVM, and vice-versa when the number of evaluations is large, but there is a large possibility that these observations are simply due to chance.

If we focus on the performance of SVM and ALP after 50 evaluations (Table 8.2), we see that SVM obtains a higher AUROC than ALP slightly more often than vice-versa, both on validation and test data. This confirms that the average performance of ALP and SVM is very close in practice. However, for individual classification problems, the choice still matters. We note that which of these two data descriptor performs better is fairly consistent between validation and test data. If for each classification problem, we choose the data descriptor that obtains a higher validation AUROC (choosing ALP in event of a tie), it will perform worse on test data than the other data descriptor in only 21% of cases. The advantage of this combination of ALP and SVM over either of ALP or SVM on its own is highly significant ($p < 0.0001$), regardless of whether we choose for each fold separately or on the basis of the mean validation AUROC across folds.

One factor that plays a role in the relative performance of SVM and ALP is the difficulty of the one-class classification problem. For the purpose of the present analysis, we can express this as the maximum of the AUROC achieved by ALP and SVM. Figure 8.7 plots the relative performance of ALP and SVM against this difficulty. SVM is better able to separate more difficult problems, but for problems for which a good

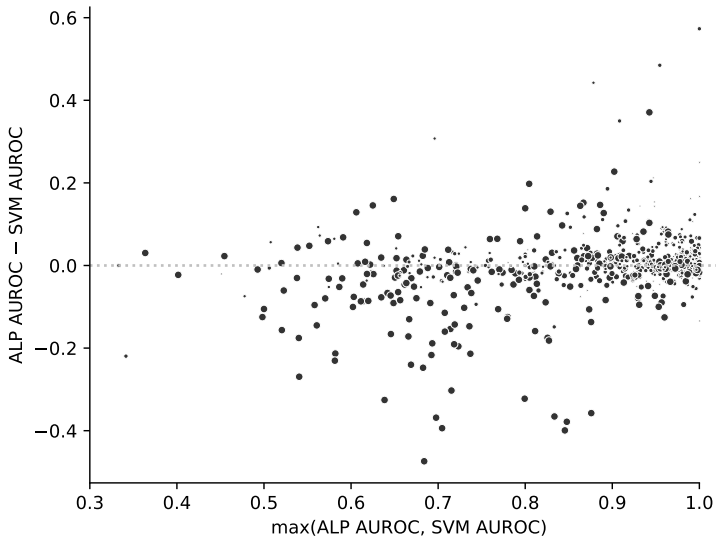


Figure 8.7: Difference between ALP and SVM AUROC, as a function of the difficulty of one-class classification problems, expressed by the maximum of ALP and SVM AUROC. Point size corresponds to the weight of a problem, which corresponds inversely to the number of problems derived from the same original dataset.

AUROC of 0.8 or more can be achieved, ALP beats SVM more often (46%) than vice-versa (41%), with a weighted mean difference of 0.0021.

Finally, Figure 8.8 displays the run times of hyperparameter optimisation as a function of the number of evaluations. These run times are implementation-dependent, but we can nevertheless make a number of broad observations. For SVM, run time is directly proportional to the number of evaluations, as no calculations are reused. NND, LNND and LOF can effectively be optimised in constant time, since the initial nearest neighbour queries dominate. The run time of LOF is higher because it uses five-fold cross-validation and needs five nearest neighbour queries. For ALP, we observe a considerable amount of additional run time per evaluation. Looking at individual evaluations, we find that their run time varies wildly, seemingly due to the computational load of working with large arrays when k and l are large.

The higher run time required by SVM for additional evaluations is compounded by the finding, reported above, that optimisation of SVM requires more evaluations than ALP, LOF, LNND and especially NND. This is illustrated by the fact that the curves of the last three data

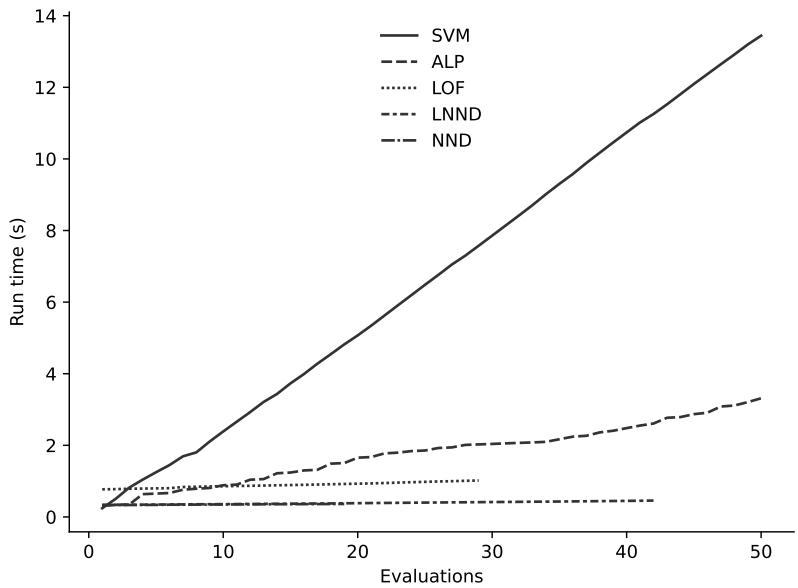


Figure 8.8: Mean run times (5 runs) of hyperparameter optimisation with Malherbe-Powell on a training set with 1000 target class instances and 1000 other instances, drawn from the *miniboone* dataset (target class 1).

descriptors in Figure 8.8 end before 50 evaluations.

8.5 Conclusion

In this chapter, we have presented a thorough analysis of hyperparameter optimisation for five data descriptors: SVM, NND, LNND, LOF and ALP. We have explained how NND, LNND and ALP can be optimised efficiently with a single nearest neighbour query and leave-one-out validation, while SVM requires building a new model for each additional hyperparameter evaluation. We then evaluated the performance of hyperparameter optimisation empirically.

From a selection of optimisation algorithms, the recent Malherbe-Powell approach provides the best overall performance with all five data descriptors. LNND and LOF are relatively sensitive to overfitting, but in all cases overfitting reduces with target set size. For all data descriptors, optimised hyperparameters significantly outperform default hyperparameter values after a handful of evaluations. As predicted, different hyperparameter values can be evaluated more efficiently for NND,

LNND, LOF and ALP than for SVM. In addition, these data descriptors also require fewer evaluations than SVM. After 50 evaluations, ALP and SVM significantly outperform LOF, NND and LNND, and LOF and NND in turn perform better than LNND. SVM also outperforms ALP on our datasets, but the difference is not significant.

A more detailed look at the difference between ALP and SVM revealed that their strengths are to some extent complementary, and that selecting one or the other based on their validation AUROC gives the best results. SVM has a strong relative advantage with difficult one-class classification problems, while ALP performs better with problems with which a good AUROC of 0.8 or higher can be achieved.

Overall, we come to the following conclusion. NND is a very simple data descriptor that can be optimised very efficiently. While the resulting gain in performance is limited, it nevertheless leads to results that are generally better than what can be obtained with a data descriptor with default hyperparameter values. SVM is a data descriptor with excellent performance, but it is expensive to optimise. The performance of ALP rivals that of SVM, and potentially surpasses it with one-class classification problems that admit a good solution, but it can be optimised much more efficiently.

Therefore, we find that ALP is a good default choice, while NND may appeal to practitioners constrained by a smaller computational budget. If the absolute best performance is desired, we recommend that practitioners consider both ALP and SVM, and make the choice dependent on validation AUROC.

In future research, we think that it could be worthwhile to investigate in greater detail what properties of datasets determine the relative strengths and weaknesses of ALP and SVM. A deeper understanding of this question could in turn be applied to modify the ALP and SVM algorithms. In addition, it would be useful if the computational cost of optimising SVM and ALP could be reduced.

We have focused our attention in this chapter on a handful of hyperparameters with the most immediate impact on the classification of different datasets. But these are not the only choices available to a practitioner. Hyperparameter optimisation is a specific form of model selection, and conversely, any modification to a classification algorithm can be seen as a hyperparameter choice. In particular, one large topic that we have set aside in the present chapter is the possibility to change how similarity and difference are measured, by choosing a different metric, kernel and/or scaling function. These are essentially open-ended choices, so part of the challenge lies in delineating the search area.

Another avenue for future research is the effect of the quality and quantity of the negative records that are available for hyperparameter optimisation. The aim of such an investigation could be to provide guidance as to when a negative sample is insufficiently representative for

binary classification, yet still good enough to make hyperparameter optimisation beneficial. However, this question may be difficult to answer in general terms, since samples of negative data can be unrepresentative in ways that are particular to a specific problem. In any case, establishing a collection of one-class classification problems that are not derived from multi-class problems would provide a useful reality check in this regard.

In the next chapter, we will apply hyperparameter optimisation to data descriptor ensembles in a multi-class setting.

Chapter 9

Fuzzy rough one-class ensembles

Fuzzy rough set theory (Chapter 1) and one-class classification (Chapter 6) are two seemingly unrelated fields of soft computing. In this chapter, we identify the conceptual overlap between these fields, and show in particular that upper and lower approximations are data descriptors, and consequently, that FRNN can be seen as a one-class classification ensemble. This insight allows us to apply results from one-class classification to obtain a more flexible definition of fuzzy rough sets. Conversely, the use of lower approximations in addition to upper approximations in fuzzy rough sets can inform the construction of one-class ensembles for multi-class classification.

One-class ensembles have been applied to problems like image segmentation (Goh et al 2005; Krawczyk et al 2014b), network intrusion detection (Giacinto et al 2008; Kassab 2021), malware detection (J Liu et al 2013), microarray classification (Krawczyk 2013), medical image classification (Zhang et al 2014), hypertension type classification (Krawczyk & Woźniak 2014b), breast cancer diagnosis (Krawczyk & Filipczuk 2014; Krawczyk et al 2014a) and handwriting recognition (Hadjadji et al 2019). They have been found to be particularly suited for problems with large numbers of classes (Krawczyk et al 2015; Mygdalis et al 2015), imbalanced classification problems in general (Hayashi & Fujita 2021; Krawczyk et al 2015), and imbalanced data streams in particular (Klikowski & Woźniak 2020).

In Section 9.1, we give a brief overview of existing work on one-class ensembles. Next, in Section 9.2, we spell out the formal connections between fuzzy rough sets and one-class classification and propose weighted nearest neighbour distance as a data descriptor and the fuzzy rough one-class ensemble for classification. We then propose a series of experiments in Section 9.3 and report the results in Section 9.4, before

concluding in Section 9.5.

9.1 Related research

One-class ensembles for multi-class classification were first introduced independently by Goh et al (2005), Ban & Abe (2006) and Giacinto et al (2008). There have been proposals with specifically adapted data descriptors, including the SVM data descriptor (Tohmé & Lengellé 2011) and the Extreme Learning Machine data descriptor (Gautam et al 2016).

Initial research focused on combining the predictions from the individual classifiers in an optimal way (Abbas et al 2013; Wilk & Wozniak 2010, 2012), but subsequently various authors investigated heterogenous ensembles, i.e. ensembles that combine multiple different data descriptors for each class to improve classification performance (Hadjadji et al 2014, 2017, 2019; Kang et al 2015; Krawczyk & Woźniak 2012, 2014a). It has also been shown that it may be worthwhile to decompose classes into clusters using an unsupervised algorithm before applying one-class ensembles (Abdallah et al 2021; Fragoso et al 2021; Krawczyk et al 2014c; Krawczyk & Cyganek 2017), and to exclude non-competent data descriptors from an ensemble (Krawczyk et al 2018).

Finally, data descriptors in a one-class ensemble can also be subjected to boosting (Xing & WT Liu 2020; CY Yeh et al 2009).

We will not evaluate any of these ideas at present, but note that they may be combined with the proposal that we present next.

9.2 Proposal

In this section, we show how the traditional upper and lower approximations can be seen as models of a data descriptor, and give a formal definition of generalised upper and lower approximations substituing other data descriptors.

Weighted nearest neighbour distance

Recall from Chapter 1 that the upper approximation of a subset C of a dataset $X \subset \mathbb{R}^m$ is the fuzzy set in $\mathbb{R}^m$ defined by:

$$\overline{C}(y) := \max(0, 1 - \overline{w} \min_{x \in C} d(y, x)), \tag{9.1}$$

for a choice of $\overline{w}$ and d. Note that this definition does not depend on X, and that the upper approximation therefore satisfies the formal definition of a data descriptor model of C, since it generalises C to a function from $\mathbb{R}^m$. The upper approximation also satisfies the purpose of a data descriptor, as it expresses similarity to the training set C.

Similarly, recall the definition of the lower approximation of C:

$$\underline{C}(y) := \min(\underline{w} \min_{x \in X \setminus C} (d(y, x)), 1). \tag{9.2}$$

At first glance, it may appear that, unlike the upper approximation, the lower approximation is dependent on both C and X. In fact, it is only dependent on $X \setminus C$, and therefore, this satisfies the definition of a data descriptor model of $X \setminus C$. However, in this case, the lower approximation does not satisfy the concept of a data descriptor, since it expresses *dissimilarity* with $X \setminus C$. Instead, we can rewrite it as the negation $z \longmapsto 1 - z$ of the upper approximation of $X \setminus C$, and therefore, interpret it as the negation of a data descriptor model:

$$\begin{aligned} \underline{C}(y) &= \min(\underline{w} \min_{x \in X \setminus C} (d(y, x)), 1) \\ &= 1 - \max(0, 1 - \underline{w} \min_{x \in X \setminus C} d(y, x)). \end{aligned} \tag{9.3}$$

We formalise the correspondence by giving a name to this data descriptor:

9.1 Definition (Weighted nearest neighbour distance). Let $X \subset \mathbb{R}^m$ be a dataset. For a choice of dissimilarity d and weight vector w, the *weighted nearest neighbour distance* (WNND) data descriptor sends X to the model defined by:

$$y \longmapsto \max(0, 1 - w \min_{x \in X} d(y, x)). \tag{9.4}$$

As with upper and lower approximations, we will assume linearly descending weights by default, reducing this hyperparameter to its length k.

Generalised upper and lower approximations

Having established that the traditional definition of upper and lower approximations can be seen as models of a specific data descriptor (WNND), we now propose a more general definition of upper and lower approximations where we allow the use of any data descriptor. This is possible because any data descriptor model is a function from $\mathbb{R}^m$ to $[0, 1]$, and thus, by definition, a fuzzy set in $\mathbb{R}^m$.

9.2 Definition. Let $X \subseteq \mathbb{R}^m$ be a dataset, let C be a crisp set in X, and let D be a choice of data descriptor. Then the upper and lower approximations of C induced by D are the fuzzy subsets of $\mathbb{R}^m$ defined as follows:

$$\begin{aligned} \overline{C}(y) &= D(C)(y); \\ \underline{C}(y) &= 1 - D(X \setminus C)(y). \end{aligned} \tag{9.5}$$

For the upper approximation, we can optimise the hyperparameters of D using $X \setminus C$, while for the lower approximation we can use C, with AUROC as the maximisation target as in Chapter 8.

Fuzzy rough one-class ensembles

We can now use the generalised upper and lower approximations to propose a generalised version of FRNN classification, the fuzzy rough one-class ensemble.

9.3 Definition (Fuzzy rough one-class ensemble). Let $X = \bigsqcup_{i \leq c} C_i \subseteq \mathbb{R}^m$ be a classification dataset, let D be a choice of data descriptor, and let $\lambda_1, \lambda_2, \ldots, \lambda_c \in [0, 1]$. Then for any $y \in \mathbb{R}^m$ and any class C_i, the *fuzzy rough one-class ensemble* induced by D assigns the following score:

$$(1 - \lambda_i) \cdot D(C_i)(y) + \lambda_i \cdot (1 - D(X \setminus C_i). \tag{9.6}$$

If $\lambda_1 = \lambda_2 = \ldots = \lambda_c$ are equal to 0, 1, or 0.5, we respectively recover the upper, lower, and mean approximation classifiers from Definition 1.6. However, what we propose is that subsequently to optimising the hyperparameters of D using C_i and $X \setminus C_i$ by maximising the AUROC of the corresponding one-class classification problems, one may then use the validation scores of the records in C_i and $X \setminus C_i$ to also optimise λ_i.

Furthermore, because we optimise the hyperparameters of D separately for each class, there is no guarantee that the resulting class scores are comparable. For instance, with WNND, larger values of k translate to larger distance values and smaller class scores. However, we do know that, everything else being equal, a higher class score indicates a higher probability that a record belongs to that class. Therefore, we post-process the class scores by training a logistic regression model (without regularisation) on the validation scores, and applying it to the predicted test scores.

While any data descriptor may be used to construct a fuzzy rough one-class ensemble, nearest neighbour data descriptors like NND, ALP and WNND that lend themselves to efficient leave-one-out validation (Section 8.1) are particularly suitable.

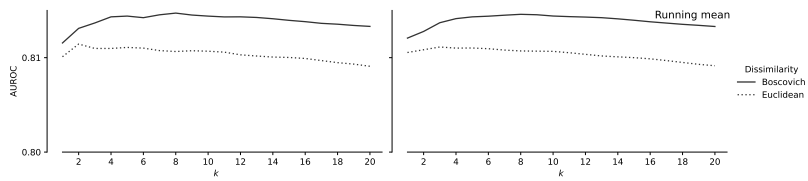


Figure 9.1: Weighted mean AUROC of WNND.

9.3 Experimental setup

We perform two sets of experiments in this chapter. Firstly, we will evaluate the performance of WNND as a data descriptor, replicating the experiments in the previous two chapters (see those chapters for more details).

Secondly, we will evaluate the performance of fuzzy rough one-class ensembles based on NND, WNND and ALP for multiclass classification, adopting the same experimental setup as in Chapter 3, restricted to datasets with more than two classes. In particular, we will use Boscovich distance and r_1 scaling, which yielded the best result for FRNN classification. We will first evaluate the effect of post-processing with logistic regression, and then compare the one-class ensembles with optimised FRNN classification.

9.4 Results

We now discuss the results of our experiments, starting with the one-class classification performance of WNND.

Weighted nearest neighbour distance

We start by considering the weighted mean AUROC of WNND as a function of k, weighting each one-class classification problem inversely proportional to the number of problems derived from the same original multiclass dataset. As with NND, we obtain a higher weighted mean AUROC when comparing k as-is, and not by reparametrising it in terms of the target class size. The result is displayed in Figure 9.1. As with NND, LNND and LOF, we obtain higher scores with Boscovich than with Euclidean distance. The weighted mean AUROC reaches its maximum at $k = 8$, which we therefore recommend as the default for WNND.

When we recalculate the optimal value for k for each dataset using the leave-one-dataset-out scheme, and compare the resulting AUROC to the other data descriptors, we obtain the p-values and weighted

Table 9.1: p: one-sided p-value of clustered Wilcoxon signed-rank test of WNND vs the data descriptor; p^*: p with Holm-Bonferroni family-wise error correction; τ: weighted Kendall's τ.

Vs	p	p^*	τ
ALP	0.99	≥ 1	0.83
SVM	0.55	≥ 1	0.87
NND	0.099	0.31	0.91
LOF	0.076	0.31	0.83
MD	0.018	0.091	0.75
IF	0.0066	0.040	0.66
EIF	< 0.0001	0.00036	0.75
SAE	< 0.0001	< 0.0001	0.67
LNND	< 0.0001	< 0.0001	0.79

Table 9.2: Weighted mean rank, AUROC and standard deviation of AUROC across cross-validation folds (CVSD) of ALP, SVM, WNND and NND.

Data descriptor	Rank	AUROC	CVSD
ALP	3.91	0.817	0.0378
SVM	4.40	0.814	0.0404
WNND	4.53	0.811	0.0425
NND	4.88	0.805	0.0414

Kendall's τ listed in Table 9.1. WNND performs comparable to SVM and somewhat better than NND on our selection of one-class classification problems, but the difference is not significant. Overall, the performance vis-à-vis the other data descriptors is somewhat better than that of NND vis-à-vis the other data descriptors (Table 7.4). The values for the weighted Kendall's τ indicate that, unsurprisingly, the performance of WNND aligns closest to NND, followed by SVM. Otherwise, these values are very similar to those between NND and SVM and the other data descriptors (Table 7.3).

In terms of weighted mean rank and weighted mean AUROC (Table 9.2), WNND also scores better than NND and approaches SVM. Somewhat surprisingly, it shows a bit more variation across cross-validation folds than NND. Overall, the ranks achieved by WNND are more consistently good than those achieved by NND (Figure 9.2).

As with NND, median AUROC is a positive significant factor for WNND, meaning that it performs better with easier one-class classification problems. This is illustrated by Figure 9.3, where we see that

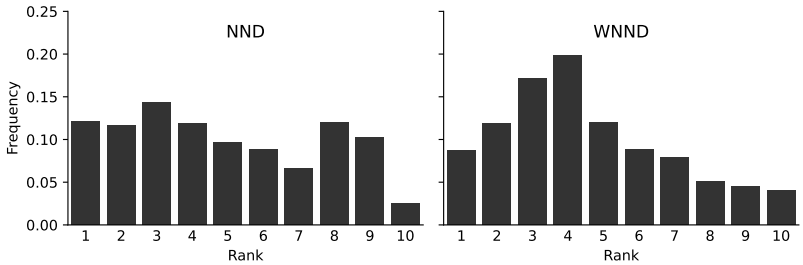


Figure 9.2: Average frequency (y-axis) of ranks (x-axis) of NND and WNND, weighted by dataset, with ties distributed evenly among the respective ranks.

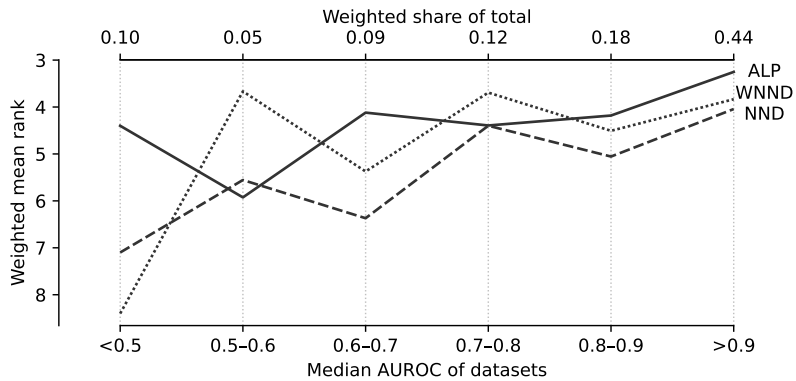


Figure 9.3: Weighted mean rank (y-axis) of ALP, NND and WNND, stratified by the median AUROC of datasets across all data descriptors (x-axis).

in each stratum except the group of essentially impossible problems where median AUROC is lower than 0.5, the performance of WNND is consistently slightly better than that of NND, and in some strata even rivals that of ALP.

Surprisingly, and unlike the other data descriptors, dimensionality is a negative significant factor for WNND ($p = 0.033$). We do not have a good explanation for this.

When we optimise k using negative data, we find that the relative advantage of WNND over NND and LOF quickly disappears (Figure 9.4), and SVM and ALP clearly outperform WNND after a handful of evaluations. Looking at the weighted mean AUROC across one-class classification problems (Figure 9.5), we see that the test set performance

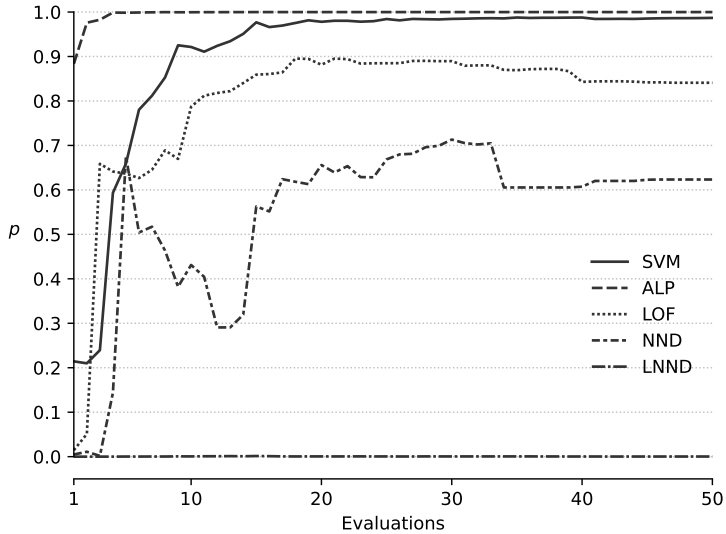


Figure 9.4: One-sided p-values of clustered Wilcoxon signed rank tests that WNND is better than other data descriptors (uncorrected for multiple testing) with hyperparameter optimisation, as a function of the number of evaluations.

Table 9.3: One-sided p-values, evaluating the effect of post-processing with logistic regression for fuzzy rough one-class ensembles of upper approximations, lower approximations, and optimised ratios of both.

descriptor	Upper	Lower	Both
ALP	0.00094	1.0e-05	0.035
NND	0.15	1.4e-05	0.0065
WNND	0.22	1.3e-05	0.011

of both WNND and NND quickly reaches a plateau, but that for the first few evaluations, NND simply improves more with each evaluation. It may be the case that WNND has a slightly better default performance than NND because the use of weights gives it greater stability, but that they also reduce the precision with which it can be adapted to a specific problem.

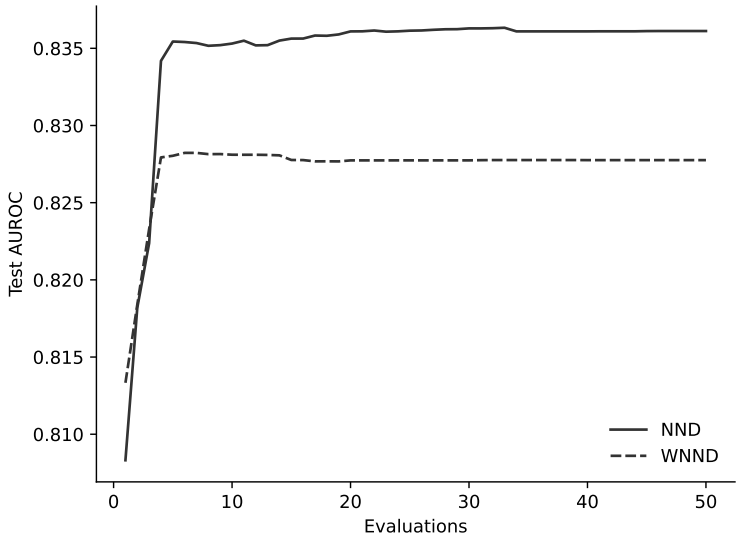


Figure 9.5: Weighted mean test AUROC of NND and WNND.

Table 9.4: One-sided p-values, comparing fuzzy rough one-class ensembles with logistic regression post-processing against optimised classical FRNN.

descriptor	Upper	Lower	Both
ALP	1.3e-05	0.99	0.86
NND	7.0e-05	1.0	0.98
WNND	1.8e-05	1.0	0.96

Table 9.5: One-sided p-values, comparing fuzzy rough one-class ensembles with optimised ratios of upper and lower approximations against just either type (with logistic regression post-processing).

strategy descriptor	Upper	Lower
ALP	0.45	0.0015
NND	0.091	8.1e-05
WNND	0.48	0.0013

Table 9.6: Multiclass AUROC. ALP: fuzzy rough one-class ensemble induced by ALP, with optimised upper and lower approximation ratio; IR: imbalance ratio.

dataset	n	c	IR	ALP	FRNN	Difference
shuttle	58000	7	560.8	0.997686	0.990	0.0081
vehicle	846	4	1.1	0.921674	0.914	0.0073
new-thyroid	215	3	3.5	0.990926	0.985	0.0057
foresttypes	523	4	1.8	0.971938	0.967	0.0048
column	310	3	1.9	0.920194	0.916	0.004
iris	150	3	1.0	0.998667	0.997	0.0013
avila	20867	12	38.7	0.997020	0.996	0.0013
faults	1941	7	3.9	0.961932	0.962	0.00037
letter	20000	26	1.0	0.999443	0.999	0.00016
page-blocks	5473	5	31.6	0.972923	0.973	0.00013
wifi	2000	4	1.0	0.999605	0.999	0.00013
waveform	5000	3	1.0	0.970540	0.971	-2.4e-05
texture	5500	11	1.0	0.999585	1.000	-0.00014
segment	2310	7	1.0	0.996499	0.997	-0.00047
dermatology	358	6	2.2	0.997808	0.999	-0.0014
sensorless	58509	11	1.0	0.997748	1.000	-0.0018
mfeat	2000	10	1.0	0.997640	1.000	-0.0019
landsat	6435	6	1.7	0.986523	0.989	-0.0021
wine	178	3	1.3	0.997294	1.000	-0.0027
seeds	210	3	1.0	0.984524	0.987	-0.0027
yeast	1484	10	11.6	0.874931	0.886	-0.011
ecoli	332	6	6.3	0.947055	0.961	-0.014
breasttissue	106	6	1.3	0.906306	0.925	-0.019
leaf	340	30	1.2	0.950147	0.975	-0.025
glass	214	6	3.6	0.884746	0.919	-0.034

Fuzzy rough one-class ensemble

We now evaluate the performance of fuzzy rough one-class ensembles. As can be seen in Table 9.3, post-processing with logistic regression improves performance, although the effect is only weakly significant for upper approximation ensembles induced by the NND and WNND data descriptors.

Next, we compare the resulting performance with classical FRNN. In particular, we compare upper approximation one-class ensembles against the upper approximation classifier with optimised k, lower approximation one-class ensembles against the lower approximation classifier with optimised k, and one-class ensembles with optimised ratios against FRNN as optimised in Chapter 3. We find (Table 9.4) that fuzzy rough one-class ensembles only perform better with upper approximations, whereas in the other two cases they perform significantly worse.

Fuzzy rough one-class ensembles appear to perform particularly

poorly with lower approximations, so much so that the optimised ratios actually do not perform better than simply using upper approximations for ALP and WNND (Table 9.5). (Note, however, that upper approximation ensembles do not perform any better against optimised FRNN.) It is possible that data descriptors are simply not a good fit for negative data, which is composed of several different decision classes.

If we compare the results for the ensemble induced by ALP and FRNN in detail (Table 9.6), we can make a few observations. Firstly, the overall poorer result by the ALP ensemble appears to be due to the fact that it performs relatively weakly on a handful of datasets, rather than that it performs worse on a large majority of datasets. Secondly, the datasets on which it performs worst are relatively small. Thirdly, it performs well on a number of datasets with a large imbalance ratio. And lastly, the ALP ensemble appears to perform relatively poorly when there are many classes. This is slightly counterintuitive, but we can think of two possible explanations. Firstly, when there are many classes, the logistic regression post-processing task to balance the class scores is more difficult. And secondly, when there are many classes, their complements that serve as the target for the lower approximation are more alike, making it more likely that a single globally optimised hyperparameter value is better than class-specific hyperparameter values that were optimised through one-class classification.

When we fit linear regression models using ordinary least squares, to predict the signed rank of each difference, we find that the imbalance ratio is not in fact significant when we also take the dataset size n into account. Instead, we obtain a good fit by including $\log n$ ($p = 0.017$) and $\log c$ ($p = 0.012$). The resulting model predicts that the ALP ensemble has a higher AUROC than FRNN when the following condition is satisfied:[1]

$$4.03 \cdot \log n - 12.6 \cdot \log c - 8.45 > 0. \tag{9.7}$$

9.5 Conclusion

In this chapter, we have investigated the connections between fuzzy rough set theory and one-class classification. We have shown that the

[1]We have also identified a small shortcoming in our implementation of ALP. Recall (Section 8.1) that the hyperparameter l controls the number of neighbours used to calculate, as a weighted average, the local kth nearest neighbour distance in the target data, as well as the slope of the weight vector. As l increases, this weighted average should approach the unweighted mean in the whole target class, but for practical reasons, we limit the calculation of the number of neighbours of a test record to $20 \log n$, where n is the target class size. We have now found that one obtains slightly better results by making the asymptotic limit available to the optimisation process by substituting the target class mean when $l > 20 \log n$. The effect of this on our analysis is small, but the significance of $\log n$ ($p = 0.004$) and $\log c$ ($p = 0.005$) in the linear regression model becomes even larger.

classical definitions of upper and lower approximation can be interpreted as involving a data descriptor, WNND, which we subjected to the same analysis as other data descriptor in the previous two chapters. We found that when used in the context of default hyperparameter values — we recommend $k = 8$ — it can be an attractive alternative to NND, although we found no definitive evidence that it performs better, and its advantage disappears in the context of hyperparameter optimisation.

In addition, we have proposed a generalised definition of upper and lower approximations, allowing the application of data descriptors other than WNND, and leading to the concept of a fuzzy rough one-class ensemble for classification. This is distinguished from standard one-class ensembles by its inclusion of the lower approximations of the decision classes, determined by their complements. The fact that the good performance of FRNN classification is to a large part due to the lower approximation — as we saw in Chapter 3 — indicates that this is of great potential benefit to one-class ensembles. Unfortunately, we failed to obtain good overall performance by optimising these lower approximations as one-class classification tasks.

After subjecting our results to closer analysis, we found that a fuzzy rough one-class ensemble induced by ALP does on the whole lead to higher multiclass classification performance than FRNN on large datasets and datasets with not too many classes. We think that the latter fact can be explained by the fact that as the number of decision classes increases, so does the overlap between their complements, making it more likely that a single hyperparameter value obtained through global optimisation is close to optimal for each class.

For the future, we think that it is important to gain a better understanding of why exactly hyperparameter optimisation through one-class classification works better for upper approximations than for lower approximations. If this issue turns out to be unsolvable, it may be worthwhile to try combining individually optimised upper approximations with globally optimised lower approximations.

Part IV

Incomplete datasets

Chapter 10

Missing-indicators[1]

In this and the next two chapters, we will consider three different ways of representing missing values in datasets.

Missing values are a frequent issue in real-life datasets, and the subject of a large body of ongoing research. Some implementations of machine learning algorithms can handle missing values natively, requiring no further action by practitioners. But whenever this is not the case, a common general strategy is to replace the missing value with an estimated value — *imputation*. An advantage of imputation is that we obtain a complete dataset, to which we can apply any and all algorithms that make no special provision for missing values. However, missing values may be informative, and a disadvantage of imputation is that it removes this information.

In the present chapter, we will review and evaluate the missing-indicator approach (J Cohen 1968), which is an old proposal to represent and thereby preserve the information encoded by missing values. For every original attribute, it adds a new binary 'indicator' or 'dummy' attribute that takes a value of 1 if the value for the original attribute is missing, and 0 if not.[2] The missing-indicator approach is often presented as an alternative to imputation, but since it does not resolve the missing values in the original attributes, it can only be used in addition to, not instead of imputation.

Both imputation and the missing-indicator approach originate in the statistical literature. While imputation strategies have been the subject of a rich body of research, the missing-indicator approach has not received a large amount of attention, and is often dismissed or disregarded in overviews of approaches towards missing values. In particular, it is an open question whether missing-indicators should be

[1] This chapter is based on Lenz et al (2022b).

[2] Some authors use the opposite convention, letting the indicator express non-missingness.

used for predictive tasks in machine learning (Sperrin et al 2020). On the one hand, the addition of missing-indicators results in a more complete, higher-dimensional representation of the data. On the other hand, their omission can be seen as a form of dimensionality reduction, which may increase the efficiency and effectiveness of a dataset by eliminating redundancy.

To determine whether this trade-off is useful, a key question is to which extent missing values in a given dataset are informative. If they are not, the phrase "missing at random" (MAR) (Rubin 1976) is used to indicate that the distribution of missing values is dependent on the known values, while the stricter phrase "missing completely at random" (MCAR) denotes values that are distributed truly randomly. In contrast, informative missing values are often denoted as "missing not at random" (MNAR). For real-life datasets, unless we have specific knowledge about the process responsible for the missing values, we have to assume some degree of informativeness in principle.[3] However, Schafer (1997) has argued that in practice, the attributes of a dataset can be sufficiently redundant that one can get away with assuming its missing values are MAR. But even if this is so, imputation may not always perform optimally, in which case missing-indicators may still prove useful.

A more subtle point is that even when missing values are informative, the information they encode need not be lost completely through imputation. This is particularly evident in the case of numerically encoded binary attributes, where imputation can represent missing values as a third, intermediary value. More generally, Le Morvan et al (2021) have observed that almost all deterministic imputation functions map records with missing values to distinct manifolds in the attribute space that can in principle be identified by sufficiently powerful algorithms. Nevertheless, including missing-indicators can still potentially make this learning task easier.

In light of these conflicting theoretical arguments, the usefulness of missing-indicators for real-life machine learning problems is an interesting empirical question. However, previous experiments in this direction have been limited in scope and number. These limitations include the use of only one or a handful of datasets, the use of datasets from which values have been removed artificially, at random (corresponding to the MCAR setting), and not comparing the same imputation strategies with and without missing-indicators.

In this chapter, we will evaluate the effect of missing-indicators on the performance of a range of popular classification algorithms, paired

[3] This is acknowledged by authors working under the assumption of MAR, e.g. "When data are missing for reasons beyond the investigator's control, one can never be certain whether MAR holds. The MAR hypothesis in such datasets cannot be formally tested unless the missing values, or at least a sample of them, are available from an external source." (Schafer 1997)

with three common types of imputation, on the basis of twenty real-life classification problems with naturally occurring missing values.

In Section 10.1, we provide a brief overview of the existing literature on missing-indicators, including previous experimental evaluations. In Section 10.2, we describe our experimental setup. We report our results in Section 10.3 and conclude in Section 10.4.

10.1 Background

We start with a brief discussion of the origins and reception of the missing-indicator approach, as well as previous experimental evaluations of the use of missing-indicators in prediction tasks.

Origins and reception

The missing-indicator approach originates in the statistical literature on linear regression. It dates back to at least J Cohen (1968), who pointed out that values in real-life datasets are typically not missing completely at random, and that the distribution of missing values may in particular depend on the values of the attribute that is to be predicted. He proposed that each attribute could be said to have two 'aspects', its value, and whether that value is present to begin with, which should be encoded with a pair of variables. For missing attribute values, the first of these variables was to be filled in with the mean of the known values, although other applications might call for different values. Cohen's proposal was subsequently expanded by J Cohen & P Cohen (1975), but received only limited recognition in the following years (AB Anderson et al 1983; Chow 1979; Hutcheson & Prather 1981; JO Kim & Curry 1977; Orme & Reis 1991; Stumpf 1978).

Cohen's proposal was subjected to a formal analysis by MP Jones (1996), who showed that, if one assumes that missing values are MAR, and the true linear regression model does not contain any terms related to missingness, it produces biased estimates of the regression coefficients (unless the sample covariance between independent variables is zero). However, these assumptions run directly counter to the position set out by J Cohen & P Cohen (1975) that a priori, the missingness of each attribute is a possible explanatory factor, that it is safer not to assume that missing values are distributed randomly, and that the usefulness of missing-indicators is ultimately an empirical question.

Allison (2001), motivated by MP Jones (1996) and working under the general assumption of MAR, dismissed missing-indicators as "clearly unacceptable", before conceding that they in fact produce optimal estimates when the missing value is not just missing, but cannot exist, such as the marital quality of an unmarried couple. However, this

semantic distinction may not always be clear-cut in practice, and the more pertinent question may be whether missing values are informative. Allison (2010) later acknowledged that missing-indicators may lead to better predictions and their use for that purpose was acceptable. Missing-indicators have also been dismissed elsewhere (Aste et al 2015; Graham 2009; Pigott 2001; Schafer & Graham 2002), and are frequently omitted in overviews of missing data strategies (Das et al 2018; Eirola 2014; Enders 2010; García et al 2015; Schafer 1997).

Previous experiments

Only a handful of experimental comparisons of missing data approaches have included the missing-indicator approach, and these have been limited in scope. Vamplew & Adams (1992) and CG Ng & Yusoff (2011) only use a single dataset with randomly removed values, and base their evaluation on the performance of a single algorithm (respectively a neural network and linear regression). Pereira Barata et al (2019) use three classification algorithms and 22 datasets, but again with randomly removed values, explicitly assuming a MCAR context. They conclude that imputation outperforms missing-indicators, but the comparison is not like-for-like, since it involves several forms of imputation but only combines indicator attributes with zero imputation. Van der Heijden et al (2006) compare missing-indicators with zero imputation against several other forms of imputation without missing-indicators on one real dataset, for logistic regression. However, it appears that they do not use a test set, and only evaluate the resulting models on the training set.

Ding & Simonoff (2010) conduct a more extensive investigation, using insights from a series of Monte Carlo simulations to systematically remove values from 36 datasets to simulate different forms of missingness. They use these datasets to compare zero imputation[4] with indicator attributes against mean/mode imputation without, as well as a number of other missing data approaches, for logistic regression. In addition, the authors evaluate a related representation of missing values[5] on the same set of 36 datasets, and on one real-life dataset with missing values, for decision trees. They find that there is strong evidence that representing missing values is the best approach when they are informative; when this is not the case their results show no strong difference with respect to imputation.

[4] Presumably, Ding & Simonoff (2010) use one-hot encoding for categorical attributes, in which case zero imputation is equivalent to treating missing values as a separate category, but they do not state this explicitly.

[5] For categorical values, encoding missing values as a separate category, for numerical values, encoding missing values as an extremely large value that can always be split from the other values.

The comparison by Grzymala-Busse & M Hu (2000) is based on 10 datasets with naturally occurring missing values. However, the setting is purely categorical — all attributes are transformed into categorical attributes — the only form of imputation is mode imputation, and the missing value approaches are evaluated on the basis of the LERS classifier (Learning from Examples based on Rough Sets (Grzymala-Busse 1988)).

Marlin (2008) compares zero imputation with missing-indicators (*augmentation with response indicators*) against several forms of imputation without missing-indicators, for logistic regression and neural networks, on the basis of an extensive series of simulations, one dataset with artificially removed values, and three real datasets. For the real datasets, there is no strong difference in performance between the different approaches.

Most recently, Josse et al (2020) and Le Morvan et al (2021) respectively evaluate missing-indicators (*missingness mask*) for regression trees, Random Forest and XGBoost, and for multilayer perceptrons, on simulated regression datasets, and conclude that when missing values are informative, using missing-indicators clearly increases performance. This work has been continued by Perez-Lebel et al (2022), who compare four different imputation techniques with and without missing-indicators on seven prediction tasks derived from four real medical datasets, and conclude that missing-indicators consistently improve performance for gradient boosted trees, ridge regression and logistic regression.

We point out that the Missingness in Attribute (MIA) proposal (BE Twala et al 2008) for decision trees and decision tree ensembles can be understood as an implicit combination of missing-indicators with automatic imputation, and has also been shown to outperform imputation without missing-indicators in small-scale experimental studies (Josse et al 2020; Perez-Lebel et al 2022).

Finally, even experimental comparisons of missing data that do not feature the missing-indicator approach generally do not involve more than a handful of real-life datasets with naturally occurring missing values. We have only found Luengo et al (2012a,b), who use 21 datasets from the UCI repository, but 12 of these are problematic.[6]

[6]The target column of the *echocardiogram* dataset ('alive-at-1') is supposed to denote whether a patient survived for at least one year, but it doesn't appear to agree with the columns from which it is derived, that denote how long a patient (has) survived and whether they were alive at the end of that period. The *audiology* dataset has a large number of small classes with complex labels and should perhaps be analysed with multi-label classification. In addition, it has ordinal attributes where the order of the values is not entirely clear, and three different values that potentially denote missingness ('?', 'unmeasured' and 'absent'), and it is not completely clear how they relate to each other. The *house-votes-84* dataset contains '?' values, but its documentation explicitly states that these values are not unknown, but indicate different forms of abstention. The *ozone* dataset is a time-series problem, while the task associated with the *sponge* and *water-treatment* datasets is clustering, with no obvious target for classification among their

10.2 Experimental setup

To evaluate the effect of the missing-indicator approach on classification performance, we conduct a series of experiments, using the Python machine learning library scikit-learn (Pedregosa et al 2011).[7]

Questions

The aim of our experiments is to answer the following questions:

- Do missing-indicators increase performance, and does it matter which imputation strategy they are paired with?
- When do missing-indicators start to become useful in terms of the number of missing values?
- Does using mean imputation instead of mode imputation allow for more information to be learned from missing categorical values?

The last question is motivated by the observation that mean imputation would keep missing categorical values distinct from non-missing values, making the information that they encode in principle easier to recover.

Evaluation

We preprocess datasets by standardising numerical attributes and one-hot encoding categorical attributes (as required by the implementations in scikit-learn).

We measure classification performance by performing stratified five-fold cross-validation, repeating this for five different random states (which determine both the dataset splits and the initialisation of algorithms with a random component), and calculating the mean AUROC. To compare two alternatives A and B, we consider the p-value of a one-sided Wilcoxon signed-rank test on the mean AUROC scores for our selection of datasets.

Imputation Strategies

We consider the following three imputation strategies:

respective attributes. Finally, the *breast-cancer* (9), *cleveland* (7), *dermatology* (8), *lung-cancer* (5), *post-operative* (3) and *wisconsin* (16) datasets contain only very few missing values, and any performance difference between missing value approaches on these datasets may to a large extent be coincidental.

[7] The code to reproduce the experiments in this chapter is available at `https://cwi.ugent.be/~oulenz/code/lenz-2022-no-imputation.tar.gz`.

Table 10.1: Classification algorithms.

Name	Description
NN-1	Nearest neighbours (Fix & Hodges 1951) with Boscovich distance
NN-2	Nearest neighbours with Euclidean distance
NN-1-D	Nearest neighbours with Boscovich distance, distance-weighted (Dudani 1976)
NN-2-D	Nearest neighbours with Euclidean distance, distance-weighted
SVM-L	Soft-margin Support Vector Machine (Cortes & Vapnik 1995) with linear kernel
SVM-G	Soft-margin Support Vector Machine with Gaussian kernel
LR	Multinomial logistic regression (Cox 1966)
MLP	Multilayer perceptron (Rosenblatt 1961) with ReLu activation (Fukushima 1969), Glorot initialisation (Glorot & Bengio 2010) and Adam optimisation (Kingma & Ba 2015)
CART	Classification and Regression Tree (Breiman et al 1984)
RF	Random Forest (Breiman 2001)
ERT	Extremely Randomised Trees (Geurts et al 2006)
ABT	Ada-boosted trees (Freund & Schapire 1995) with SAMME (stagewise additive modeling using a multi-class exponential loss function) (Zhu et al 2009)
GBM	Gradient Boosting Machine (Friedman 2001)

- *Mean/mode imputation* replaces missing values of numerical and categorical attributes by, respectively, the mean and the mode of the non-missing values.

- *Nearest neighbour imputation* (Troyanskaya et al 2001) replaces missing values of numerical and categorical attributes by, respectively, the mean and the mode of the 5 nearest non-missing values, with distance determined by the corresponding non-missing values for the other attributes.

- *Iterative imputation*, as implemented in scikit-learn, based on Van Buuren & Groothuis-Oudshoorn (2011), predicts missing values of one attribute on the basis of the other attribute values using a round-robin approach. For numerical attributes, this uses Bayesian ridge regression (Tipping 2001), initialised with mean imputation, while for categorical attributes, we use logistic regression, initialised with mode imputation.

The scikit-learn implementations of nearest neighbour and iterative imputation can currently only impute numerical features, so we had to adapt them for categorical imputation. In all other aspects, we follow the default settings of scikit-learn.[8]

[8]For the *nomao* dataset, iterative imputation diverged (i.e. the imputed values would grow towards infinity), so we forced imputed values to stay in the interval $[-100, 100]$.

Classification Algorithms

We consider the classification algorithms listed in Table 10.1, as implemented in scikit-learn. Hyperparameters take their default values, except for SVM-L, LR and MLP, where we increase the maximum number of iterations of the optimisation algorithm to 10 000 to increase the probability of convergence.

For a number of these algorithms, specific ways have been proposed to handle missing values: e.g. NN-2-D (Dixon 1979), SVM-G (Śmieja et al 2019), MLP (Ipsen et al 2020; Śmieja et al 2018; Tresp et al 1994) and CART (Quinlan 1989; BE Twala et al 2008). We do not consider these here, as the purpose of the present experiment is to evaluate the general approach of using imputation with missing-indicators that can be applied in contexts where these solutions have not been implemented, as is the case in scikit-learn.

Datasets

We use twenty real-life datasets with naturally occurring missing values. These are described in Section B.3.

10.3 Results and discussion

Using the experimental setup detailed in the previous section, we now try to answer the questions listed in Subsection 10.2. For the AUROC scores, see Section C.2 in the Appendix.

Do missing-indicators increase performance, and does it matter which imputation strategy they are paired with?

Missing-indicators generally lead to increased performance (Table 10.2) — with the notable exception of CART. The more complicated imputation strategies do not result in much better results than mean/mode imputation when we pair imputation with missing-indicators (Table 10.3). At best, nearest neighbour and iterative imputation only lead to a modest improvement, and for many classifiers, they actually decrease performance. Therefore, we focus on mean/mode imputation for the remainder of this section.

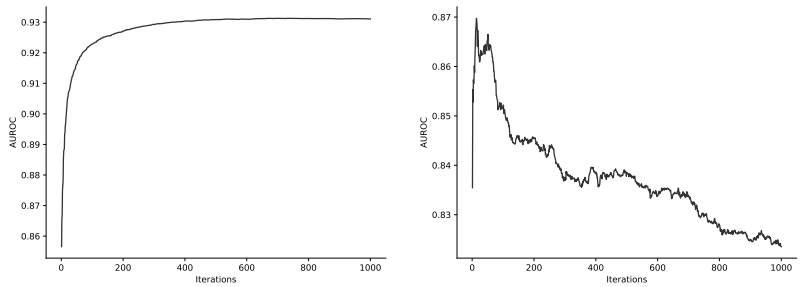
A possible reason for the poor performance of missing-indicators with CART, is that by default, the scikit-learn implementation of this classifier does not perform pruning, making it prone to overfitting. To test this hypothesis, we repeat our experiment for CART and mean imputation, but this time we apply cost complexity pruning ($\alpha = 0.01$). The resulting AUROC scores are now much better than the original

Table 10.2: One-sided p-values obtained by comparing AUROC from imputation with missing-indicators vs without.

Classifier	Imputation strategy		
	Mean/mode	Neighbours	Iterative
NN-1	0.024	0.0027	0.0011
NN-2	0.035	0.0050	0.00085
NN-1-D	0.016	0.0031	0.00085
NN-2-D	0.0063	0.0070	0.00042
SVM-L	0.18	0.31	0.11
SVM-G	0.0063	0.0063	0.0027
LR	0.092	0.079	0.074
MLP	0.0050	0.013	0.011
CART	0.84	0.75	0.70
RF	0.058	0.12	0.29
ERT	0.36	0.018	0.027
ABT	0.089	0.10	0.49
GBM	0.39	0.022	0.18

Table 10.3: One-sided p-values obtained by comparing AUROC from missing-indicators with iterative and nearest neighbour vs mean/mode imputation.

Classifier	Imputation strategy	
	Neighbours	Iterative
NN-1	0.90	0.27
NN-2	0.74	0.26
NN-1-D	0.95	0.71
NN-2-D	0.80	0.34
SVM-L	0.48	0.61
SVM-G	0.47	0.94
LR	0.36	0.85
MLP	0.29	0.56
CART	0.67	0.69
RF	0.63	0.86
ERT	0.47	0.51
ABT	0.63	0.94
GBM	0.94	0.83



(a) adult (b) mammographic-masses

Figure 10.1: Test AUROC from one cross-validation fold for GBM as a function of the number of steps of gradient descent, for two illustrative datasets. Using a constant number of iterations can lead to underfitting (a) or overfitting (b).

scores without missing-indicators ($p = 0.013$) and somewhat better than cost complexity pruning without missing-indicators ($p = 0.23$).

In the case of ERT, missing-indicators may not lead to a clear performance improvement because of underfitting. If we increase the number of trees from the default 100 to 1000, the improvement becomes somewhat clearer ($p = 0.15$).

For GBM, the default choice of 100 iterations of gradient descent can lead to both under- or overfitting, depending on the dataset (Figure 10.1). We believe that it is generally preferable to continue training until an early-stopping criterion is met. If we apply the same criterion as the scikit-learn default for MLP,[9] the performance increase due to missing-indicators also becomes clearer ($p = 0.19$).

When do missing-indicators start to become useful in terms of the number of missing values?

The theoretical motivation for representing missing values through missing-indicators is that this allows classifiers to learn the information encoded in their distribution. In principle, this should be easier when there are more examples to learn from. We can use this to obtain a better understanding of when missing-indicators might be useful on a per-attribute level.

We test this with the following additional experiment. For each attribute with missing values in each dataset, we reduce the original

[9] Setting aside 10% of the data for validation, stopping when validation loss has not decreased by at least 0.0001 for ten iterations, with a maximum of 10 000 iterations.

Table 10.4: Thresholds above which missing-indicators are more likely than not to increase AUROC, in terms of the absolute number of missing values or the missing value rate

Classifier	Missing values		Missing value rate	
	Categorical	Numerical	Categorical	Numerical
NN-1	20	296		
NN-2	10	146		
NN-1-D	21	370		
NN-2-D	6	80		
SVM-L			0.0	0.0
SVM-G			0.0	0.44
LR			0.0	0.0
CART	1	13		
ERT			0.0	1.0
ABT	1	18850		
GBM			0.0	0.0

dataset by removing all other attributes with missing values. We thus obtain 1148 derived datasets, on which we again apply each of our classifiers (with pruning for CART, 1000 trees for ERT and early-stopping for GBM) and consider whether missing-indicators increase or decrease AUROC (we dismiss ties). Finally, for each classifier we fit a logistic regression model with cluster robust covariance (clustered by the originating dataset), with the following potential parameters: categoricalness (whether the attribute is categorical) and either the number of missing values (log-transformed) or the missing value rate. We use the Akaike information criterion (Akaike 1971) to decide whether to select these parameters.

We find that for most classifiers, either the absolute or the relative number of missing values is an informative parameter with positive coefficient. For MLP, neither parameter is informative, while for RF, the number of missing values is an informative parameter with negative coefficient, for which we have no explanation at present. For every classifier except NN-1, NN-1-D and LR, categoricalness is an informative parameter with positive coefficient, meaning that missing-indicators are more beneficial for categorical than for numerical attributes.

The fitted logistic regression models allow us to calculate attribute-specific thresholds at and above which missing-indicators are more likely than not to increase AUROC, for all classifiers except MLP and RF (Table 10.4). In many cases, these thresholds are 1 or 0.0, indicating that missing-indicators are always likely to increase AUROC. We have included NN-1 and NN-1-D in this table on the basis of a model that includes the categoricalness parameter, since we find it implausible that it should not be relevant only for these specific classifiers. If we exclude it, the thresholds respectively become 233 and 285 records for categorical

and numerical attributes alike. For LR, the threshold is a missing value rate of 0.0, whether we include the categoricalness parameter or not.

Does using mean imputation instead of mode imputation allow for more information to be learned from missing categorical values?

As indicated above, missing-indicators are generally more likely to increase performance for categorical than for numerical attributes. A potential explanation for this is the fact that the mode of a categorical attribute is one of the non-missing values, whereas the mean of a numerical attribute is generally not equal to one of the non-missing values. Therefore, categorical imputation renders missing values truly indistinguishable from non-missing values, whereas numerical imputation does not — the information expressed by missing values may be partially recoverable, as argued by Le Morvan et al (2021) and discussed in the Introduction.

We can achieve a similar partial representation of missing categorical values by changing the order in which we perform imputation and one-hot encoding, i.e. by performing numerical imputation on one-hot encoded categorical attributes with missing values. For imputation without missing-indicators, this indeed leads to better performance for some classifiers, while in combination with missing-indicators, it does not make much of a difference (Table 10.5)[10].

10.4 Conclusion

In this chapter, we have presented the first large-scale experimental evaluation of the effect of the missing-indicator approach on classification performance, conducted on real datasets with naturally occurring missing values, paired with three different imputation techniques. The central question was whether, on balance, more benefit can be derived from the additional information encoded in a representation of missing values, or from the lower-dimensional projection of the data obtained by omitting missing-indicators.

On the whole, missing-indicators increase performance for the classification algorithms that we considered, although we cannot be sure for each classifier that this result will generalise to other datasets. The only classifier for which missing-indicators decreased performance was CART. We have argued that this is due to overfitting by the default configuration of the scikit-learn implementation of CART, and showed that

[10] LR is an exception here. We have no explanation for this, although we note that it corresponds with our finding in Subsection 10.3 that categoricalness is not an informative parameter for LR.

Table 10.5: One-sided p-values obtained by comparing AUROC from mean imputation after one-hot encoding vs mode imputation of missing categorical values.

Classifier	Without —	With missing-indicators
NN-1	0.030	0.22
NN-2	0.32	0.17
NN-1-D	0.030	0.36
NN-2-D	0.29	0.17
SVM-L	0.44	0.71
SVM-G	0.22	0.56
LR	0.88	0.023
MLP	0.14	0.52
CART	0.50	0.34
RF	0.084	0.78
ERT	0.023	0.95
ABT	0.56	0.66
GBM	0.12	0.56

missing-indicators do increase performance when pruning is applied. For ERT and GBM, we were able to show that the advantage of including missing-indicators becomes more significant when the number of trees of ERT is increased to limit underfitting, and the number of iterations of GBM is determined dynamically by an early-stopping criterion to avoid both under- and overfitting.

We also found that, in the presence of missing-indicators, nearest neighbour and iterative imputation do not increase performance over simple mean/mode imputation, with the possible exception of NN-2 and NN-2-D in the case of iterative imputation. This is a useful finding, because implementations of more sophisticated imputation strategies may not always be available to practitioners working in different frameworks, or easy to apply.

In a follow-up experiment, we determined attribute-specific missingness thresholds above which missing-indicators are more likely than not to increase performance. For categorical attributes, this threshold is generally very low, while for numerical attributes, there is more variation among classifiers, in particular as to whether this threshold is absolute or relative to the total number of records.

The greater usefulness of missing-indicators for categorical than for numerical attributes can be explained by the fact that the mean of a numerical attribute is not generally identical to any of the non-missing values, and that mean imputation therefore preserves some of the information of missing values. Accordingly, in the absence of missing-indicators, applying mean imputation to one-hot encoded categorical attributes results in somewhat better performance than mode imputation.

On the basis of these experiments, we conclude that the combination of mean/mode imputation with missing-indicators represents the best known general-purpose solution for missing values in a classification context, that may be used when the algorithm to be used has no special provision for missing values, and when values are not known to be missing at random. While over- or underfitting is a concern for certain classifiers, it is a concern for these classifiers with or without missing-indicators. The use of missing-indicators can also be combined with dimensionality reduction algorithms to increase the information density of the resulting dataset.

In the next chapter, we will consider a more specific proposal for representing missing information with fuzzy rough sets.

Chapter 11

Interval-valued fuzzy rough sets[1]

Recall from Chapter 1 that fuzzy rough sets are a combination of fuzzy and rough sets. Accordingly, they can be used to model two different types of uncertainty. As fuzzy sets, upper and lower approximations model partial membership of a concept C, while the difference between the upper and lower approximation captures the conflicting ways in which C may be predicted from a set of independent attributes: the upper approximation generalises the positive evidence for C, whereas the lower approximation generalises the negative evidence $X \setminus C$.

In this chapter, we consider a third type of uncertainty: incomplete information. There exist several proposals for missing data that involve rough or fuzzy rough sets (Thangavel & Pethalakshmi 2009). In particular, fuzzy rough sets have been used for imputation (Amiri & Jensen 2016), there have been proposals to adapt both crisp and fuzzy decision rules to the presence of missing values (Grzymala-Busse 2006; Hong et al 2010; Kryszkiewicz 1998), and in the context of classical rough sets, Grzymala-Busse (2006) has provided three alternative definitions of upper and lower approximations in datasets with missing values. In contrast, our strategy will be to incorporate the uncertainty of incomplete information directly into the representation of concepts, by extending the notion of upper and lower approximation.

We propose to mimic the dual approach of rough sets by modelling an optimistic and a pessimistic scenario when comparing a missing value with another value. The optimistic scenario is that the two values are really identical, while the pessimistic scenario is that they are maximally different. We cannot know what the ground truth is, but we know that it must lie somewhere in between these two extremes. Formally, we can represent this with an interval-valued fuzzy set (Dubois & Prade 2005,

[1]This chapter is based on Lenz et al (2021a).

and references therein). Since the uncertainty of incomplete information is orthogonal to the uncertainty that arises from positive and negative information, the resulting interval-valued fuzzy rough set is defined by four fuzzy sets: the optimistic and pessimistic upper and lower approximations $\overline{C}^{\min}, \overline{C}^{\max}, \underline{C}_{\min}, \underline{C}_{\max}$.

This work builds on the earlier proposal for interval-valued fuzzy rough sets in the context of feature selection by Jensen & Shen (2009), as well as a related proposal of *ill-known* fuzzy rough sets (Couso & Dubois 2011) based on twofold fuzzy sets (Dubois & Prade 1987), but this approach has otherwise remained relatively underexplored. We present an up-to-date definition in Section 11.1. In Section 11.2, we modify Fuzzy Rough Nearest Neighbour (FRNN) classification to incorporate interval-valued fuzzy rough sets, and evaluate its performance on a number of real-life datasets.

11.1 Interval-valued fuzzy rough sets

In order to formulate the application of interval-valued fuzzy rough sets to missing values, we have to use the definition of upper and lower approximations in terms of a tolerance relation R (Definition 1.9), which is the mean of attribute-specific tolerance relations R_i on $\mathbb{R}$.

Interval-valued fuzzy sets (Dubois & Prade 2005, and references therein) are defined as follows:

11.1 Definition (Interval-valued fuzzy set). Let X be a set. An interval-valued fuzzy set in X is a pair of fuzzy sets (F_1, F_2) in X such that $F_1(x) \leq F_2(x)$ for all $x \in X$.

Equivalently, an interval-valued fuzzy set in X can also be defined as a function $X \longrightarrow \mathcal{I}([0,1])$, where the range is the set of intervals in $[0,1]$, i.e. the subset of $[0,1] \times [0,1]$ whose values in the first component are always less than or equal to the values in the second component.

We can accommodate the possibility of missing data by adjoining a formal symbol denoting a missing value to each copy of $\mathbb{R}$ to obtain $\mathbb{R}_? := \mathbb{R} \cup \{?\}$, and by letting a dataset X be a multisubset of $\mathbb{R}_?^m$. The task then is to extend any choice of R_i to ?. We define *optimistic* and *pessimistic* per-attribute relations $R_i^{\max}$ and $R_i^{\min}$ by stipulating that for any $a, b \in \mathbb{R}$:

$$\begin{aligned} R_i^{\max}(a,b) = R_i^{\min}(a,b) &= R_i(a,b) \\ R_i^{\max}(a,?) = R_i^{\max}(?,b) &= R_i^{\max}(?,?) = 1 \\ R_i^{\min}(a,?) = R_i^{\min}(?,b) &= R_i^{\min}(?,?) = 0 \end{aligned} \tag{11.1}$$

Accordingly, we define interval-valued upper and lower approximations through the aggregated relations $R^{\max}$ and $R^{\min}$:

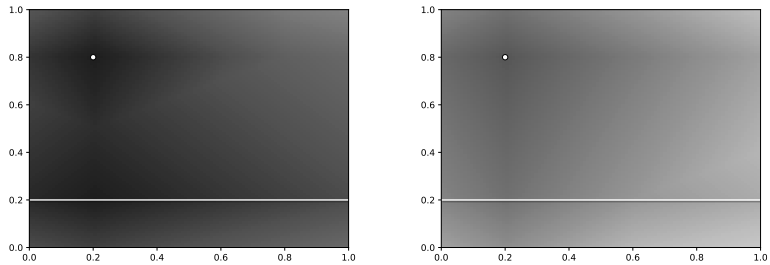
11.2 Definition (Interval-valued upper and lower approximation). Let $X \subset \mathbb{R}^m_?$ be a finite multisubset for some $m \in \mathbb{N}$, let w be a weight vector of some length k, T a t-norm and I a fuzzy implication, and let $(R_i)_{i \leq m}$ be a family of tolerance relations. Then for any fuzzy submultiset C of X, the interval-valued upper and lower approximations of C are, respectively, the interval-valued fuzzy sets $(\overline{C}^{\min}, \overline{C}^{\max})$ and $(\underline{C}_{\min}, \underline{C}_{\max})$, defined as:

$$\begin{aligned}
\overline{C}^{\min}(y) &= w \max_{x \in X} (T(R^{\min}(y,x), C(x)) \\
\overline{C}^{\max}(y) &= w \max_{x \in X} (T(R^{\max}(y,x), C(x)) \\
\underline{C}_{\min}(y) &= w \min_{x \in X} (I(R^{\max}(y,x), C(x)) \\
\underline{C}_{\max}(y) &= w \min_{x \in X} (I(R^{\min}(y,x), C(x))
\end{aligned} \tag{11.2}$$

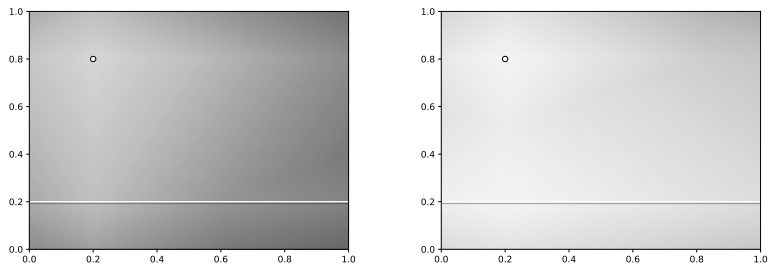
Because t-norms and fuzzy implications are respectively monotonic and anti-monotonic in the first argument, the pessimistic approximations $\overline{C}^{\min}$ and $\underline{C}_{\min}$ encode the minimum membership degrees in the upper and lower approximations, while the optimistic approximations $\overline{C}^{\max}$ and $\underline{C}_{\max}$ encode the maximum membership degrees.

Membership in the optimistic and pessimistic approximations — like membership in ordinary upper and lower approximations — is determined purely on the basis of the attribute values of an instance, so it is possible to plot membership degrees across the attribute space. This is illustrated for a toy example in Figure 11.1. Here, C is a crisp set containing two elements, one of which has a missing attribute value, which we have represented with a line. We have chosen $R_i(y,x) = 1 - |y_i - x_i|$. Recall (Chapter 1) that for crisp sets, the choice of t-norm becomes void and that we resolve the choice of fuzzy implication by choosing the standard negation $z \longmapsto 1 - z$. We set $w = \left\langle \frac{2}{3}, \frac{1}{3} \right\rangle$ (linear weights with $k = 2$). Darker shades of grey indicate higher membership degrees. It can be seen that membership degrees of the optimistic approximations are uniformly higher than membership degrees of the pessimistic approximations.

The treatment in this section is essentially an updated version of Jensen & Shen (2009). The differences are mainly practical. Firstly, Jensen & Shen (2009) use a more general setting, where R_i is an interval-valued relation, but this greater generality potentially obscures the fact that this approach can be applied in any context that currently uses ordinary fuzzy rough sets, where R_i is scalar-valued. And secondly, Jensen &



(a) $\overline{C}^{\max}$ (b) $\overline{C}^{\min}$



(c) $\underline{(X \setminus C)}_{\max}$ (d) $\underline{(X \setminus C)}_{\min}$

Figure 11.1: Toy example with $C = \{(0.2, 0.8), (?, 0.2)\}$. Missing value displayed as a line. Optimistic and pessimistic upper approximations of C and optimistic and pessimistic lower approximations of $X \setminus C$.

Shen (2009) aggregate $R^{\min}$ and $R^{\max}$ using a t-norm, instead of the mean. As a result, $R^{\min}$ will always be 0 if any of the attribute values are missing, and we lose the information encoded by the non-missing attribute values.

11.2 FRNN with interval-valued approximations

We can adapt FRNN classification to datasets with missing values by considering the membership of unseen records in the interval-valued upper and lower approximations of the decision classes. The computational complexity of calculating membership in the bounds of the interval-values upper and lower approximations is in principle the same as that of calculating membership in traditional upper and lower approximations. However, the specific definition of $R^{\min}$ and $R^{\max}$ for records with missing values prevents any clear reformulation of $R^{\min}$

and $R^{\max}$ in terms of distance measures with existing nearest neighbour search implementations, which only makes this approach practical for smaller datasets.

We test interval-valued FRNN classification with a small experiment on binary classification sets. As upper and lower approximations produce equivalent results with binary datasets, we simplify the experiment by only using the upper approximation.

As in Chapter 1, we use linearly decreasing weights of length $k = 20$. For the tolerance relation, we select $R_i(y, x) = 1 - |y_i - x_i|/r_i$, where r_i is the range of values in the training set.

We evaluate performance with the mean AUROC across 5-fold cross-validation. We apply this to eleven datasets with missing values (Section B.3).[2] Where applicable, we remove classes with fewer than five instances, and select a stratified sample of 2000 instances.

We experiment with two strategies: using the mean membership values in the optimistic and pessimistic approximations, and optimising a weighted mean on the basis of training data.

For the second strategy, we use the efficient form of leave-one-out validation detailed in Sections 3.2 and 8.1. Recall that this entails taking a single nearest neighbour query for the entire training set, and correcting it by removing nearest neighbour distances from a training instance to itself. The remaining values can then be used to calculate optimistic and pessimistic approximation memberships $\overline{C \setminus \{x\}}^{\max}(x)$ and $\overline{C \setminus \{x\}}^{\min}(x)$. As in Chapter 9, we parametrise the average of these two values with a value $\lambda \in [0, 1]$:

$$(1 - \lambda) \cdot \overline{C \setminus \{x\}}^{\min}(x) + \lambda \cdot \overline{C \setminus \{x\}}^{\max}(x) \tag{11.3}$$

We optimise λ by calculating the resulting AUROC and applying Malherbe-Powell optimisation (Section 8.2) with a budget of 20 evaluations.

Note that the computational complexity of this strategy is equal to the computational complexity of a $k + 1$-nearest neighbour query with n query instances and n target instances, where n is the size of the training set. For large n, this can potentially be mitigated by using only a subset of the training set to optimise λ.

The results are displayed in Table 11.1. Optimising the weighted mean increases AUROC for 7 datasets and decreases it for 3. Applying a one-sided Wilcoxon signed-rank test, we find that this is weakly significant ($p = 0.057$).

For comparison, we have also included the results obtained from simple imputation with the mean (numerical attributes) or mode (cat-

[2]The eleven datasets used in this chapter represent an initial selection, which we later extended for the experiments in Chapters 10 and 12.

Table 11.1: Datasets with the number of records n and missing value rate ?, as well as the AUROC from classification with the mean of optimistic and pessimistic upper approximation memberships, with an optimised ratio of both, and with normal upper approximation memberships after imputation with the mean and mode.

Dataset	n	?	Mean	Optimised	Imputation
adult	2000	0.010	**0.863**	0.863	0.860
aps-failure	2000	0.083	0.969	0.985	**0.993**
arrhythmia	443	0.003	0.878	**0.880**	0.877
ckd	400	0.105	1.000	1.000	**1.000**
exasens	399	0.428	0.738	**0.748**	0.734
hcc	165	0.102	0.746	0.741	**0.771**
hepatitis	155	0.057	0.879	**0.884**	0.877
mammographic-masses	961	0.042	0.833	**0.834**	0.827
primary-tumor	330	0.039	**0.779**	0.777	0.775
secom	1567	0.045	0.678	0.681	**0.689**
soybean	683	0.098	0.993	0.995	**0.996**

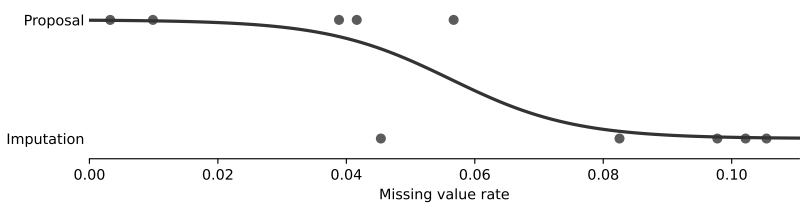


Figure 11.2: Distribution of datasets for which imputation or the proposal of this chapter achieves higher AUROC as a function of the missing value rate, with logistic regression fit.

egorical attributes) of the known values in the training data. For 6 datasets, both the mean and optimised weighted mean optimistic and pessimistic approximations achieve a higher AUROC than simple imputation, whereas for 5 datasets, simple imputation achieves a higher AUROC. If we exclude the outlying dataset *exasens*, we see that the optimistic and pessimistic approximations perform better on datasets with a lower missing value rate, and imputation on datasets with a higher missing value rate (Figure 11.2). When we fit a logistic regression model, the odds are even at a missing value rate of 0.056.

11.3 Conclusion

In this chapter, we have presented an approach towards datasets with missing values that has received relatively little attention so far. While the existing literature is typically devoted to resolving these missing

values in an optimal manner, we have argued that in the context of fuzzy rough sets, which are motivated by a desire to model different kinds of uncertainty, it is worthwhile to also model the uncertainty of incomplete information.

We have recalled the concept of interval-valued fuzzy rough sets, which iterate on the dualistic nature of rough sets and replace the upper and lower approximations by interval-valued fuzzy sets: secondary pairs of approximations, reflecting optimistic and pessimistic assumptions about the values that are missing. These define a bandwidth that contains the true (but unknown) upper and lower approximation memberships. We think that this can offer a valuable perspective for qualitative analyses of datasets with missing values.

We have shown how the interval-valued upper and lower approximations can be used to extend Fuzzy Rough Nearest Neighbour (FRNN) classification to problems with missing values. From an evaluation on several real-world datasets, we found that the best results can be obtained by taking a weighted average of the optimistic and pessimistic approximation memberships, and by optimising the relative weight on the basis of training data. This results in a comparable overall performance to simple imputation with the mean and mode, but is more directly interpretable as it does not involve the insertion of artificial values. Further analysis revealed that our proposal in particular outperforms imputation on datasets with a missing value rate below 0.056. However, a major limitation of our proposal is that it is difficult to implement efficiently, making it suitable only for datasets of up to a few thousand records.

In the next chapter, we will propose a third and final approach towards missing values that represents them as distinct values that a classifier can learn from (like the missing-indicator approach) but which does not assume a higher similarity with any of the non-missing values (like the approach in the present chapter).

Chapter 12

Polar encoding[1]

In Chapter 10, we found that the use of missing-indicators generally improves performance on real-life datasets for a range of classification algorithms. However, the missing-indicator approach can only be used in addition to, not instead of imputation, which is still required to provide a value for the original attribute. This is of particular relevance for algorithms that are based on distance, like nearest neighbours algorithms, because it means that a missing value is closest to the non-missing value that corresponds to the imputed value (Figure 12.1a). It is also an issue for decision tree algorithms, because missing values will always split together with their imputed value when the algorithm splits on the original attribute.

In this chapter, we present *polar encoding*[2] as an alternative representation of missing values that does not rely on imputation. Its default form, to be used with 1-distance or with algorithms that are not based on distance, is as follows. It represents each $[0, 1]$-scaled numerical attribute as a pair of features, with the following map:

$$\begin{aligned} x &\longmapsto \langle x, 1-x \rangle \,, \\ ? &\longmapsto \langle 0, 0 \rangle \,, \end{aligned} \tag{12.1}$$

where ? is a missing value. For categorical attributes, polar encoding corresponds to one-hot encoding, with missing values also represented as zero vectors.

We present three theoretical justifications for this proposal. Firstly, in Section 12.1, we show that polar encoding maps non-missing values onto one quadrant of a unit circle centred on missing values. As a consequence, missing values become equidistant from all non-missing

[1] This chapter is based on Lenz et al (2022d).

[2] We have chosen the name *polar encoding* as a loose analogy to polar coordinates, because values are encoded in relation to a number of poles: the origin and the unit vectors $\langle 1, 0 \rangle$ and $\langle 0, 1 \rangle$ (and higher-dimensional unit vectors for categorical attributes).

values, and we avoid having to choose a non-missing value that missing values are most similar to. In addition to our main proposal, we will give a variant of polar encoding that can be used with 2-distance.

Secondly, in Section 12.2 we argue that because polar encoding results in two copies of each $[0, 1]$-valued attribute, with missing values located at either end, it effectively allows decision tree algorithms to choose which side of each split missing values should be grouped with, and that it thus offers a practical realisation of the *missingness incorporated in attributes* (MIA) proposal by BE Twala et al (2008).

And thirdly, in Section 12.3, we present the concept of *barycentric* attributes and show that these are fuzzified categorical attributes, and also generalise numerical $[0, 1]$-valued attributes. In particular, the categorical and numerical representations of binary attributes turn out to be two sides of the same coin. Barycentric attributes also have a fuzzified equivalent of one-hot encoding, and when this is applied to $[0, 1]$-valued attributes, viewed as barycentric attributes, we obtain polar encoding.

We complement these theoretical arguments in Section 12.4 with an experimental evaluation of the practical usefulness of our proposal, by comparing it against the missing-indicator approach for a number of distance-based and decision tree classification algorithms, using 20 real-life datasets with missing values. For FRNN classification, we will also compare our approach to the use of interval-valued upper and lower approximations (Chapter 11).

Finally, we present our conclusions in Section 12.5.

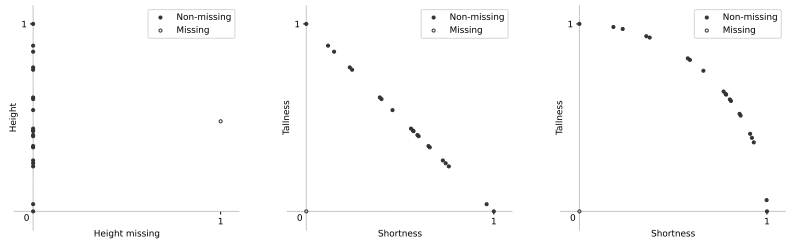
12.1 Polar encoding as mapping onto the unit circle

In this section, we will explain how polar encoding ensures that missing values are equidistant from all non-missing values, and present a variant proposal for Euclidean distance.

Boscovich distance

Recall the definition of the Minkowski p-norm of a vector $x \in \mathbb{R}^m$ from Section 2.1. The Minkowski p-norm unit sphere in $\mathbb{R}^m$ consists of all points with p-norm equal to 1. For $m = 2$, this gives us the p-norm unit circles (Figure 12.2).

Figure 12.1b illustrates the application of polar encoding with a toy example. The key observation to make is that unlike with the missing-indicator approach, the Boscovich distance between a missing value and any non-missing value is always 1. In fact, this is a simple consequence of the fact that polar encoding maps non-missing values onto the non-negative quadrant of the Boscovich (1-norm) unit circle.



(a) Missing-indicator (b) Boscovich polar encoding (c) Euclidean polar encoding

Figure 12.1: Illustrative example of a $[0, 1]$-valued attribute for height with missing value, with missing-indicator and polar encoding.

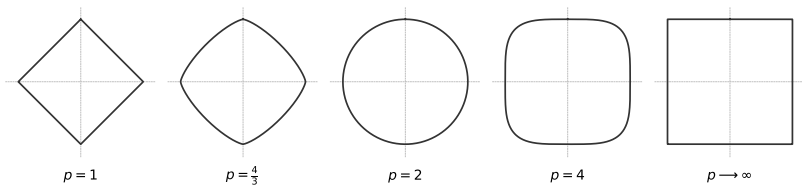


Figure 12.2: Minkowski p-norm unit circles for various values of p.

Moreover, with polar encoding, the Boscovich distance between any two non-missing values $x, y \in [0, 1]$ becomes twice the original distance $|x - y|$. In other words, the distances between non-missing values remain essentially unchanged, except for a scaling factor of 2. The Boscovich distance between a missing value and non-missing values is 1, which is exactly half the maximum distance 2 between two non-missing values, reflecting the fact that we do not know what the 'true' value of a missing value is. This distance can be used directly, or transformed into a similarity value with $a \longmapsto 1 - a/2$. In this case, the similarity between a missing value and any non-missing value is always 0.5, exactly half the maximum similarity of 1.

This contrasts with the approach taken by Dai (2013), who stipulate that the similarity between a missing value and any other value should always be 1. Similarly, in the previous chapter, we proposed propagating the uncertainty from missing values using interval-valued fuzzy sets. These interval values are bounded by an optimistic scenario, corresponding to the proposal by Dai (2013), and a pessimistic scenario, in which the similarity between a missing value and any other value (possibly also missing) is 0 (complete dissimilarity). In both cases the

problem is that missing values are not more similar to each other than to non-missing values — missing values are not treated as a signal to generalise from. Moreover, in practice these similarity relations scale poorly to larger datasets, because they do not admit straightforward implementations in terms of an existing distance measure.

Euclidean distance

Based on the discussion in the previous subsection, a straightforward way to obtain polar encoding for Euclidean distance is to map non-missing values onto the non-negative quadrant of the Euclidean (2-norm) unit circle (Figure 12.1c). We propose to do this with the following mapping, which establishes a linear correspondence between distance in $[0, 1]$ and arc length (scaling by a factor $\sqrt{2}$):

$$\begin{aligned} x &\longmapsto \left\langle \sin \frac{x \cdot \pi}{2}, \cos \frac{x \cdot \pi}{2} \right\rangle, \\ ? &\longmapsto \langle 0, 0 \rangle . \end{aligned} \tag{12.2}$$

Note that this map cannot preserve Euclidean distance. When encoding a $[0, 1]$-valued attribute in this manner, larger distances become relatively less large. However, the difference is relatively small and may not be problematic in practice. For instance, the Euclidean distance between the minimum and maximum values becomes $\sqrt{2} \approx 1.41$, which is slightly less than twice the distance between either value and the midrange:

$$\left(\left| \sin \frac{\pi}{4} - 0 \right|^2 + \left| \cos \frac{\pi}{4} - 1 \right|^2 \right)^{\frac{1}{2}} \approx 0.765.$$

Furthermore, this maximum distance between two non-missing values ($\sqrt{2}$), is now comparatively smaller than with Boscovich distance (2). This is completely consistent with the distance between two different one-hot encoded categorical values, which is likewise $\sqrt{2}$ for Euclidean distance and 2 for Boscovich distance.

For other values of p, there exist generalisations of sin and cos that could be used instead to parametrise the non-negative quadrant of the p-unit sphere (Lindqvist & Peetre 2000; Shelupsky 1959). However, these functions are defined as the inverses of integrals, and so are not easy to apply in practice.

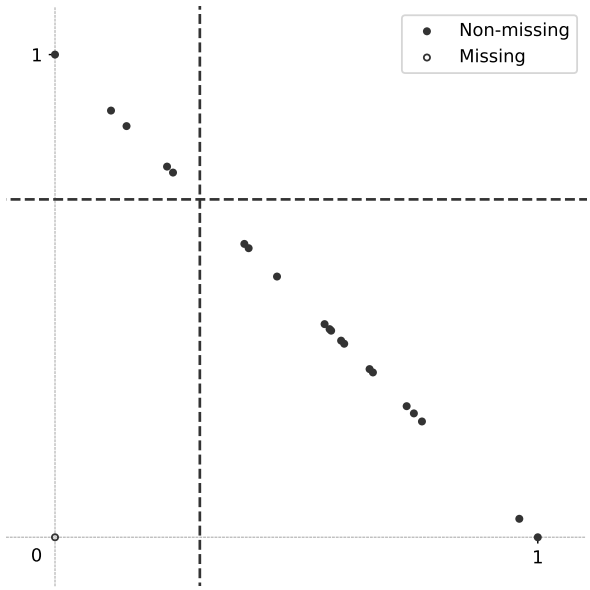


Figure 12.3: Illustrative example of equivalent splits on a polar-encoded attribute, with missing values on either side.

12.2 Polar encoding as "missingness incorporated in attributes"

Polar encoding also allows decision tree algorithms to learn from missing values. The two dimensions of a polar-encoded attribute induce identical splits on the data, except that missing values end up on either side of each split (Figure 12.3). Therefore, decision trees are effectively offered a choice as to which side of each split missing values should be grouped with. Missing values can also be split off on their own by splitting on both dimensions of a polar-encoded attribute.

This contrasts with the missing-indicator approach, where missing values either group together with their imputed value (when the tree splits on the original attribute), or alone (when the tree splits on the missing-indicator).

The effect of polar encoding on decision trees is very similar to the *missingness incorporated in attributes* (MIA) proposal by BE Twala et al (2008), which stipulates that when splitting on an attribute with missing values, the algorithm should consider each potential split twice, with missing values on either side, and additionally a split that separates non-missing and missing values. MIA has been added to the scikit-learn implementation of LightGBM (Ke et al 2017), and a similar strategy is

part of XGBoost (T Chen & Guestrin 2016). The advantage of polar encoding is that it can be applied by the user, and combined with off-the-shelf implementations of decision tree algorithms that do not natively support MIA.[3]

The performance of MIA has mostly been evaluated on the basis of simulated data with informative missing values.

For decision trees, BE Twala et al (2008) showed that MIA performs better than resolving missing values as a weighted combination of the two branches according to the prior probabilities of the non-missing values (Cestnik et al 1987), and about as good as multiple imputation with expectation maximisation (Schafer 1997), which had emerged as the two best-performing strategies in a previous comparison by B Twala (2009).

Kapelner & Bleich (2015) have shown that for Bayesian additive regression trees, MIA outperforms random forest imputation. Similarly, MIA has been shown to outperform mean imputation with missing-indicators and a handful of other strategies for regression with decision trees, Random Forest and XGBoost (Josse et al 2020).

Finally, the scikit-learn implementation of LightGBM mentioned above has also been evaluated on four large, real-life medical datasets by Perez-Lebel et al (2022), who found that MIA produces somewhat to considerably better regression and classification performance than the missing-indicator approach with various forms of imputation.

12.3 Polar encoding as representation of barycentric attributes

In this section, we will show how polar encoding can be seen as the representation of *barycentric* attributes, which generalise both categorical and $[0, 1]$-valued attributes. In particular, this explains how polar encoding generalises one-hot encoding. To begin with, we establish some working definitions.

Numerical and categorical attributes

Recall from Definition 0.6 that we can transform a categorical attribute into a tuple of numerical features through one-hot encoding. In addition to this *redundant* form of one-hot encoding, it is also possible to define *compact* one-hot encoding:

12.1 Definition. Let V be a categorical attribute. For a chosen order $V = (v_1, v_2, \ldots, v_p)$, its *compact one-hot encoding* is the map $V \longrightarrow$

[3] A similar trick is suggested by Josse et al (2020): repeat each attribute with missing features twice, and encode missing values alternatively as $-\infty$ and $+\infty$.

$$\left(\begin{array}{c|c|c} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{array}\right) \qquad \left(\begin{array}{ccc} 0 & 0 & 1 \\ \hline 0 & 0 & 1 \\ \hline 1 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 1 & 0 & 0 \\ \hline 0 & 0 & 1 \end{array}\right)$$

(a) Crisp partition (b) Categorical attribute

Figure 12.4: Example illustrating the correspondence between crisp partitions and categorical attributes of a dataset X. Rows correspond to the records of X, columns to the partition classes and categories. The values 1 and 0 indicate membership and non-membership, respectively.

$[0,1]^{p-1}$ that sends v_p to $\mathbf{0}$ and v_i for $i < p$ to the standard basis vector $\mathbf{e}_i = \langle 0, \ldots, 0, 1, 0, \ldots, 0 \rangle$.

Compact one-hot encoding is sufficient to ensure that all categorical values are linearly separable, but it also introduces an asymmetry that can be undesirable.

12.2 Remark. Binary attributes can be represented both as categorical attributes and as numerical attributes. In the latter case, a typical choice is to use the values 0 and 1. This numerical representation corresponds directly to a compact one-hot encoding of its categorical representation. We will exploit this correspondence to argue that barycentric attributes generalise not just categorical, but also $[0,1]$-valued numerical attributes.

It is a classical observation that categorical attributes correspond to partitions (Quinlan 1986). Formally, a categorical attribute V induces a partition on a dataset X through the equivalence relation that equates elements of X with the same value in V. Conversely, if we have a partition $\mathcal{U}$ of X, we can derive a categorical attribute of X that takes, for each $x \in X$, the value U in $\mathcal{U}$ that contains y.

Both categorical attributes (through one-hot encoding) and partitions can be represented with a matrix of values in $\{0,1\}$, with exactly one value equal to 1 on each row (Figure 12.4). Later in this section, we will extend this correspondence between categorical attributes and partitions to barycentric attributes and fuzzy partitions.

Barycentric attributes

Barycentric values (or *coordinates*; also known as *homogeneous* coordinates) are numerical values that sum to a fixed number (typically 1), or where

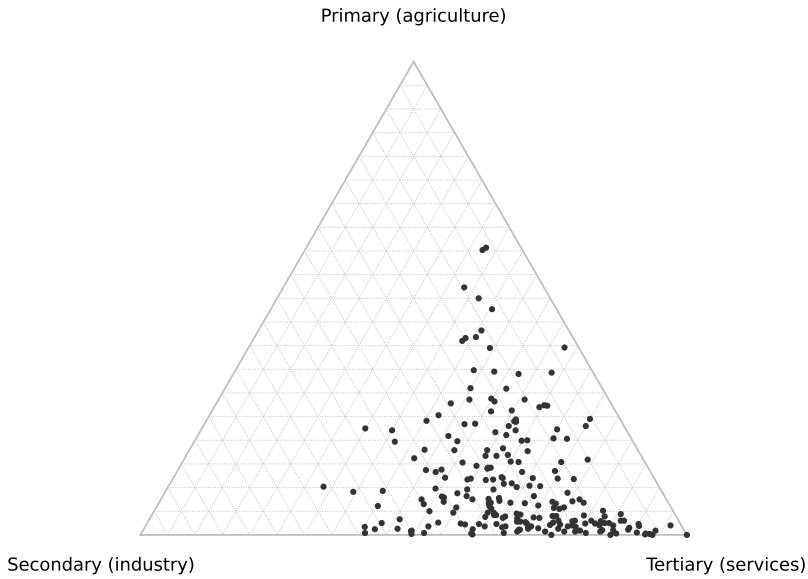


Figure 12.5: Example of a ternary plot: distribution of GDP over economic sectors of countries and territories (CIA World Factbook 2022).

only the relative proportions are considered important. The concept dates back to at least Möbius (1827), who used it to express a point as the weighted sum (the *barycentre*, where the weights cancel each other out) of the vertices of a simplex (Allardice 1891; Boyer 1956). Barycentric values are also used to define the points that make up projective space. For the purpose of the present chapter, we will assume that barycentric values are non-negative, and use the following formal definition:

12.3 Definition. An attribute is *barycentric* if it is equal to a copy of $\left(\mathbb{R}^m_{\geq 0} \setminus \{\mathbf{0}\}\right)/\!\sim$ for some $m \geq 1$, where $\sim$ is the equivalence relation defined by $\langle x_1, x_2, \ldots, x_m\rangle \sim \langle \lambda x_1, \lambda x_2, \ldots, \lambda x_m\rangle$ for all $\lambda \in \mathbb{R}_{>0}$. The *normalised representation* of a value $[x_1, x_2, \ldots, x_m] \in \left(\mathbb{R}^m_{\geq 0} \setminus \{\mathbf{0}\}\right)/\!\sim$ is the vector $\langle x_1/s, x_2/s, \ldots, x_m/s\rangle \in \mathbb{R}^m$, where $s := \sum_{i \leq m} x_i$.

Barycentric values are often encountered in the literature in the form of ternary plots (Figure 12.5), which display the relative frequencies of three components. Recent examples include the composition of planets (core, mantle and hydrosphere) (Haldemann et al 2022; Huang et al 2022; MacDonald et al 2022), seabed sediment (F Wang et al 2021), ternary mixtures of fluids (Stemplinger et al 2021; Tönsmann et al 2021), ternary

compounds (WC Chen et al 2021; Nolan et al 2021) and even human behaviour (M Kim et al 2021; Molter et al 2022).

In addition, some machine learning problems are typically approached by considering relative token frequencies. For instance, this can be part of the calculation of the cosine similarity between text records (Sangma et al in press; Tian et al 2021; Zhao & Mao 2018).

Finally, the confidence scores produced by a classification model (or some other estimate), when normalised to sum to 1, are also a natural example of barycentric values.

Barycentric attributes as fuzzified categorical attributes

Barycentric attributes generalise categorical attributes in the following way. If $(\mathbb{R}^m_{\geq 0} \setminus \{\mathbf{0}\})/\sim$ is a barycentric attribute, then the subset V of values with only one non-zero coefficient forms a categorical attribute, and we will write $B(V) := (\mathbb{R}^m_{\geq 0} \setminus \{\mathbf{0}\})/\sim$ and say that V is the set of categories of $B(V)$. In particular, the normalised representation of $B(V)$ reduces precisely to one-hot encoding when restricted to V.

This relationship can also be understood geometrically. The set of normalised representations of a barycentric attribute coincides with the standard $m-1$-simplex, which is spanned by m vertices, the one-hot encoded values of V.

Conversely, barycentric attributes can be understood as fuzzified categorical attributes, allowing us to give a fuzzy answer to the question of category membership:

12.4 Remark. Let $B(V)$ be a barycentric attribute with m categories. Then we can associate to each value in $B(V)$ with normal representation $\langle x_1, x_2, \ldots, x_m \rangle$ the fuzzy set in V with membership degrees $x_1, x_2, \ldots, x_m$. These are precisely the fuzzy sets in V with cardinality 1.

This is reinforced by the fact that barycentric attributes correspond to fuzzy partitions in the same way that categorical attributes correspond to crisp partitions (as discussed at the beginning of this section). Recall the definition of a fuzzy partition (Dunn 1974; Ruspini 1969):

12.5 Definition. Let X be a finite set. A *fuzzy partition* on X is a finite set $\mathcal{F}$ of fuzzy sets in X such that, for each $x \in X$, we have $\sum_{F \in \mathcal{F}} F(x) = 1$.

To see that a barycentric attribute $B(V)$ on a dataset X contains the same information as a fuzzy partition on X, consider that both can be represented by a $|X| \times |V|$ matrix of values in $[0, 1]$, such that the rows sum to 1 (Bezdek & Harris 1978). The columns of such a matrix correspond to a fuzzy partition (Figure 12.6a), whereas its

$$\left(\begin{array}{c|c|c} 0.2 & 0.6 & 0.2 \\ 0.1 & 0.9 & 0.0 \\ 0.4 & 0.1 & 0.5 \\ 1.0 & 0.0 & 0.0 \\ 0.1 & 0.9 & 0.0 \\ 0.2 & 0.2 & 0.6 \end{array}\right)$$

(a) Fuzzy partition

$$\left(\begin{array}{ccc} 0.2 & 0.6 & 0.2 \\ \hline 0.1 & 0.9 & 0.0 \\ \hline 0.4 & 0.1 & 0.5 \\ \hline 1.0 & 0.0 & 0.0 \\ \hline 0.1 & 0.9 & 0.0 \\ \hline 0.2 & 0.2 & 0.6 \end{array}\right)$$

(b) Barycentric attribute

Figure 12.6: Example illustrating the correspondence between fuzzy partitions and fuzzy categorical attributes of a dataset X. Rows correspond to the records of X, columns to the partition classes and categories. Values are membership degrees.

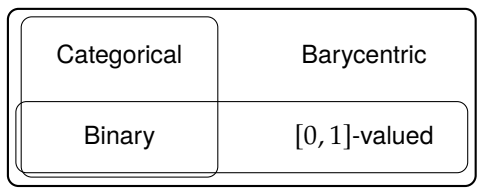


Figure 12.7: Euler diagram of different attribute types. Barycentric attributes generalise both categorical and $[0, 1]$-valued attributes.

rows correspond to the normalised values of a barycentric attribute (Figure 12.6b).

$[0, 1]$-valued attributes as barycentric attributes

Just as one-hot encoding is redundant and we can use compact one-hot encoding to represent the same information with one fewer value (Definition 0.6), so the normalised representation of a barycentric attribute $\langle x_1, x_2, \ldots, x_m \rangle$ is redundant, and we can encode it compactly as $\langle x_1, x_2, \ldots, x_{m-1} \rangle$. Together, these compactly encoded values form the $m-1$-simplex in $\mathbb{R}^{m-1}$ spanned by the standard $m-2$-simplex and the origin. Conversely, we can reconstruct the full representation from a compactly encoded value $\langle x_1, x_2, \ldots, x_{p-1} \rangle$ by appending the value $1 - \sum_{i \leq p-1} x_i$.

The compact encoding of a barycentric attribute with only two categories is a single value in $[0, 1]$. This leads us to the following observation:

12.6 Remark. Let A be a $[0, 1]$-valued attribute. Then the values of A are compactly encoded values of a barycentric attribute with two categories. We obtain the corresponding redundant representation

with $x \longmapsto \langle x, 1-x \rangle$. Thus, barycentric attributes generalise not just categorical attributes, but also $[0,1]$-valued attributes (Figure 12.7).

This redundant representation of $[0,1]$-valued attributes generalises the categorical representation of binary attributes that we noted in Remark 12.2. We can illustrate this with an example. Suppose that we have a binary attribute denoting height, with two values, 'short' and 'tall'. Its compact encoding is as a single numerical attribute A with two values, 0 and 1, expressing 'tallness'. Its redundant encoding is as two numerical attributes, tallness (A) and shortness ($1-A$). Likewise, suppose that we have $[0,1]$-valued attribute A' denoting height, then its redundant encoding $\langle A', 1-A' \rangle$ consists of fuzzy expressions of 'tallness' and 'shortness'.

Of course, this redundant encoding of a $[0,1]$-valued attribute is precisely the polar encoding that we propose in this chapter (Figure 12.1b).

Representing missing values

We now turn to the representation of missing values. Recall our example from the previous subsection: suppose that we have a barycentric attribute $B(V)$ denoting height, with V containing the two categories 'tall' and 'short', then a missing value does not convey positive information about either category. Therefore, we accommodate the possibility that a barycentric attribute can have a missing value by expanding the set $\left(\mathbb{R}^m_{\geq 0} \setminus \{\mathbf{0}\}\right)/\!\sim$ to $\mathbb{R}^m_{\geq 0}/\!\sim$, and by stipulating that the normalised representation of $[0,0,\ldots,0]$ is the zero vector $\mathbf{0}$. This corresponds to the unique fuzzy set in V with cardinality 0 (the empty set).

Note that barycentric attributes with missing values can no longer be represented compactly, since doing so would also encode the non-missing value $[0,0,\ldots,1]$ as $\mathbf{0}$. It is precisely the redundancy of the redundant normal representation (in particular, redundant one-hot encoding) that enables us to encode missing values as zeroes. For $[0,1]$-valued numerical attributes, this means that our proposed polar encoding is necessary if we want to represent missing values.

12.4 Experimental evaluation

We now describe our experimental evaluation of using polar encoding for classification. Concretely, we ask whether it leads to better classification performance than the traditional approach of mean/mode imputation with missing-indicators.

Table 12.1: Classification Algorithms used for the experimental comparison

Distance-based classifiers	
NN	Nearest Neighbours (Fix & Hodges 1951)
NN-D	Nearest Neighbours, distance-weighted (Dudani 1976)
FRNN	Fuzzy Rough Nearest Neighbours, mean approximation with linear weights and $k = 20$ (Chapter 1)
SVM-G	Soft-margin Support Vector Machine (Cortes & Vapnik 1995) with Gaussian kernel
Decision tree classifiers	
CART	Classification and Regression Tree (Breiman et al 1984)
RF	Random Forest(Breiman 2001)
ERT	Extremely Randomised Trees(Geurts et al 2006)
ABT	Ada-Boosted Trees (Freund & Schapire 1995) with SAMME (stagewise additive modeling using a multi-class exponential loss function) (Zhu et al 2009)
GBM	Gradient Boosting Machine (Friedman 2001)

Setup

We evaluate polar encoding for two sets of classifiers: distance-based and decision tree based algorithms (Table 12.1). For the Support Vector Machine with Gaussian kernel that is based on Euclidean distance, we evaluate the Euclidean variant of polar encoding, while for the nearest neighbour algorithms that allow setting the distance measure as a hyperparameter, we evaluate both the standard and the Euclidean variant.

For all classifiers we use the implementations provided by scikit-learn, except for FRNN, where we use our own implementation in fuzzy-rough-learn (Appendix A). Based on the findings in Chapter 10, we apply cost complexity pruning ($\alpha = 0.01$) with CART, set the number of trees of ERT to 1000, and apply early-stopping with GBM. Otherwise, we use default hyperparameter values.

We evaluate classification performance with the AUROC. For each dataset, we perform five-fold stratified cross-validation, repeat this five times for different random divisions of the data, and reduce the resulting 25 AUROC scores by taking the mean. To establish whether the performance of polar encoding vis-à-vis imputation with missing-indicators generalises to other (similar) datasets, we test for significance using one-sided Wilcoxon signed-ranks tests.

We use the same collection of twenty datasets with naturally occurring missing values that we previously used in Chapter 10 (described in Section B.3). We rescale numerical attributes to $[0, 1]$, before applying polar encoding or mean/mode imputation with missing-indicators. In the latter case, we then also apply one-hot encoding to categorical

Table 12.2: p-values of one-sided Wilcoxon signed-rank tests of barycentric encoding vs mean/mode imputation with missing-indicators, for distance-based classifiers

Classifier	Boscovich distance	Euclidean distance
NN	0.18	0.13
NN-D	0.18	0.15
FRNN	0.0021	0.0040
SVM-G		0.018

attributes.

For the FRNN classifier, we separately compare our results against the approach proposed in the previous chapter, using interval-valued upper and lower approximations. The interval-valued class predictions are transformed into specific values by averaging the lower and upper bounds. Specifically, we optimise a weighted mean through leave-one-out validation on the training data. We improve upon the experiment in the previous chapter by using not just the upper, but also the lower approximation, for which we optimise the weighted average separately, and calculating the mean of both. However, we still run into the limitation that the custom tolerance relations prevent the use of standard nearest neighbour search algorithms, and thus, their application to large datasets. Therefore, we downsample datasets for this approach to 2000 records, and we similarly evaluate polar encoding on these downsampled datasets when comparing against interval-valued fuzzy rough sets.

Results

The full results of our experiments are listed in the appendix (Section C.3).

Table 12.2 lists the p-values of the AUROC scores obtained with polar encoding versus mean/mode imputation with missing-indicators, for distance-based classifiers. For all classifiers, and both Boscovich and Euclidean distance, polar encoding leads to improved results on our selection of datasets. In particular, it performs significantly better for FRNN and SVM-G. The fact that the p-values for Euclidean distance are not higher than the p-values for Boscovich distance seemingly indicates that the distortion introduced by Euclidean polar encoding is not harmful for classification performance.

In addition to these results on the full datasets, FRNN also scores better with polar encoding than with interval-valued upper and lower approximations on the datasets downsampled to 2000 records ($p = 0.18$).

The picture is more mixed for decision tree classifiers (Table 12.3). Polar encoding leads to improved performance for CART, ERT and

Table 12.3: p-values of one-sided Wilcoxon signed-rank tests of barycentric encoding vs imputation with missing-indicators, for decision tree classifiers.

Classifier	p
CART	0.060
RF	0.42
ERT	0.14
ABT	0.054
GBM	0.63

ABT, and about the same performance as mean/mode imputation with missing-indicators for RF and GBM.

12.5 Conclusion

In this chapter we have presented the polar encoding of categorical and $[0, 1]$-valued attributes as a solution for the representation of missing values. It preserves the information encoded in their distribution, does not require the use of imputation, and can be easily used as input for existing machine learning algorithms. For distance-based algorithms, it ensures that missing values are equidistant from all non-missing values, while for decision tree algorithms, it allows splits to choose how to group missing values.

We have provided further justification for our proposal by giving a formal definition of barycentric attributes, and showing that they can be understood as fuzzified categorical attributes and also generalise $[0, 1]$-valued attributes. Moreover, the normalised representation of barycentric attributes reduces to traditional one-hot encoding for categorical values. Because this representation is slightly redundant, using one more dimension than strictly necessary, it allows us to represent missing values as zero vectors, symbolising the absence of information.

Having previously shown in Chapter 10 that missing-indicators improve classification performance on real-life datasets, in the present chapter we conducted an experiment to test whether polar encoding works even better. For the distance-based classifiers, this was the case on our selection of datasets, with significant improvement in particular for FRNN and SVM-G. For the decision tree classifiers, the picture was more mixed, but polar encoding improved performance for CART, ERT and ABT.

Accordingly, we recommend polar encoding as a conservative baseline approach for missing values that seemingly performs as well or better than the missing-indicator approach. For classifiers where there

is no notable difference in performance, like RF and GBM, practitioners may want to try out both approaches.

Conclusion

We will end this thesis by summing up the main results (Section 12.5) and discussing some open questions and possibilities for future work (Section 12.5).

Results

Part I In Chapter 1, we started by tracing the development of fuzzy rough nearest neighbour (FRNN) classification. We then argued that by adapting ordered weighted averaging (OWA) operators into weighted mimima and maxima, predictions by FRNN can be efficiently calculated using nearest neighbour queries. In Chapter 2, we reviewed the concept of Minkowski distance, with the important special cases of Boscovich, Euclidean and Chebyshev distance. We then recalled how Minkowski distance can also be used to define certain measures of central tendency and dispersion. In particular, we identified r_1, the mean absolute distance to the median, as the measure of dispersion induced by Boscovich distance. In Chapter 3, we conducted an experiment on fifty real-life datasets and found that on average, nearest neighbour (NN) and FRNN classification perform best with Boscovich distance and r_1 scaling and that FRNN outperforms NN, for which we recommend linear distance-weights. Finally, we determined good default choices for the number of neighbours k.

Part II Chapter 4 contained a small case study, which demonstrated that FRNN can be scaled to very large datasets using distributed computing, although this approach has several limitations and is perhaps not very practical. More significantly, we also showed in Chapter 5 that it is possible to fundamentally reduce the run time complexity of FRNN by substituting approximate nearest neighbour queries, which incur only a minimal loss in classification performance.

Part III This part was devoted to one-class classification. We proposed average localised proximity (ALP), a new data descriptor

(Chapter 6), and used a series of 246 one-class classification tasks to determine good default hyperparameter values for ALP and other data descriptors, and to show that ALP has the best overall performance (Chapter 7). In Chapter 8, we found that the Malherbe-Powell algorithm is a good choice for optimising the hyperparameter values of data descriptors, and we determined the number of necessary evaluations. After hyperparameter optimisation, the support vector machine (SVM) data descriptor performs slightly better than ALP, but ALP can be optimised much more efficiently. Finally, in Chapter 9, we showed that upper and lower approximations can be reinterpreted as models produced by a weighted nearest neighbour distance (WNND) data descriptor. We evaluated its performance for one-class classification, and concluded that it is an interesting alternative for nearest neighbour distance (NND). We also proposed a generalisation of FRNN as a one-class classification ensemble, but were not able to improve its overall multiclass performance by substituting different data descriptors and optimising their hyperparameter values on the basis of one-class classification. However, we found some evidence that an ensemble based on ALP does perform better for large datasets with a small number of decision classes.

Part IV In the last part, we evaluated three approaches that allow classifiers to learn the information encoded by missing values. In Chapter 10, we showed that missing-indicators increase classification performance for a range of algorithms, and that it suffices to pair them with mean/mode imputation. In Chapter 11, we showed that by using interval-valued fuzzy sets, FRNN can directly represent the uncertainty from missing values. However, this approach is difficult to implement efficiently, and it only possibly outperforms mean/mode imputation for datasets with few missing values. Finally, in Chapter 12, we proposed polar encoding, a representation of missing values with $[0, 1]$-scaled data that does not require imputation. We showed that this makes missing values equidistant from all non-missing values and provides a practical implementation of a previous proposal for decision trees, missingness incorporated in attributes (MIA). We also argued that polar encoding can be interpreted as the standard representation of barycentric attributes, which generalise both categorical and $[0, 1]$-valued attributes. We showed that polar encoding outperforms missing-indicators for nearest neighbour algorithms, support vector machines and some decision tree algorithms, while for others there is no real difference.

We also wish to highlight three results of potentially broad practical

relevance for other researchers:

- We have generally obtained better results with Boscovich distance than with Euclidean distance, whereas many machine learning applications appear to employ Euclidean distance by default. This is understandable, because Euclidean distance has a number of special properties and is most natural in a physical sense, but our findings suggest that, where possible, this unspoken assumption should be challenged.
- The efficient leave-one-out validation technique that we have identified for nearest neighbour classifiers (Section 3.2) seems generally preferable to cross-validation when we need a single validation score (e.g. for hyperparameter optimisation) and are not interested in calculating model variance.
- We have converted (and preprocessed where necessary) a large collection of real-life classification datasets from the UCI repository for machine learning into a standard format (Appendix B). These datasets can now easily be used by other researchers for benchmark purposes.

Open questions and future work

In this last section, we will consider a number of open questions and suggest avenues for future research.

Revisiting older fuzzy rough set algorithms

In view of the fact that we were able to make FRNN classification more powerful and efficient through the use of weighted maxima and minima instead of full-length OWA operators, it could be worthwhile to revisit and improve a number of other, more specialised fuzzy rough set algorithms, like FROVOCO (imbalanced classification), FRONEC (multilabel classification), FRNN (regression), FRFS (feature selection) and FRPS (instance selection). (These are briefly discussed in Section A.5.)

The relative scale of numerical and categorical attributes

The measures of dispersion that we evaluated in Chapter 3 determine the relative scale between numerical attributes. But when a dataset additionally has categorical attributes, one also has to consider the relative scale between numerical and categorical attributes. If one adopts one-hot encoding, this question is reduced to the absolute scale of numerical attributes. That is, in the presence of categorical attributes, not just the choice between scaling by the standard deviation and the

half-range matters, but also the choice between scaling by the half-range and by the range (or any other multiple), which is largely irrelevant in purely numerical datasets.

Applying ALP to unsupervised outlier detection

Given that ALP performed well in our one-class classification experiments, a natural question to ask is whether it is also suitable for unsupervised outlier detection.

The influence of scale on the default hyperparameter values of data descriptors

For our one-class classification experiments in Chapter 7, we chose to scale by the interquartile range. It would be worthwhile to investigate to which extent different scales affect the recommended default hyperparameter values.

Realistic one-class classification problems

The data descriptors in Chapters 7 and 8 were evaluated on the basis of one-class classification tasks derived from multiclass datasets. One may ask whether these tasks are sufficiently representative of real-life problems for which one-class classification is a good fit. Finding better tasks is complicated by the fact that one-class classification is appropriate particularly in contexts where it is difficult to obtain a representative sample of negative records, but that for the purpose of *evaluation*, we ought to have such a representative sample.

As a first step to address this situation, we would like to suggest a systematic inventorisation of one-class classification applications in the literature, the collection of datasets used in these studies, and a critical evaluation of whether one-class classification is really the best approach to solve these problems.

Fuzzy rough one-class ensembles

It remains an open question whether lower approximations can be incorporated into one-class ensembles to obtain a classifier with even better performance than FRNN. This will likely require a deeper understanding of why using one-class classification to optimise hyperparameter values separately for each decision class appears to increase the overall performance of upper approximations, but not of lower approximations. If this cannot be overcome, it may be possible to combine globally optimised lower approximations with class-wise optimised upper approximations.

Once this issue is solved, an additional step would be to combine the use of lower approximations with other proposals for one-class ensembles in the literature.

Polar encoding for other scales

A limitation of polar encoding is that it pertains to $[0, 1]$-scaled data. Therefore, a natural direction for future research is its extension to numerical data that is scaled differently. The useful properties of polar encoding could be preserved for a central region of each attribute corresponding to the interval $[0, 1]$.

A benchmark collection of classification datasets

The collection of multiclass datasets that we have used in this thesis could form the core of a standardised benchmark test. For this purpose, the collection should be made more robust by increasing its size (to mitigate overfitting on this particular collection of datasets), and in particular by increasing its diversity. This presupposes the identification of important dataset characteristics, like the imbalance ratio, the number of classes, the types of attributes and the difficulty of the classification task, in terms of which diversity may be defined. Having a large, diverse collection of datasets with interesting characteristics would not just allow one to empirically evaluate overall classification performance, but also to determine whether a given algorithm is specifically suited to certain classification tasks.

Appendices

Appendix A

fuzzy-rough-learn[1]

In this appendix, we describe the Python library *fuzzy-rough-learn*. The principal goal of fuzzy-rough-learn is to make soft computing algorithms from the literature available to other researchers, both to enable their application to practical problems and to facilitate empirical comparisons and further development. It includes some of the algorithms in this thesis, as well as reference implementations for a number of older fuzzy rough set algorithms.

fuzzy-rough-learn is inspired by scikit-learn (Pedregosa et al 2011) and relies on some of its algorithms as a backend. scikit-learn is one of the most popular Python libraries for general machine learning, and only admits "well-established algorithms" with at least 200 citations[2], a criterion not (yet) satisfied by most algorithms in fuzzy-rough-learn.

The target audience for fuzzy-rough-learn is researchers with some programming skills, in particular those who are familiar with scikit-learn. We envision two principal use cases:

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

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

We will give brief sketches of existing implementations (Section A.1), the release history of fuzzy-rough-learn (Section A.2), its formal properties (Section A.3) and our design principles (Section A.4), before

[1] This appendix is based on Lenz et al (2020a) and Lenz et al (2022a).

[2] `https://scikit-learn.org/stable/faq.html`

describing the core algorithms (Section A.5) and utility functions (Section A.6) in fuzzy-rough-learn and briefly discussing the future direction of work (Section A.7).

A.1 Background

Since its conception by Dubois & Prade (1990), fuzzy rough set theory has been applied as part of a growing number of machine learning algorithms (Vluymans et al 2015b). Simultaneously, the distribution and communication of machine learning algorithms has spread beyond academic literature to a multitude of publicly available software implementations (Jović et al 2014; Nguyen et al 2019; Z Wang et al in press). And also during the same period, Python has grown from its first release (Rossum & Boer 1991) to become one of the world's most popular high-level programming languages. Python has become especially popular in the field of data science, in part due to the self-reinforcing growth of its package ecosystem.

Only a limited number of fuzzy rough set machine learning algorithms have received publicly available software implementations. Variants of FRNN classification, *fuzzy rough rule induction* (Jensen et al 2009), *fuzzy rough feature selection* (FRFS, Cornelis et al 2010) and *fuzzy rough prototype selection* (FRPS, Verbiest et al 2013; Verbiest 2014) are included in the R package *RoughSets* (Riza et al 2014), and have also been released for use with the Java machine learning software suite WEKA (Hall et al 2009; Jensen 2010).

So far, none of these algorithms seem to have been made available for Python in a systematic way.

A.2 Release history

To date, there have been three minor releases of fuzzy-rough-learn (as well as a number of bug patches):

0.0 | 2019-07-30

- Proof of concept, implementation of FRNN classification.

0.1 | 2020-06-22

- Reference implementations of FROVOCO, FRONEC, FRFS and FRPS.

0.2 | 2021-09-13

- Expansion of the scope of fuzzy-rough-learn to also cover one-class classification (Chapter 6) to facilitate exploration of the conceptual overlap with upper and lower approximations (Chapter 9).
- Inclusion of FRNN regression, additional hyperparameters for FROVOCO.
- Incorporation of a convenient method for specifying feature preprocessing.
- Redesign of class structure and hyperparameters, to make it easier for users to control the behaviour of the included algorithms and supplement their own alternatives.

A.3 Formal specifications

fuzzy-rough-learn is hosted on the two principal repositories for Python libraries, pipy and conda-forge, and thus can easily be installed with either `pip` or `conda`. API documentation is integrated into the code and automatically updated on the documentation website[3] whenever a new version is released, and includes references to the literature. fuzzy-rough-learn also has an integrated test suite to limit the opportunities for bugs to be introduced.

The source code is accessible on GitHub[4], which also offers users the opportunity to report issues or contribute improvements or additional material. fuzzy-rough-learn is distributed under the MIT license (Saltzer 2020), making it freely usable for any purpose.

fuzzy-rough-learn 0.2 requires Python 3.7 or later. Its dependencies are NumPy 1.17, SciPy 1.1 and scikit-learn 0.22. In addition, use of the EIF data descriptor requires the eif library, and the SAE feature preprocessor requires TensorFlow and Keras.

A.4 Design principles

The two primary design principles of fuzzy-rough-learn are consistency and modularity. To achieve this, we have chosen a uniform structure based on NumPy arrays and functions. As in scikit-learn, datasets are n by m two-dimensional arrays, where n is the number of records, and m the number of features. Machine learning models are functions that take a (test) dataset and return one or more values for each record. This means that machine learning *algorithms* are second-order functions,

[3] `https://fuzzy-rough-learn.readthedocs.io`

[4] `https://github.com/oulenz/fuzzy-rough-learn`

Core algorithms

Classifiers

FRNN

FRONEC

FROVOCO

Data Descriptors

ALP

CD

EIF‡

IF†

LNND

LOF

MD

NND

SVM†

Feature Preprocessors

FRFS

LinearNormaliser

IQRNormaliser

MaxAbsNormaliser

RangeNormaliser

Standardiser

SAE

VectorSizeNormaliser

Instance Preprocessors

FRPS

Regressors

FRNN

Utility functions

Array functions

div_or

first

greatest

last

least

remove_diagonal

soft_head

soft_max

soft_min

soft_tail

Dispersion measures

interquartile_range

maximum_absolute_value

standard_deviation

total_range

Location measures

maximum

mean

median

midhinge

midrange

minimum

Neighbour search methods

BallTree†

KDTree†

Parametrisations

log_multiple

multiple

T-norms

goguen_t_norm

heyting_t_norm

lukasiewicz_t_norm

Transformations

contract

shifted_reciprocal

truncated_complement

Vector size measures

MinkowskiSize

Weights

ConstantWeights

ExponentialWeights

LinearWeights

QuantifierWeights

ReciprocallyLinearWeights

Other (postprocessing)

discretise

probabilities_from_scores

select_class

Figure A.1: Schematic overview of the contents of fuzzy-rough-learn. †Wrapper for implementation in scikit-learn. ‡Wrapper for implementation in eif library.

Table A.1: Internal subdivision of fuzzy-rough-learn.

Name	Description
`neighbours`	Nearest neighbour algorithms
`networks`	Neural networks
`statistics`	Statistical functions
`support_vectors`	Support vector machines
`trees`	Decision trees
`uncategorised`	Other functions

which take a (training) dataset and return a model. An algorithm is supervised if, in addition to a dataset, it also takes an array of labels.

fuzzy-rough-learn uses classes, but in a restricted way. At present, the classes only possess two user-facing methods: the initialisation method `__init__`, and the call method `__call__`. Initialisation allows users to set hyperparameter values, while `__call__` determines that an object of this class, once initialised, behaves just like a function.[5] Therefore, we call this a parametrisable function. To make it clear for users whether a function is a parametrisable function that must be initialised, we use `CamelCase` if this is the case, and `snake_case` if not (in line with the Python convention for class and function names).

The advantage of this design pattern is that it offers users a large amount of flexibility. They can control the principal design choices of each algorithm by setting one or more hyperparameter values. If a user requires greater flexibility, they can substitute their own class by inheriting from one of the abstract base classes. And finally, thanks to the concept of duck typing in Python, they can also substitute any simple function that accepts the relevant input and produces the required output.

Conceptually, the functions in fuzzy-rough-learn can be subdivided into core algorithms and utility functions (Figure A.1, Sections A.5 and A.6). The source code is organised into domains of machine learning (Table A.1). However, end users are presented with a flat hierarchy, which only groups functions according to their functionality (the groups in Figure A.1). Thus, e.g. the FRNN classifier can be imported with:

```
from frlearn.classifiers import FRNN
```

Among the algorithms, feature preprocessors play a special role, since they are not generally used on their own, but in combination with one of the other algorithms. For users, this can be impractical, since it is

[5] To use the technical term, it is a 'callable'. Classes themselves are also callables, and many so-called functions in the Python standard library are in fact classes.

not enough to apply each feature preprocessor on the training data to construct the respective models, but each model then has to be applied to transform the training data, as well as any and all test data, in the correct order. Therefore, we automate this process and allow users to supply any number of feature preprocessors as a hyperparameter when initialising any algorithm. We also use this functionality to equip some algorithms with default feature preprocessors, which may be overridden by the user if so desired.

Algorithm 1 contains a small example illustrating the use of fuzzy-rough-learn.

Algorithm 1 Application of FRNN classification to the *iris* dataset using fuzzy-rough-learn.

```
from sklearn import datasets
from sklearn.metrics import accuracy_score, roc_auc_score
from sklearn.model_selection import train_test_split

from frlearn.base import probabilities_from_scores, select_class
from frlearn.classifiers import FRNN
from frlearn.feature_preprocessors import RangeNormaliser

# Import example data.
iris = datasets.load_iris()
X = iris.data
y = iris.target

# Split into train and test sets.
X_train, X_test, y_train, y_test = train_test_split(
    X, y, stratify=y, random_state=0
)

# Create an instance of the FRNN classifier, construct the model,
# and query on the test set.
clf = FRNN(preprocessors=(RangeNormaliser(), ))
model = clf(X_train, y_train)
scores = model(X_test)

# Convert scores to probabilities and calculate AUROC.
probabilities = probabilities_from_scores(scores)
auroc = roc_auc_score(y_test, probabilities, multi_class='ovo')
print('AUROC:', auroc)

# Select classes with the highest scores and calculate accuracy.
classes = select_class(scores)
accuracy = accuracy_score(y_test, classes)
print('accuracy:', accuracy)
```

A.5 Core algorithms

In this section, we discuss the main algorithms included in fuzzy-rough-learn. As a general rule and where applicable, users can set the dissimilarity measure, the number of nearest neighbours, the weights used for aggregation, and the nearest neighbour search algorithm.

Classifiers

FRNN

See Chapter 1.

FROVOCO

Fuzzy Rough OVO COmbination (FROVOCO, Vluymans et al 2018b) is an ensemble classifier specifically designed for, but not restricted to, imbalanced data, which adapts itself to the Imbalance Ratio (IR) between classes. It balances one-versus-one decomposition with two global class afinity measures.

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

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

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

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

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

The final class score is calculated from these components as follows:

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

We have added hyperparameters to FROVOCO that control the IR threshold that determines which classification subtasks are considered imbalanced, as well as the weights that are used for balanced and imbalanced subtasks.

FRONEC

Fuzzy Rough Neighbourhood Consensus (FRONEC, Vluymans et al 2018a) is a multilabel classifier. It combines the instance similarity R, based on the attribute values of instances, with the label similarity R_d, which is based on the label sets of instances. It offers two possible definitions for R_d. The first, $R_d^{(1)}$, is simply Hamming similarity scaled to $[0,1]$. The second label similarity, $R_d^{(2)}$, takes into account the prior probability p_l of a label l in the training set. Let L the set of possible labels, and L_1, L_2 two particular label sets. Then $R_d^{(2)}$ is defined as follows:

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

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

$$\begin{aligned} Q_1(y,x) &= \underset{w_l}{\text{owa}}(\{I(R(z,y), R_d(x,z)) | z \in N(y)\}); \\ Q_2(y,x) &= \underset{w_u}{\text{owa}}(\{T(R(z,y), R_d(x,z)) | z \in N(y)\}); \\ Q_3(y,x) &= \frac{Q_1(y,x) + Q_2(y,x)}{2}, \end{aligned} \tag{A.3}$$

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

For a choice of quality measure Q, FRONEC predicts the labels of the training instance with the highest quality. If there are several such

training instances, it predicts all labels that appear in at least half of the cases.

Data descriptors

The data descriptors in fuzzy-rough-learn have for the most part been discussed in Section 6.2, and will not be described in detail here. IF and SVM are wrappers for the implementations in scikit-learn, EIF is a wrapper for the implementation in the eif package[6].

We note that NND can be initialised with a weight vector, resulting in WNND (Definition 9.1), which is used internally to define FRNN classification.

fuzzy-rough-learn also implements *centre distance*, a generalisation of centroid distance (used in combination with the SAE preprocessor) to any measure of central tendency.

Feature preprocessors

FRFS

Fuzzy Rough Feature Selection (FRFS, Cornelis et al 2010) greedily selects features that induce the greatest increase in the size of the positive region, until it matches the size of the positive region with all features, or until the required number of features is selected.

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

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

$$R_a(x, y) = \max(1 - \frac{\left|x_a - y_a\right|}{\sigma_a}, 0). \tag{A.4}$$

LinearNormaliser

Unsupervised. Rescales the data by centring it on the specified measure of central tendency and/or dividing by the specified measure of dispersion. We provide a number of convenience functions (Table A.2), but the more general class allows users to define their own variants.

[6] `https://github.com/sahandha/eif`

Table A.2: Convenience functions for linear normalisers in fuzzy-rough-learn.

	Centre	Dispersion
`IQRNormaliser`	`midhinge`	`interquartile_range`
`MaxAbsNormaliser`		`maximum_absolute_value`
`RangeNormaliser`	`midrange`	`total_range`
`Standardiser`	`mean`	`standard_deviation`

SAE

Shrink Autoencoder (Section 6.2). Unsupervised, designed to make target data easier to separate from other data by data descriptors. Learns a $\lfloor \sqrt{m} \rfloor + 1$-dimensional latent representation of the target data, which is induced to shrink around the origin by the cost function, which balances reconstruction error against the Euclidean norm of the latent representation.

This is a reimplementation using the Keras and TensorFlow framework, based on the code provided by the original authors.[7]

The SAE neural network consists of six dense layers. The first three layers encode the data by linearly decreasing the number of features from m to $\lfloor \sqrt{m} \rfloor + 1$, while the last three layers use the same (tied) weights to decode the data again. We use the hyperbolic tangent activation function and Glorot uniform weight initialisation (Glorot & Bengio 2010).

The target data is split into a training set (80%) and a validation set (20%). Batch size is equal to 5% of the training set, with a maximum of 100. Validation accuracy is calculated every 5 epochs. The network is trained for 1000 epochs, or until the early stopping criterion is satisfied, which is the case when validation accuracy has not substantially increased for a number of epochs corresponding to 400 batches. The network is trained with the ADADELTA optimiser (Zeiler 2012) with an initial learning rate of 0.01.

VectorSizeNormaliser

Unsupervised. Projects all vectors onto the unit sphere. Typically used in natural language processing (NLP) for frequency counts, when only relative frequencies are deemed important. What is commonly called the cosine dissimilarity can then be obtained by measuring the squared Euclidean distance.

[7] `https://github.com/vanloicao/SAEDVAE`

Instance preprocessors

FRPS

Fuzzy Rough Prototype Selection (FRPS, Verbiest et al 2013; Verbiest 2014) uses upper and/or lower approximation membership as a quality measure to select instances. It follows the following steps:

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

There are a number of differences between the definition of FRPS by Verbiest et al 2013 and Verbiest 2014. In each case, the present implementation follows Verbiest 2014:

- While Verbiest et al (2013) use instances of all decision classes to calculate upper and lower approximations, Verbiest 2014 calculates the upper approximation membership of an instance using only instances of the same decision class, and its lower approximation membership using only instances of the other decision classes. This choice affects the length of the weight vector.
- Verbiest (2014) excludes each instance from the calculation of its own upper approximation membership, while Verbiest et al (2013) do not.
- Verbiest et al 2013 use linear weights, while Verbiest 2014 uses reciprocally linear weights.
- Verbiest et al 2013 define the similarity relation R by aggregating the per-attribute similarities R_a using the Łukasiewicz t-norm, whereas Verbiest 2014 recommends using the mean.
- In case of a tie between several best-scoring candidate prototype sets, Verbiest et al 2013 return the set corresponding to the median of the corresponding thresholds, while Verbiest 2014 returns the largest set (corresponding to the smallest threshold).

In addition, there are two implementation issues not addressed by either Verbiest et al (2013) or Verbiest (2014):

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

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

Regressors

FRNN

Fuzzy Rough Nearest Neighbour regression (FRNN, Jensen & Cornelis 2011). Let X be the training set and $d : X \longrightarrow \mathbb{R}$ the target value. For a test instance y, the fuzzy tolerance class $R_d(\cdot, \mathrm{NN}_i(y))$ of its ith nearest neighbour $\mathrm{NN}_i(y)$ is the fuzzy set defined as:

$$R_d(\cdot, \mathrm{NN}_i(y))(x) := \frac{|d(\mathrm{NN}_i(y)) - d(x)|}{\max(d[X]) - \min(d[X])}, \tag{A.5}$$

for any x in X. We use this to define a weight w_i, equal to the mean membership degrees of y in the strict upper and lower approximations of $R_d(\cdot, \mathrm{NN}_i(y))$:

$$w_i := (\overline{R_d(\cdot, \mathrm{NN}_i(y))}(y) + \underline{R_d(\cdot, \mathrm{NN}_i(y))}(y))/2. \tag{A.6}$$

The upper approximation membership is equal to at least the similarity between y and $\mathrm{NN}_i(y)$, but can be higher if there is a closer neighbour with a sufficiently similar target value. The lower approximation membership is equal to at least the distance to the closest neighbour $\mathrm{NN}_1(y)$, but can be higher if the target value of $\mathrm{NN}_1(y)$ is sufficiently similar to that of $\mathrm{NN}_i(y)$. In sum, greater weight is given to the values of closer neighbours, and to values of neighbours that reinforce each other by having similar target values.

Finally, these weights are used to predict a target value for y as the weighted sum of the target values of its k neighbours (default: 20):

$$d(y) := \sum_{i \leq k} w_i \cdot d(\mathrm{NN}_i(y)) / \sum_{i \leq k} w_i. \tag{A.7}$$

While the calculation of the upper approximation membership values can be restricted to the k nearest neighbours of y, the calculation of the lower approximation membership values in principle requires considering all instances in the training data. Therefore, FRNN regression does not at present scale well to large datasets.

A.6 Utilities

In this section, we discuss all the other functions included in fuzzy-rough-learn. These can be used as input for various hyperparameters of the algorithms discussed in the previous section.

Array functions

A collection of diverse utility functions for working with NumPy arrays. `first`, `last`, `least` and `greatest` return, respectively, the k first, last, least and greatest elements along the specified axis of an array. These functions are used internally by the functions `soft_head`, `soft_tail`, `soft_min`, `soft_max`, which additionally take a weight function (Subsection A.6), and use this to aggregate the k values along the specified axis. `soft_max` and `soft_min` are implementations of the weighted maximum and minimum (Definition 1.8).

`remove_diagonal` takes an n by n square matrix, and removes the diagonal to obtain an n by $n-1$ matrix. This can be used to remove comparisons of an instance with itself. `div_or` is simple division of an array, but wherever the division results in `NaN` (e.g. from $0/0$), a fallback value (default: 1) is substituted.

Dispersion measures

Functions that take a dataset, and return a one-dimensional array, indicating the dispersion of the dataset in each dimension. `total_range` is the difference between the largest and smallest values. `interquartile_range` ignores extreme values, and returns the difference between the third and first quartile. `maximum_absolute_value` is the maximum of the absolute values of the maximum and the minimum. `standard_deviation` is the Euclidean distance between the dataset and its mean.

Location measures

Functions that take a dataset, and return a one-dimensional array, indicating the relevant location of the dataset in each dimension. `minimum`, `maximum`, `mean` and `median` are thin wrappers for the corresponding NumPy functions, with the only specific difference that `NaN` values are

ignored. `midrange` is the mean of the minimum and the maximum, while `midhinge` is the mean of the first and third quartiles.

Neighbour search methods

Algorithms that construct a model which returns, for some $0 < k \leq n$, the k nearest neighbours of each query instance and their distances. The two algorithms `BallTree` (Omohundro 1989) and `KDTree` (Bentley 1975) are wrappers for the corresponding implementations in `scikit-learn`. By subclassing the abstract base class, users can substitute their own nearest neighbour search algorithms, such as Hierarchical Navigable Small World (HNSW, Section 5.1), which scales well to large datasets, but delivers only approximately accurate results.

Parametrisations

Utility functions that can be used to express a dependency of a value on another value. The concrete use case in this library is to let the number of nearest neighbours k depend on the number of available instances n. `multiple` multiplies n with the provided coefficient (typically a fraction). `log_multiple` multiplies $\log n$ with the provided coefficient.

T-norms

Triangular norms (Definition 0.3), expressed here as aggregation functions that take an array and reduce it along the indicated axis. The `goguen_t_norm` is the product. The `heyting_t_norm` (also known as the Gödel norm) is the minimum. Finally, the `lukasiewicz_t_norm` is equal to the sum of values, minus their count reduced by one.

Transformations

Functions to transform distance values in $[0, \infty]$ or signed distance values in $[-\infty, \infty]$ into similarity values in $[0, 1]$. `shifted_reciprocal` transforms distance values with:

$$x \longmapsto \frac{1}{1+x}. \tag{A.8}$$

`truncated_complement` transforms distance values with:

$$x \longmapsto \max(0, 1-x). \tag{A.9}$$

`contract` transforms signed distance values with:

$$x \longmapsto \frac{x}{2 \cdot (|x| + c)} + 0.5. \tag{A.10}$$

Vector size measures

Functions from real vector spaces to $[0, \infty]$ like norms. Can also be used as dissimilarity measures through application to $y - x$.

At present contains a single parametrisable function, `MinkowskiSize`, which encompasses the family of measures discussed in Chapter 2. The corresponding Minkowski mean and unrooted Minkowski size or mean can be obtained with the boolean parameters `scale_by_dimensionality` and `unrooted`.

Weights

Parametrisable functions that take a positive integer k and return a weight vector. Can be used as input for the soft head, maximum, minimum and tail functions (Subsection A.6).

In addition to the weight types in Table 1.1 (`ExponentialWeights`, `LinearWeights` and `ReciprocallyLinearWeights`), we have also included `ConstantWeights`:

$$\frac{1}{k}, \frac{1}{k}, \ldots, \frac{1}{k}, \tag{A.11}$$

and `QuantifierWeights` (Yager 1988):

$$\frac{q(\frac{1}{k})}{q(\frac{0}{k})}, \frac{q(\frac{2}{k})}{q(\frac{1}{k})}, \ldots, \frac{q(\frac{k}{k})}{q(\frac{k-1}{k})}, \tag{A.12}$$

where q is a non-decreasing, regular quantifier $[0, 1] \longrightarrow [0, 1]$.

Other: postprocessing functions

In addition to the utilities listed above, we also provide some functions to help with postprocessing classifier predictions.

`select_class` takes a two-dimensional array of records and class predictions, and returns a one-dimensional array that contains for each record the class with the highest score. In addition, an abstention threshold can be set, such that records with no class score above this threshold are labelled separately.

`discretise` takes an array of label scores and discretises these to either 0 or 1 depending on a threshold value. This can be used both for the scores produced by data descriptors and for multilabel classifiers.

`probabilities_from_scores` divides class scores by their sum, such that they sum to 1 for each record and can be used to calculate the AUROC score.

A.7 Future work

Like many software libraries, fuzzy-rough-learn is a permanent work-in-progress. Since it is still in its early stages of development, its shape has not crystallised into a definite form, and there are a number of unresolved issues. We are still looking for a way to better handle components, like the SAE preprocessor, that rely on optional dependencies. In addition, we are looking for a way to show how algorithms from other libraries, like approximative nearest neighbour search algorithms, can be used together with the algorithms in fuzzy-rough-learn, without turning those libraries into dependencies.

Another long-term goal is the inclusion of low-level algorithms, for which we currently use scikit-learn as a backend. Apart from making fuzzy-rough-learn more self-contained, this would also make it easier to expand our algorithms with new functionality.

In the near term, we hope to make pre- and post-processing more convenient to the user.

Not all of the results of this thesis have found their way into fuzzy-rough-learn yet. In particular, we want to add options for handling categorical and missing values, and to perform hyperparameter optimisation.

Appendix B

Datasets

In this appendix, we describe the datasets used in our experiments. Section B.1 covers the numerical datasets that we used in Parts I and III, Section B.2 the large datasets that we used in Part II, and Section B.3 the datasets with missing values that we used in Part IV. Finally, Section B.4 contains three corresponding tables with dataset statistics.

B.1 Numerical two- and multiclass datasets

For our experiments with multiclass and one-class classification in Chapter 3 and Part III of this thesis, we have selected fifty numerical datasets. 48 of these are from the UCI repository, while the *appendicitis* and *texture* datasets are, respectively, from the Keel repository and the ELENA project. This selection includes datasets with attributes that were originally represented with text labels, but which we converted to integers because they were ordinal in nature. In particular, we have represented all binary attributes with the values 0 and 1.

Table B.1 in Section B.4 lists the properties of each dataset and each decision class.

appendicitis **(Weiss & Kulikowski 1991)**

106 patients who were admitted to the emergency room of Overlook Hospital in Summit, New Jersey, and who subsequently underwent surgery for appendicitis, during three months in 1980 and six months in 1981-1982. The task is to predict whether the patients in fact had accute appendicitis (85) or not (21) based on 7 attributes, including the scores from laboratory tests.

This dataset is based on the data from Marchand et al (1983). It is unclear what exactly the attributes signify. Weiss & Kulikowski (1991) state that the dataset originally consisted of the results from eight laboratory tests, of which they removed one because it contained

missing values. However, Marchand et al (1983) only consider four laboratory tests, two of which are expressed both in absolute and in relative terms, as well as the temperature of the patient and the time since the onset of symptoms.

avila **(De Stefano et al 2018)**

20 867 sequences of four successive rows from the 12th century Avila bible. The classes are twelve different copyists who have produced different parts of the manuscript. The attributes consist of 10 physical characteristics of the text.

banknote **(Lohweg et al 2013)**

1372 images of banknotes, to be classified as genuine or counterfeit based on 4 attributes: the entropy, as well as the variance, skewness and excess kurtosis of the wavelet transformed image, as well as the entropy of the image.

Lohweg et al (2013) do not state how many records their dataset contains, and it is not clear whether class 0 corresponds to genuine banknotes and class 1 to counterfeit banknotes, or vice-versa.

biodeg **(Mansouri et al 2013)**

1055 molecules, 356 of which are biodegradable and 699 which are not, to be classified on the basis of 41 molecular descriptors.

breasttissue **(Estrela da Silva et al 2000)**

106 impedivity spectra of samples of breast tissue (collected from 64 patients undergoing breast surgery), to be classified as one of six types (three normal, three pathological) on the basis of 10 characteristics.

The dataset lacks three attributes ('BREAK', 'NOTCH' and 'SLOPE') used by Estrela da Silva et al (2000). A larger collection of 120 samples has been used in earlier work (Jossinet 1996) — 14 spectra were removed by Estrela da Silva et al (2000) during preprocessing since they were judged to exhibit poor tissue collection and/or data measurement.

coimbra **(Patrício et al 2018)**

116 people, of which 64 newly diagnosed breast cancer patients and 52 healthy volunteers, to be distinguished on the basis of 9 clinical attributes.

column **(Da Rocha Neto 2006)**

310 people, of which 60 patients diagnosed with disc hernia, 150 patients diagnosed with spondylolisthesis and 100 healthy volunteers, to be distinguished on the basis of 6 biomechanical attributes.

The dataset is based on data collected by Dr Henrique de Mota during a medical residence at the *Centre Medico-Chirurgical de Réadaptation des Massues* in Lyon in 2000–01.

debrecen **(B Antal & Hajdu 2014)**

1151 eye scans, to be classified as exhibiting signs of diabetic retinopathy or not. Based on the Messidor database, which consists of 1200 images. It is unclear why the number of records is fewer than 1200.

dermatology **(Güvenir et al 1998)**

366 patients with erythemato-squamous diseases, with 6 different diagnoses and 34 attributes encoding symptoms.

divorce **(Yöntem et al 2019)**

170 people from Turkey, either happily married (84) or divorced (86), to be classified on the basis of 54 questions about their marriage.

ecoli **(Horton & Nakai 1996)**

336 proteins from E.coli, to be classified by localisation site on the basis of 7 measurement scores. The problem is strongly imbalanced. Of the 8 classes, two only occur twice, and one only five times.

faults **(Buscema 1998)**

1941 steel plates with 7 different types of defect, to be classified based on 27 different physical characteristics. Buscema (1998) works with a reduced version of this dataset consisting of 1268 plates with 6 types of defect; the seventh class in the dataset consists of 673 'other faults'.

The data was collected by the Semeion Research Centre in Rome, under commission by the *Centro Sviluppo Materiali*.

foresttypes **(BA Johnson et al 2012)**

523 pixels from three aerial images of the same 13 by 12 km area of forested land in Ibaraki Prefecture, Japan, on three different dates (26 September 2010, 19 March 2011, 8 May 2011), to be classified as either 'Sugi', 'Hinoki' (two tree species), 'mixed broadleaf' or 'other' landcover. Each record consists of 27 attribute values: nine spectral values for three

different bands (green, red and near-infrared) on the three different dates, as well as eighteen similarity values with the Sugi and Hinoki classes. These similarity values are calculated by interpolating the spectral values of training records with, respectively, Sugi and Hinoki landcover, and subtracting the actual spectral values of each record.[1]

glass **(Evett & Spiehler 1987)**

214 samples of 6 different types of glass found on crime scenes, with 9 attributes: refractive index as well as levels of eight different oxides.

The six classes are labelled 1, 2, 3, 5, 6 and 7 in the dataset. Evett & Spiehler (1987) only distinguish between five classes: 'float window', 'non-float window', 'container', 'tableware' and 'headlamp', and the documentation of the dataset claims that these five classes correspond, respectively, to the numbers 1, 2, 3, 4 and 5. However, on the basis of the class frequencies (Bennett 1993) and the extract of the dataset reproduced by Evett & Spiehler (1987) in their Table 1, it is clear that 1 = 'float building window', 2 = 'non-float window', 3 = 'float vehicle window', 5 = 'container', 6 = 'tableware' and 7 = 'headlamp'.

In addition, there is at least one discrepancy between the extract of records reproduced by Evett & Spiehler (1987) and the values in the dataset. The first record in the extract has a value of 13.98 for sodium oxide, but 13.89 in the dataset.[2]

The data was originally provided by Mr B German of the Home Office Forensic Science Laboratory in Birmingham.

haberman **(Haberman 1976)**

306 female stage 1 or 2 breast cancer patients who had undergone radical mastectomy at the University of Chicago's Billings Hospital between 1958 and 1969 (Haberman (1976) says 1970, but there are no such patients in the dataset), to be classified as whether they survived for at least five years (225) or not (81), on the basis of the number of positive axillary nodes detected, the year of the operation and the patient's age at the time. Although Haberman (1976) says that there are 307 records, he lists only 306, and this is the number in the dataset that was handed down.

The data was originally provided by Dr Thomas Ferguson, who analysed in Ferguson & Meier (1976) the complete set of breast cancer patients who were first diagnosed between 1 January 1960 and 31

[1] The use of interpolated values on the basis of training records complicates the division of this dataset into training and test sets. I have applied cross-validation on the entire dataset, but in retrospect, I should perhaps only have used the nine attributes expressing pure spectral values.

[2] Furthermore, I have accidentally used the version of this dataset from the Keel repository, which, while not substantially different, is out of order and contains extraneous digits, most likely due to the handling of floating point numbers.

December 1969. If we restrict the dataset to the period 1960-69, we find a corresponding number of 243 stage 1 or 2 patients who underwent radical mastectomy. However, the number of patients that died within five years is higher in the dataset (60) than in Ferguson & Meier (1976) (53, as well as 1 'lost'). Perhaps this can be attributed to the fact that follow-up was only 'essentially complete' by the time of Ferguson & Meier (1976).

htru2 **(Lyon et al 2016)**

17 898 pulsar candidates collected during the High Time Resolution Universe Survey (South) using the Parkes Observatory in New South Wales, described by 8 attributes: the mean, standard deviation, excess kurtosis and skewness of the integrated pulse profile and the DM-SNR curve of the event in question. There are 1639 actual pulsars and 16 259 spurious events.

ilpd **(Ramana et al 2012)**

583 people from north east Andhra Pradesh, consisting of 416 liver patients and 167 non-liver patients, to be classified on the basis of 10 attributes, consisting of basic patient characteristics and relevant measurements. It is unclear how this dataset relates to the similar but larger (751 records) dataset used by the same authors in Babu et al (2010) and a number of other publications.

ionosphere **(Sigillito et al 1989)**

351 returns from multipulse patterns emitted by a radar to the ionosphere, collected in Goose Bay, Labrador, by the Space Physics Group of the Johns Hopkins University Applied Physics Laboratory. The aim is to distinguish 'good' returns from some kind of structure in the ionosphere (225) from 'bad' returns (126) on the basis of 34 attributes, corresponding to the real and imaginary parts of 17 complex numbers corresponding to the pulses that make up one emitted pattern.

As acknowledged by the author in the dataset documentation, the dataset contains one record more than the number given in the paper.

iris **(RA Fisher 1936)**

150 irises, 50 each of 'Iris Setosa', 'Iris Versicolor' and 'Iris Virginica', to be distinguished on the basis of 4 attributes: sepal and petal length and width. The flowers were collected and measured by Edgar Anderson. The Iris Setosa and Iris Versicolor samples were collected by E Anderson (1935) in the field running from Ile Verte to Trois Pistoles on the Gaspé Peninsula in Quebec, on the shore of the St Lawrence, apparently in the

early summer of 1935. Unwin & Kleinman (2021) argue convincingly that the data from the remaining class, Iris Virginica, corresponds to the sample taken from a single colony in Camden, Tennessee, in 1926, analysed by E Anderson (1928) along with many other samples.

As pointed out by Bezdek et al (1999), two of the records in the dataset as stored in the UCI repository contain mistakes. However, we have used the uncorrected version.

landsat **(Michie et al 1994)**

6435 pixels from a single 82 by 100 pixel aerial image of a section of Australia, to be classified as one of 6 types of land cover, on the basis of 36 attributes, corresponding to four spectral values of the pixel in question and its eight neighbours.

The image was purchased from NASA by the Australian Centre for Remote Sensing, and its use dates back to at least Lee & Richards (1984). The pixels of the image were annoted *in situ* by Karen Hall.

While the documentation warns against performing cross-validation on this dataset, there does not appear to be any reason for this.

leaf **(Silva 2013)**

340 leaves, belonging to 30 different species, characterised by 14 attributes, half of which describe the shape and the other half of which describe the texture of the leaf. Based on a collection of 443 photographs encompassing 40 different species, only species with simple leaves were included in the dataset.

letter **(Frey & Slate 1991)**

20 000 capital letter shapes in twenty different fonts, with various forms of added distortion, to be classified as one of 26 letters based on 16 physical attributes.

magic **(Bock et al 2003)**

19 020 Monte Carlo generated simulations of detection events by a Cherenkov gamma telescope. The task is to distinguish Cherenkov radiation caused by high energy gamma particles (12 332) from the hadronic showers caused by the impact of cosmic rays in the upper atmosphere (6688), based on 10 attributes describing the image produced by the event.

mfeat **(AK Jain et al 2000)**

2000 30 by 48 pixel images of handwritten digits extracted from nine Dutch utility maps, with 200 records for each of the 10 digits. This dataset has a very large number of 649 attributes, consisting of 76 Fourier coefficients of the character shapes, 216 profile correlations, 64 Karhunen-Love coefficients, 240 pixel averages of 2 by 3 pixel patches, 47 Zernike moments and 6 morphological features.

The same dataset appears to have been used first by Van Breukelen et al (1997). The documentation indicates that this was a slightly different dataset but the difference is not clear, and AK Jain et al (2000) refers back to Van Breukelen et al (1997) for its description of the data.

miniboone **(Neal 2006)**

130 065 Monte Carlo simulations of particle detection events: 36 499 electron neutrinos and 93 565 muon neutrinos, to be distinguished based on 50 particle identification variables.

This dataset was used by Radford Neal at the 2006 BIRS workshop. It was provided by Byron Roe, who used an earlier version with 250 890 records and 52 attributes in Roe et al (2005).

new-thyroid **(Coomans et al 1978)**

215 patients from the *Sint-Pietersziekenhuis* in Brussels with normal thyroid function (150), hypothyroidism (30) or hyperthyroidism (35), to be classified based on the results of 5 laboratory tests.

page-blocks **(Esposito et al 1995)**

5473 rectangular page segments, taken from 53 documents, to be classified as 'text' (4913), 'horizontal line' (329), 'graphic' (28), 'vertical line' (88) or 'picture' (115) based on 10 physical attributes.

The data used for this dataset may date back to at least Esposito et al (1990).

pop-failures **(Lucas et al 2013)**

540 simulations with POP2, the ocean simulation component of the CCSM4 climate model. The task is to predict whether the simulation crashes (46) or not (494), based on its 18 parameter values.

seeds **(Charytanowicz et al 2010)**

210 wheat kernels, 70 each of the varieties 'Kama', 'Rosa' and 'Canadian', to be distinguished on the basis of 7 physical attributes.

segment **(Utgoff & Brodley 1991)**

2310 pixels from seven outdoor images, 330 each from the 7 classes 'brickface', 'sky', 'foliage', 'cement', 'window', 'path' and 'grass', to be classified on the basis of 19 attributes of the 3 by 3 pixel patch centred on the relevant pixel.

This dataset is part of the Statlog project. The data was originally collected by Bruce Draper and Robert Collins at the Visions lab of the University of Massachusetts, who used an earlier version of this dataset in i.a. Draper et al (1989).

seismic-bumps **(Sikora 2011)**

2584 shifts of eight hours in two longwalls of the Mysłowice–Wesoła coal mine in Poland. The task is to predict hazardous shifts, defined as shifts in which a seismic tremor with energy greater than $10^4 J$ occurs, on the basis of information from previous shifts expressed in 18 attributes.

This classification task is already implicitly mentioned by Sikora & Wróbel (2010). There is a discrepancy between the dataset and the data described in Sikora (2011) — both have 170 hazardous shifts, but Sikora (2011) only has 1791 non-hazardous shifts, whereas the dataset has 2414. Our best guess is that the the additional non-hazardous shifts were removed during preprocessing.

sensorless **(Paschke et al 2013)**

58 509 runs of a demonstration model of a motor drive that is used in conveyor belts, 5319 runs for each of 11 different defect states, expressed by different configurations of components with various types of damage. The task is to distinguish these defect states on the basis of 48 attributes, corresponding to the mean, standard deviation, skewness and kurtosis of the first three instrinsic mode functions (IMF) and their residuals of two of the three electric phase currents of the synchronous motor (i.e. no external sensors are used, hence *sensorless*).

While variants of this dataset have been used in a number of different publications, it is unclear how exactly this dataset was generated, both in terms of the classes and in terms of records.

The experiments use three components with respectively three, three and four types of damage. Bayer et al (2013) and Lessmeier et al (2014) argue that on the basis of "design of experiment" (DoE) principles, it is sufficient to test 12 suitably chosen configurations of these three damage types to cover all possibilities, of which the first configuration corresponds to the absence of any damage (normal operating behaviour). Bayer et al (2013) then remove the twelfth class from the training set, to act as an 'unknown' state during testing. In contrast, Lessmeier et al (2014) propose to add three defect states corresponding to damage to

only one of each of the three components. Paschke et al (2013) start with 11 defect states and use one of these as the unknown state during testing, without explaining why there were only 11 states to begin with. Dörksen & Lohweg (2014) and Dörksen et al (2014) use only 10 classes. Consequently, it is possible that the dataset represents the situation of Bayer et al (2013) with either the first or the twelfth class removed, or the unexplained situation of Paschke et al (2013).

Similarly, the number of records in each class is somewhat mysterious. Bayer et al (2013) and Lessmeier et al (2014) both propose to let three types of operating behaviour vary at three different levels each, which can be covered by 9 different configurations on the basis of DoE principles, although Lessmeier et al (2014) then again add three more configurations expressing only one of each of the three types of operating behaviour. Paschke et al (2013) use 12 different configurations, without saying which. Only Dörksen & Lohweg (2014) and Dörksen et al (2014) explicitly give the number of records per class as 5319, without explaining how this number is arrived at. Its prime decomposition is $3^3 \cdot 197$, which suggests that it could represent 591 runs for each of 9 configurations, but 591 is a curiously specific number.

Lastly, while Dörksen et al (2014) announce the immininent submission of the dataset to the UCI repository, the version of the dataset used by them and Dörksen & Lohweg (2014) has 72 attributes. Presumably these correspond to all three electric phase currents, rather than just the first two.

shuttle **(Michie et al 1994)**

58 000 sets of measurements of the space radiator subsystem of the Space Shuttle, to be classified as one of seven states. Each record consists of 9 measurements from three sensors. According to the documentation, the first (integer-valued) attribute corresponds to 'time', but it is unclear what this means. The problem is highly imbalanced: class 1 contains 45 586 records, while classes 6 and 7 contain only 10 and 13 records, respectively.

This dataset is part of the Statlog project. It is unclear how exactly it relates to the earlier version used by Catlett (1991), containing 427 622 and 402 780 records from two flights (72 490 and 55 051 after removing duplicates). This original data was provided by Roger Burke of the Johnson Space Centre and the GHG Corporation, both in Houston.

skin **(Bhatt et al 2009)**

245 057 pixels from face images of a wide range of people drawn from the FERET and PAL databases, to be classified as skin (50 859) or non-skin (194 198) on the basis of their 3 RGB values.

There is no explanation for the different number of records in the dataset and Bhatt et al (2009) (only 51 444).

somerville **(Koczkodaj et al 2018)**

143 responses to the Somerville Happiness Survey conducted in 2015 in Somerville, Massachusetts. The task is to predict whether respondents respond to the question "How satisfied are you with Somerville as a place to live?" with a score of less than eight out of ten (66) or at least eight out of ten (77), based on their responses on a scale of five to 6 questions about aspects of life in Somerville.

sonar **(Gorman & Sejnowski 1988)**

208 response signals from casting a sonar signal at mines (111) or rocks (97). There are 60 attributes, corresponding to the energy within a certain frequency band, integrated over a certain period of time.

spambase

4601 emails received between June and July 1999 at Hewlett-Packard Labs in Palo Alto, consisting either of spam (1813) or non-spam (2788). There are 57 attributes, of which 48 correspond to specific word frequencies, 6 correspond to specific character frequencies and 3 characterise the use of capital letters.

While the documentation promises that an external publication is 'forthcoming', this dataset appears to have only been described (supposedly) in a technical report internal to Hewlett-Packard. The data was collected by Mark Hopkins during his internship under George Forman and Jaap Suermondt.

spectf **(Kurgan et al 2001)**

267 sets of two cardiac Single Proton Emission Computed Tomography (SPEC) images (at rest and under stress) of patients at the Medical College of Ohio judged to be normal (55) or abnormal (212). The 44 attributes are levels of perfusion for 22 regions of interest of the two images.

sportsarticles **(Hajj et al 2019)**

1000 English-language sports articles gathered from over fifty different websites, labelled as objective (658) or subjective (342) using Amazon's Mechanical Turk service. The 59 attributes consist of syntactic and morphological frequency counts, as well as a number of scores. Seven of these attributes are absent from Hajj et al (2019).

An earlier version of this dataset may have been used by Rizk & Awad (2012).

texture

5500 images of 11 different textures (with 500 records each), to be classified on the basis of 40 attributes, consisting of ten fourth order modified moments in four orientations.

This dataset was created by the *Laboratoire de Traitement d'Image et de Reconnaissance de Formes* of the *Institut National Polytechnique de Grenoble* for the Esprit project 6891 ELENA, and the Esprit working group 6620 ATHOS, based on the textures of Brodatz (1966).

transfusion **(IC Yeh et al 2009)**

748 blood donors from a university in Hsinchu, 178 of which donated blood in March 2007 and 570 of which did not, to be classified on the basis of 4 attributes characterising the time since their first and last donations and the total number and volume of past donations.

vehicle **(Michie et al 1994)**

846 silhouettes of 4 Corgi model cars (double decker bus, Chevrolet van, Saab 9000 and Opel Manta 400). The task is to classify the silhouettes on the basis of 18 physical attributes.

Part of the Statlog project. The dataset was created in 1986–87 by Paul Siebert at the Turing Institute in Glasgow with partial financing by Barr and Stroud Ltd.

The dataset originally contained 946 records, but 100 records were kept behind for validation purposes. The original dataset contained 240 silhouettes for each model except the Chevrolet van, 60 images at regular intervals of a full 360 degree rotation, at three different levels of elevation (one level of elevation was repeated twice). For the Chevrolet van, some angles were omitted at the highest elevation because the model would not fit inside the frame.

waveform **(Breiman et al 1984)**

5000 functions belonging to 3 different wave forms, characterised by their y values corresponding to 21 fixed-interval x values, with additional normally distributed noise.

wdbc **(Street et al 1992)**

569 images of aspirated breast tumor samples, to be classified as benign (357) or malignant (212) on the basis of 30 attributes, consisting of the

mean, standard deviation and pessimum of ten physical properties of the cell nuclei in the image.

This dataset appears to be based on a subset of the breast tumor aspirates used for the *wisconsin* dataset, but it uses a different set of attributes, derived from a single photograph of each sample (Street 1991).

wifi **(Narayanan et al 2016)**

2000 measurements of Wi-Fi signal strength with a mobile phone running Android at 4 different locations (500 measurements each) in an office in Pittsburgh: conference room, kitchen, indoor sports room and work areas. The location has to be predicted on the basis of the strength of the Wi-Fi signal from the 7 different routers in the office.

An earlier version of this dataset with only three of the four classes was used by Rohra et al (2016).

wilt **(BA Johnson et al 2013)**

4839 segments of a QuickBird multispectral aerial image of a 3 by 2.5 km area of forested land near Yonezawa, consisting of 261 diseased trees and 4578 other segments, to be classified using 5 attributes: the mean values of the green, red and near-infrared spectral bands, and the standard deviation and grey-level co-occurrence matrix (GLCM) mean of the panchromatic band.

wine **(Aeberhard et al 1992)**

178 wines from Piedmont with vintages between 1970–79. The classification task is to predict the cultivar, Barolo (59), Grignolino (71) or Barbera (48), based on 13 attributes describing their chemical composition.

Earlier versions of this dataset had 8 (Forina & Lanteri 1984) and 28 (Forina et al 1986) attributes. The dataset was apparently brought to Australia by Danny Coomans.

wisconsin **(Wolberg & Mangasarian 1990)**

699 fine-needle aspirates (FNA) of breast tumors, to be classified as benign (458) or malignant (241) on the basis of 9 cytological characteristics.

The original version of this dataset used by Wolberg & Mangasarian (1990) contained 369 records, the remainder after preprocessing of a larger collection of 482 FNAs, collected between 1984 and January 1989 by Dr William Wolberg at the Department of Surgery of the University of Wisconsin Clinical Sciences Center. According to its documentation, the dataset was then updated seven times up to November 1991 with batches of new records, while two records were removed from the original batch.

Thus, the version of the dataset in Bennett & Mangasarian (1990) had 487 records, and the version in Mangasarian & Wolberg (1990) 535.

The final version of the dataset doesn't appear to have been described explicitly in the literature by its creators. The closest match is probably Bennett (1992), who used 681 records, of which 442 benign and 239 malignant. The intermediate version of Bennett & Mangasarian (1991), with 566 records, may have formed the basis of the *wdbc* dataset, since its number of benign (354) and malignant (212) patients is very similar to that of *wdbc*. Note that the number of records in these last two versions of the wisconsin dataset does not correspond to the total number of records added in batches up to that point, so some form of preprocessing must have been applied.

wpbc **(Street et al 1996)**

141 patients who have undergone surgery for an invasive malignant breast cancer tumor with no evidence of distant metastases. The task is to predict recurrence within two years (29) or not (112) on the basis of 32 attributes.

The dataset as provided contains 198 records. As suggested by the documentation we preprocessed this by selecting only patients with recurrence within two years and non-recurrence for at least two years.[3]

This dataset is based on the *wdbc* dataset, and it uses the same attributes, complemented with tumor size and lymph node status. The earlier version used by Bennett & Mangasarian (1992) had 122 records and 11 attributes, the version used by Mangasarian et al (1994) had 175 records, the version used by Street (1994) and Street et al (1995) had 187 records, while the version used by Wolberg et al (1995) consisted of 190 records.

yeast **(Horton & Nakai 1996)**

1484 yeast proteins to be classified by localisation site on the basis of 8 measurement scores. The problem is quite imbalanced. Of the 10 classes, one only occurs five times.

B.2 Large datasets

In this section, we describe the large datasets from the UCI repository that we have used in Part II of this thesis. Table B.2 in Section B.4 summarises the properties of these datasets.

[3]It would probably have been correcter to also include patients with recurrence after two years, and count these as non-recurrent — I may have misinterpreted the documentation on this point.

hepmass **(Baldi et al 2016)**

10 500 000 Monte Carlo simulations of particle collisions, 5 250 124 of which involve an exotic new particle and 5 249 876 of which do not. There are 28 attributes: 22 low-level kinematic features, five high-level features expressing the invariant masses of intermediate particles, and the mass of the potential exotic particle (the records are drawn uniformly from five mass levels).

higgs **(Baldi et al 2014)**

11 000 000 Monte Carlo simulations of particle collisions with identical decay products, 5 829 123 of which produce a Higgs boson and 5 170 877 of which do not. There are 28 attributes: 21 low-level kinematic features and seven high-level features expressing the invariant masses of intermediate particles.

poker-hand **(Cattral et al 2002)**

1 025 010 hands of five playing cards. The task is to classify the hand as one of 10 possible poker hands, based on the 10 suits and ranks of the cards.

This dataset is strongly imbalanced, the classes 'Royal flush' and 'Straight flush' only contain 8 and 17 records, respectively.

susy **(Baldi et al 2014)**

5 000 000 Monte Carlo simulations of particle collisions with identical detectable decay products, 2 287 827 of which produce supersymmetric particles and 2 712 173, of which do not. There are 18 attributes: 8 low-level kinematic features and 10 high-level features (unlike the *higgs* and *hepmass* datasets, these do not express the invariant masses of intermediate particles, since there are too many unknowns to reconstruct these).

Baldi et al (2014) appears to use a slightly different set of 17 attributes (5 low-level and 12 high-level). It appears to be missing the low-level features 'lepton 1 eta', 'lepton 1 phi', 'lepton 2 eta', 'lepton 2 phi', 'missing energy magnitude' and 'missing energy phi'. Instead, it has the additional low-level features 'Sum jet p_T', 'Missing trans. mom (GeV)' and 'N jets', as well as the additional high-level features 'β_R' and 'β_{R+1}'.

B.3 Datasets with missing values

For our experiments with missing values, we have selected twenty datasets from the UCI repository with missing values, consisting of both

numerical and categorical attributes.[4] These datasets are quite varied — they cover a number of different domains and contain between 155 and 76 000 records, between 4 and 590 attributes, between 2 and 21 decision classes and missing value rates between 0.0032 and 0.43.

We have preprocessed these datasets in the following manner. When it was clear from the description that an attribute was categorical, we treated it as such, even if it was originally represented with numerals. Conversely, where the possible values of an attribute admitted a semantic order, we encoded them numerically. We have removed attributes that were labelled non-informative by the accompanying documentation, as well as identifiers and alternative target values.

Table B.3 in Section B.4 summarises the properties of these datasets, including their missing value rate.

adult **(Kohavi 1996)**

48 842 1994 census records of American adults. The task is to predict whether each person earns more than $50 000 per year (11 687) or not (37 155), based on 13 census questions.

We have removed the 'fnlwgt' attribute, the weight that needs to be applied to each record to obtain a representative socio-economic sample within each US state.

The version of the dataset used by Kohavi (1996) has 45 222 records — these are the records without missing values.

The data was extracted from the 1994 census database by Barry Becker.

agaricus-lepiota **(Schlimmer 1987)**

8124 mushrooms from the *Agaricus* and *Lepiota* families, to be classified as edible (4208) or poisonous (3916) on the basis of 22 physical characteristics.

It is unclear whether the missing values, all in 'stalk-root', represent actually missing information, or a missing stalk-root.

This dataset was created on the basis of the information provided by Lincoff (1981). The version of the dataset used by Schlimmer (1987) contained only 3078 mushrooms (from 23 species). Although Schlimmer (1987) claims 23 attributes, he lists only 22, so this is most likely a mistake (possibly due to the number of mushroom species).

[4] These datasets are available at `https://cwi.ugent.be/~oulenz/datasets/missing-values.tar.gz`.

aps-failure **(Ferreira Costa & Nascimento 2016)**

76 000 component failures in Scania trucks. The task is to predict whether a specific component in the air pressure system (APS) has failed (1375) or some other component (74 625), based on 170 measurements.

This dataset was provided by Tony Lindgren of the Department of Computer and Systems Sciences at Stockholm University and Jonas Biteus at Scania for the industrial challenge at the 15th International Symposium on Intelligent Data Analysis (IDA) in 2016.

arrhythmia **(Güvenir et al 1997)**

452 patients in 13 classes, indicating the presence and type of arrhythmia. The 279 attributes consist of the age, sex, height and weight of the patients as well as a large number of characteristics of their ECG recordings.

This dataset is strongly imbalanced: 245 patients have no arrhythmia, while there are five classes with fewer than 10 records.

bands **(Evans & D Fisher 1994)**

540 rotogravure printing cylinders, displaying banding (228) or not (312), to be classified on the basis of 34 attributes describing the printing press and its use.

We have preprocessed this dataset by removing the 'timestamp', 'cylinder number', 'customer' and 'job number' attributes, which do not really form categories, as well as the 'ink color' attribute, which only has one value. None of these attributes are used by Evans & D Fisher (1994).

There are additional differences with respect to the variant of this dataset used by Evans & D Fisher (1994). That variant does not have the 'cylinder division', 'press type', 'paper mill location', 'callper' and 'roller durometer' attributes, but does have additional 'blade oscillation' and 'basis weight' attributes, for a total of 31. It also covers a shorter time period than is contained in the final version of the dataset.

ckd **(Rubini & Eswaran 2015)**

400 people, 250 of which with chronic kidney disease (CKD), 150 without, to be classified on the basis of 24 measurements.[5]

The origin of this dataset is not explained by Rubini & Eswaran (2015).

[5]The provided file of this dataset contains a number of spurious tab characters that escaped my attention and which may have had a minor effect on the experiments.

crx **(Quinlan 1987)**

690 credit card applications, 307 of which were approved and 383 of which were not.

The data was provided by a large bank. The meaning of the 15 attributes is confidential.

dress-sales

500 dresses offered for sale by AliExpress between August and October 2013, recommended (210) or not (290) on the basis of 12 properties.

We have preprocessed this dataset by eliminating spelling variations and interpreting certain values as missing values.

This dataset was created by Muhammad Usman and Adeel Ahmed at the Air University in Islamabad, who do not appear to have used it in any publication. It is unclear what the meaning of the two classes is. The documentation suggests that there is a connection with the number of sales of each dress, which are also provided, but there doesn't appear to be any direct link.

exasens **(Soltani Zarrin et al 2020)**

399 patients of the medical clinic in Borstel, near Sülfeld, Germany, and healthy controls, to be classified as healthy (160) or having chronic obstructive pulmonary disease (COPD, 79), asthma (80) or a respiratory infection (80) based on 7 attributes: age, gender, smoking status and four values expressing saliva permittivity.

The dataset used by Soltani Zarrin et al (2020) only contains the healthy and COPD patients.

hcc **(Santos et al 2015)**

165 hepatocellular carcinoma (HCC) patients of the Coimbra University Hospital. The task is to predict 1-year survival (102) or not (63) on the basis of 49 attributes expressing risk factors, comorbidies and a range of tests.

heart-disease **(Detrano et al 1989)**

1611 patients from five hospitals, with (903) or without (708) heart disease, defined as more than 50% narrowing of any major bloodvessel. There are 14 attributes, including the hospital, the age and sex of the patient and the results of a number of tests.

We have preprocessed this dataset by reducing the id-attribute to only identify the source hospital.

This dataset has a complicated history. The data was collected by Dr Robert Detrano at the Cleveland Clinic and at the Veterans Administration Medical Center in Long Beach, Dr Andras Janosi at the Hungarian Institute of Cardiology in Budapest, Dr William Steinbrunn at the University Hospital in Zürich, and Dr Matthias Pfisterer at the University Hospital in Basel.

On the basis of the id-attribute, it is possible to identify several batches of records: one batch from Cleveland (303 records), two batches from Budapest (428 and 351 records), two batches from Long Beach (200 and 201 records), one batch from Zürich (58 records) and one batch from Basel (73 records). On the basis of these numbers, we can deduce that the dataset used by Detrano et al (1989) does not contain the second batches from Budapest and Long Beach, nor three records at the end of the first batch from Budapest. It contains 85 records from Basel, which means that 12 records are missing. The pilot study by Detrano et al (1984) used only 154 patients from Cleveland.

The first batch of Long Beach records appears to have three duplicate pairs of records, with the same or nearly the same name, social security number, age, sex and other attribute values. Given that four of the clinical attribute values are slightly different, it is unclear whether these are truly duplicate records or separate examinations of the same patients. Nevertheless, we have decided to remove the second record of each pair during preprocessing.

hepatitis **(Efron & Gong 1981)**

155 chronic hepatitis patients, 33 of which died and 122 of which lived. There are 19 attributes, consisting of patient characteristics, symptoms and test results.

The data was collected by Dr Peter Gregory.

horse-colic **(McLeish & Cecile 1990)**

368 horses with colic presented to the Ontario Veterinary College hospital in Guelph. The task is to predict whether (in retrospect) the lesion was surgical (232) or not (136), based on 20 symptoms and measurements.

We have preprocessed this dataset by deleting two non-informative attributes and five attributes that are alternative prediction targets (according to the documentation). It is not clear whether McLeish & Cecile (1990) used the exact same attribute set.

mammographic-masses **(Elter et al 2007)**

961 full-field digital mammograms, to be classified as benign (516) or malignant (445) on the basis of 4 attributes: the patient's age and the shape, margin and density of the masses.

The data was collected at the Institute of Radiology of the University of Erlangen-Nuremberg between 2003 and 2006.

mi **(Golovenkin et al 2020)**

1700 patients with myocardial infarction (MI). The 8 classes describe whether the patient died, and if so, what the cause of death was. The 111 attributes consist of patient characteristics, comorbidies, test outcomes and symptoms.

This dataset is very imbalanced, as the class with surviving patients contains 1429 records.

The data was collected at the Krasnoyarsk Interdistrict Clinical Hospital between 1992 and 1995. Earlier versions of this dataset were used by e.g. Rossiev et al (1995).

nomao **(Candillier & Lemaire 2012)**

34 465 pairs of place records. The task is to predict whether the two records refer to the same place (24 621) or to different places (9844), on the basis of 118 attributes expressing the similarity or difference of the attributes of the two original records.

The data for this dataset was provided by Nomao for the 'Nomao Challenge' of the 2012 Active Learning in Real-world Applications ECML-PKDD Workshop.

primary-tumor **(Cestnik et al 1987)**

339 cancer patients. The task is to identify the site of the primary tumor out of 21 possibilities, based on 17 attributes. Most attributes are boolean and refer to body parts. Their meaning is slightly unclear, it is possible that they refer to the locations that the cancer has spread to.

Many of the classes are very small. There are six classes with fewer than 5 records. In fact, by design the number of classes is 22, but one class is empty.

The data was collected at the University Medical Centre in Ljubljana by M Zwitter and M Soklic.

secom **(McCann et al 2008)**

1567 produced wafers at a production line of a semiconductor fabrication plant, 1463 of which passed testing and 104 of which failed, to be classified on the basis of 590 signals.

This dataset was created for the 'Causality Challenge' of the 2008 NIPS Workshop on Causality.

soybean **(Michalski & Chilausky 1980)**

683 soybean plants, displaying 19 different diseases, to be classified on the basis of 35 symptoms.

Michalski & Chilausky (1980) omitted the four smallest classes, using only 630 records.

thyroid0387 **(Quinlan et al 1986)**

9172 thyroid patients at St Vincent's Hospital in Sydney between August 1984 and January 1987. The task is to predict the diagnosis out of 18 classes, based on 29 patient characteristics and test scores.

This dataset is strongly imbalanced: 6771 patients have no diagnosis, while there are three classes with fewer than ten records.

We have had to preprocess this dataset because a small number of records belonged to multiple classes. When one diagnosis was indicated as being more likely than another, we retained the more likely diagnosis. Otherwise, we resolved this by retaining the most specific class.

The variant of this dataset used by Quinlan et al (1986) only had 3066 records, and didn't have the 'I131 treatment', 'hypopituitary', 'psych' and 'referral source' attributes.

B.4 Dataset statistics

In this section, we summarise the properties of the datasets used in this thesis. Table B.1 lists the properties of the numerical datasets described in Section B.1, as well as the properties each decision class. Table B.2 contains the large dataset described in Section B.2. Finally, Table B.3 summarises the properties of the incomplete datasets described in Section B.3, including their missing value rate.

Table B.1: Numerical real-life datasets from the UCI repository. m: number of attributes; n: number of records; s: sparsity.

Dataset	m	Class	n	s
appendicitis	7	0	85	0.05
		1	21	0.12
avila	10	A	8572	0.08
		B	10	0.57
		C	206	0.14
		D	705	0.11
		E	2190	0.09
		F	3923	0.09
		G	893	0.13
		H	1039	0.12

Continued on next page

Table B.1: Numerical real-life datasets from the UCI repository. m: number of attributes; n: number of records; s: sparsity.

Dataset	m	Class	n	s
		I	1663	0.10
		W	89	0.19
		X	1044	0.09
		Y	533	0.08
banknote	4	0	762	0.01
		1	610	0.00
biodeg	41	NRB	699	0.46
		RB	356	0.51
breasttissue	9	adi	22	0.06
		car	21	0.05
		con	14	0.08
		fad	15	0.07
		gla	16	0.12
		mas	18	0.06
coimbra	9	1	52	0.04
		2	64	0.04
column	6	DH	60	0.03
		NO	100	0.02
		SL	150	0.01
debrecen	19	0	540	0.28
		1	611	0.25
dermatology	34	1	111	0.70
		2	60	0.79
		3	71	0.74
		4	48	0.83
		5	48	0.79
		6	20	0.75
divorce	54	0	86	0.69
		1	84	0.55
ecoli	7	cp	143	0.33
		im	77	0.34
		imL	2	0.64
		imS	2	0.64
		imU	35	0.36
		om	20	0.38
		omL	5	0.46
		pp	52	0.36
faults	27	Bumps	402	0.14
		Dirtiness	55	0.21
		K_Scatch	391	0.22
		Other_Faults	673	0.13
		Pastry	158	0.17
		Stains	72	0.31
		Z_Scratch	190	0.19

Continued on next page

Table B.1: Numerical real-life datasets from the UCI repository. m: number of attributes; n: number of records; s: sparsity.

Dataset	m	Class	n	s
foresttypes	27	d	159	0.04
		h	86	0.09
		o	83	0.04
		s	195	0.06
glass	9	1	70	0.23
		2	76	0.21
		3	17	0.27
		5	13	0.32
		6	9	0.43
		7	29	0.29
haberman	3	1	225	0.22
		2	81	0.16
htru2	8	0	16259	0.00
		1	1639	0.00
ilpd	10	1	414	0.17
		2	165	0.19
ionosphere	34	b	126	0.32
		g	225	0.13
iris	4	Iris-setosa	50	0.29
		Iris-versicolor	50	0.17
		Iris-virginica	50	0.19
landsat	36	1	1533	0.10
		2	703	0.13
		3	1358	0.14
		4	626	0.12
		5	707	0.09
		7	1508	0.12
leaf	14	1	12	0.12
		2	10	0.16
		3	10	0.13
		4	8	0.14
		5	12	0.08
		6	8	0.13
		7	10	0.11
		8	11	0.11
		9	14	0.08
		10	13	0.09
		11	16	0.07
		12	12	0.09
		13	13	0.13
		14	12	0.10
		15	10	0.11
		22	12	0.09
		23	11	0.09

Continued on next page

Table B.1: Numerical real-life datasets from the UCI repository. m: number of attributes; n: number of records; s: sparsity.

Dataset	m	Class	n	s
		24	13	0.10
		25	9	0.13
		26	12	0.12
		27	11	0.15
		28	12	0.14
		29	12	0.11
		30	12	0.10
		31	11	0.10
		32	11	0.12
		33	11	0.13
		34	11	0.10
		35	11	0.13
		36	10	0.10
letter	16	A	789	0.35
		B	766	0.35
		C	736	0.31
		D	805	0.35
		E	768	0.33
		F	775	0.27
		G	773	0.32
		H	734	0.39
		I	755	0.49
		J	747	0.28
		K	739	0.30
		L	761	0.28
		M	792	0.32
		N	783	0.32
		O	753	0.41
		P	803	0.30
		Q	783	0.30
		R	758	0.31
		S	748	0.32
		T	796	0.27
		U	813	0.34
		V	764	0.35
		W	752	0.32
		X	787	0.40
		Y	786	0.31
		Z	734	0.37
magic	10	g	12332	0.00
		h	6688	0.00
mfeat	649	0	200	0.30
		1	200	0.30
		2	200	0.29
		3	200	0.29
		4	200	0.29
		5	200	0.28

Continued on next page

Table B.1: Numerical real-life datasets from the UCI repository. m: number of attributes; n: number of records; s: sparsity.

Dataset	m	Class	n	s
		6	200	0.29
		7	200	0.31
		8	200	0.27
		9	200	0.29
miniboone	50	0	93565	0.03
		1	36499	0.02
new-thyroid	5	1	150	0.08
		2	35	0.13
		3	30	0.11
page-blocks	10	1	4913	0.04
		2	329	0.19
		3	28	0.09
		4	88	0.39
		5	115	0.06
pop-failures	18	0	46	0.02
		1	494	0.00
seeds	7	1	70	0.03
		2	70	0.03
		3	70	0.03
segment	19	1	330	0.17
		2	330	0.17
		3	330	0.20
		4	330	0.16
		5	330	0.26
		6	330	0.16
		7	330	0.16
seismic-bumps	18	0	2414	0.61
		1	170	0.52
sensorless	48	1	5319	0.00
		2	5319	0.00
		3	5319	0.00
		4	5319	0.00
		5	5319	0.00
		6	5319	0.00
		7	5319	0.00
		8	5319	0.00
		9	5319	0.00
		10	5319	0.00
		11	5319	0.00
shuttle	9	1	45586	0.30
		2	50	0.37
		3	171	0.27
		4	8903	0.31
		5	3267	0.27

Continued on next page

Table B.1: Numerical real-life datasets from the UCI repository. m: number of attributes; n: number of records; s: sparsity.

Dataset	m	Class	n	s
		6	10	0.31
		7	13	0.49
skin	3	1	50859	0.02
		2	194198	0.03
somerville	6	0	66	0.40
		1	77	0.46
sonar	60	M	111	0.03
		R	97	0.03
spambase	57	0	2788	0.81
		1	1813	0.72
spectf	44	0	55	0.14
		1	212	0.08
sportsarticles	59	objective	635	0.38
		subjective	365	0.28
texture	40	2	500	0.00
		3	500	0.00
		4	500	0.00
		6	500	0.00
		7	500	0.00
		8	500	0.00
		9	500	0.00
		10	500	0.00
		12	500	0.00
		13	500	0.00
		14	500	0.00
transfusion	4	0	570	0.19
		1	178	0.18
vehicle	18	bus	218	0.15
		opel	212	0.10
		saab	217	0.09
		van	199	0.11
waveform	21	0	1657	0.01
		1	1647	0.01
		2	1696	0.01
wdbc	30	B	357	0.01
		M	212	0.01
wifi	7	1	500	0.13
		2	500	0.11
		3	500	0.14
		4	500	0.14
wilt	5	n	4578	0.00
		w	261	0.01

Continued on next page

Table B.1: Numerical real-life datasets from the UCI repository. m: number of attributes; n: number of records; s: sparsity.

Dataset	m	Class	n	s
wine	13	1	59	0.08
		2	71	0.07
		3	48	0.08
wisconsin	9	2	444	0.73
		4	239	0.31
wpbc	32	N	110	0.04
		R	28	0.06
yeast	8	CYT	463	0.36
		ERL	5	0.47
		EXC	35	0.44
		ME1	44	0.42
		ME2	51	0.41
		ME3	163	0.38
		MIT	244	0.38
		NUC	429	0.33
		POX	20	0.38
		VAC	30	0.41

Table B.2: Large real-life datasets from the UCI repository. n: number of records; m: number of attributes; c: number of classes.

Dataset	n	Type	m	c
hepmass	10 500 000	Numerical	28	2
higgs	11 000 000	Numerical	28	2
poker-hand	1 025 010	Categorical	10	10
susy	5 000 000	Numerical	18	2

Table B.3: Real-life classification datasets with missing values from the UCI repository for machine learning. n: number of records; c: number of classes.

Dataset	n	c	Attributes			Missing value rate		
			Num	Cat	Total	Num	Cat	Total
adult	48 842	2	5	8	13	0.0	0.017	0.010
agaricus-lepiota	8124	2	2	20	22	0.0	0.015	0.014
aps-failure	76 000	2	170	0	170	0.083		0.083
arrhythmia	452	13	279	0	279	0.0032		0.0032
bands	540	2	19	15	34	0.054	0.054	0.054
ckd	400	2	24	0	24	0.11		0.11
crx	690	2	6	9	15	0.0060	0.0068	0.0065
dress-sales	500	2	3	9	12	0.20	0.19	0.19
exasens	399	4	7	0	7	0.43		0.43
hcc	165	2	49	0	49	0.10		0.10
heart-disease	1611	2	13	1	14	0.18	0.0	0.17
hepatitis	155	2	19	0	19	0.057		0.057
horse-colic	368	2	19	1	20	0.25	0.39	0.26
mammographic-masses	961	2	2	2	4	0.042	0.041	0.042
mi	1700	8	111	0	111	0.085		0.085
nomao	34 465	2	89	29	118	0.38	0.37	0.38
primary-tumor	339	21	16	1	17	0.029	0.20	0.039
secom	1567	2	590	0	590	0.045		0.045
soybean	683	19	22	13	35	0.099	0.096	0.098
thyroid0387	9172	18	7	22	29	0.22	0.0015	0.055

Appendix C

Full results

In this chapter, we list the results of some of our experiments in greater detail.

C.1 One-class classification with default hyperparameter values

Table C.1 is part of the experiment in Chapter 7. It contains the mean AUROC of every data descriptor per one-class classification problem, with default hyperparameter values determined by the leave-one-dataset-out scheme.

Table C.1: 5-fold cross validation AUROC of each data descriptor with leave-one-dataset-out optimal hyperparameter values (where applicable).

Dataset	Target class	NND	LNND	LOF	MD	SVM	IF	EIF	SAE	ALP
appendicitis	0	0.741	0.644	0.744	0.691	0.665	**0.765**	0.734	0.518	0.721
	1	0.688	0.741	0.787	0.683	0.686	**0.799**	0.787	0.603	0.780
avila	A	**0.914**	0.759	0.844	0.622	0.743	0.599	0.596	0.582	0.891
	B	**1.000**	**1.000**	**1.000**	0.844	1.000	0.910	**1.000**	0.894	**1.000**
	C	**0.964**	0.825	0.944	0.591	0.923	0.631	0.706	0.566	0.959
	D	**0.974**	0.887	0.959	0.692	0.952	0.665	0.688	0.685	0.966
	E	**0.967**	0.828	0.934	0.567	0.857	0.649	0.672	0.569	0.953
	F	**0.936**	0.785	0.878	0.698	0.804	0.698	0.697	0.634	0.912
	G	**0.987**	0.889	0.962	0.789	0.959	0.762	0.844	0.803	0.981
	H	0.973	0.884	0.951	0.871	0.929	0.840	0.847	0.739	**0.974**
	I	**0.996**	0.915	0.980	0.799	0.994	0.837	0.852	0.900	0.982
	W	**0.969**	0.788	0.954	0.958	0.896	0.627	0.801	0.920	0.950
	X	**0.983**	0.876	0.972	0.732	0.960	0.699	0.678	0.751	0.981
	Y	**0.998**	0.948	0.990	0.885	0.994	0.840	0.861	0.808	0.995
banknote	0	0.998	0.946	0.993	**0.999**	0.997	0.922	0.950	0.985	0.996

Continued on next page

Table C.1: 5-fold cross validation AUROC of each data descriptor with leave-one-dataset-out optimal hyperparameter values (where applicable).

Dataset	Target class	NND	LNND	LOF	MD	SVM	IF	EIF	SAE	ALP
	1	**1.000**	0.993	0.999	0.994	0.997	0.884	0.941	0.999	1.000
biodeg	NRB	**0.597**	0.557	0.502	0.496	0.558	0.443	0.587	0.516	0.538
	RB	0.852	0.784	0.850	**0.870**	0.835	0.866	0.758	0.847	0.866
breasttissue	adi	0.925	0.904	0.920	0.893	0.926	0.816	0.881	**0.947**	0.928
	car	0.926	0.939	**0.972**	0.924	0.945	0.886	0.910	0.830	0.962
	con	0.888	0.858	0.933	0.836	0.874	0.778	0.817	0.841	**0.936**
	fad	**0.896**	0.847	0.851	0.862	0.888	0.814	0.858	0.822	0.854
	gla	0.922	0.866	0.918	**0.939**	0.904	0.910	0.912	0.831	0.919
	mas	0.737	0.739	0.696	**0.741**	0.715	0.728	0.661	0.672	0.707
coimbra	1	**0.743**	0.691	0.714	0.665	0.732	0.639	0.706	0.632	0.714
	2	0.424	0.417	0.479	**0.587**	0.520	0.548	0.441	0.561	0.491
column	DH	0.875	0.837	0.856	0.874	**0.888**	0.846	0.857	0.801	0.876
	NO	0.875	0.875	**0.881**	0.869	0.870	0.871	0.880	0.866	0.880
	SL	0.742	0.749	0.813	0.712	**0.847**	0.817	0.772	0.523	0.846
debrecen	0	0.619	0.642	0.636	**0.768**	0.631	0.678	0.622	0.660	0.643
	1	0.404	0.448	0.432	0.370	0.386	0.400	0.388	0.415	**0.450**
dermatology	1	0.995	0.953	0.985	0.955	**1.000**	0.985	0.992	0.981	0.996
	2	0.952	0.938	0.953	0.937	0.959	0.878	0.952	0.930	**0.961**
	3	0.998	0.990	0.997	**0.999**	0.999	0.996	0.997	0.958	0.997
	4	0.985	0.974	**0.986**	0.961	0.976	0.883	0.964	0.972	0.986
	5	0.997	0.982	**0.999**	0.975	0.997	0.955	0.996	0.948	**0.999**
	6	0.993	0.981	**0.998**	0.997	0.995	0.962	0.932	0.734	**0.998**
divorce	0	**1.000**	0.995	**1.000**	0.992	0.998	**1.000**	0.997	0.936	**1.000**
	1	0.888	0.873	0.852	0.889	0.900	**0.999**	0.994	0.819	0.878
ecoli	cp	0.968	0.966	**0.979**	0.974	0.966	0.972	0.969	0.951	0.978
	im	0.869	0.831	0.886	0.894	0.885	**0.906**	0.890	0.730	0.891
	imU	0.908	0.892	0.910	0.910	0.913	**0.935**	0.933	0.758	0.916
	om	0.949	0.963	0.972	0.967	0.962	0.972	0.960	0.562	**0.979**
	omL	0.985	**0.991**	0.983	0.985	0.985	0.632	0.655	0.852	0.988
	pp	0.925	0.931	0.951	0.928	0.938	0.938	0.936	0.813	**0.952**
faults	Bumps	**0.841**	0.777	0.823	0.826	0.834	0.821	0.792	0.777	0.833
	Dirtiness	**0.947**	0.824	0.888	0.931	0.920	0.886	0.876	0.930	0.944
	K_Scatch	**0.986**	0.804	0.892	0.983	0.983	0.972	0.972	0.967	0.904
	Other_Faults	**0.662**	0.585	0.632	0.600	0.608	0.603	0.597	0.620	0.631
	Pastry	0.854	0.779	0.815	**0.870**	0.837	0.852	0.786	0.840	0.840
	Stains	0.997	0.995	**0.997**	0.994	0.986	0.994	0.994	0.981	0.997
	Z_Scratch	0.972	0.932	0.936	0.958	**0.975**	0.915	0.946	0.922	0.971
foresttypes	d	0.838	0.836	0.870	0.844	0.868	0.880	0.818	0.776	**0.880**
	h	0.936	0.919	0.938	0.898	0.941	0.937	**0.950**	0.885	0.937
	o	0.690	0.736	0.732	0.698	0.739	**0.879**	0.770	0.780	0.713
	s	0.931	0.907	0.941	0.910	0.936	0.940	0.928	0.825	**0.944**
glass	1	0.803	0.820	0.874	0.797	0.840	0.800	0.795	0.813	**0.893**

Continued on next page

Table C.1: 5-fold cross validation AUROC of each data descriptor with leave-one-dataset-out optimal hyperparameter values (where applicable).

Dataset	Target class	NND	LNND	LOF	MD	SVM	IF	EIF	SAE	ALP
	2	0.689	0.641	0.708	0.659	0.674	0.637	0.623	0.568	**0.718**
	3	0.719	0.715	0.689	**0.845**	0.806	0.661	0.734	0.753	0.776
	5	0.641	0.571	0.611	0.702	0.762	0.704	0.610	0.385	**0.763**
	6	0.668	0.563	0.571	**0.971**	0.599	0.456	0.524	0.654	0.829
	7	0.801	0.814	0.800	0.738	**0.865**	0.827	0.794	0.750	0.839
haberman	1	0.679	0.642	0.625	0.616	0.652	0.670	**0.695**	0.561	0.608
	2	0.404	0.458	0.464	0.515	**0.526**	0.456	0.438	0.507	0.469
htru2	0	**0.953**	0.918	0.952	0.944	0.951	0.949	0.937	0.946	0.946
	1	0.837	0.667	0.684	0.870	0.913	0.900	**0.924**	0.750	0.677
ilpd	1	0.346	0.371	0.342	**0.453**	0.365	0.400	0.345	0.428	0.389
	2	**0.753**	0.725	0.729	0.703	0.753	0.705	0.738	0.697	0.734
ionosphere	b	0.363	**0.584**	0.426	0.246	0.303	0.367	0.301	0.300	0.370
	g	0.949	0.903	0.951	0.963	**0.971**	0.908	0.956	0.947	0.956
iris	Iris-setosa	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
	Iris-versicolor	0.974	0.962	**0.990**	**0.990**	0.975	0.979	**0.990**	0.968	0.985
	Iris-virginica	0.942	0.916	0.942	**0.962**	0.955	0.941	0.924	0.925	0.956
landsat	1	**0.997**	0.966	0.991	0.969	0.992	0.990	0.992	0.937	0.992
	2	**0.988**	0.747	0.860	0.859	0.986	0.986	0.975	0.916	0.881
	3	0.975	0.939	0.971	0.908	**0.976**	0.969	0.970	0.928	0.974
	4	0.896	0.812	0.843	0.799	0.888	**0.925**	0.902	0.792	0.853
	5	**0.941**	0.779	0.858	0.760	0.924	0.893	0.898	0.792	0.894
	7	0.954	0.872	0.903	0.894	0.940	0.960	**0.960**	0.910	0.903
leaf	1	0.980	0.978	**0.981**	0.978	0.978	0.946	0.915	0.888	0.978
	2	0.959	0.953	0.964	0.932	0.923	**0.980**	0.977	0.871	0.958
	3	0.968	0.964	0.986	0.977	0.974	0.889	0.885	0.780	**0.991**
	4	0.866	0.826	0.834	0.887	0.891	0.703	0.837	0.775	**0.907**
	5	0.998	0.993	**0.999**	0.955	0.995	0.956	0.994	0.795	0.997
	6	**0.997**	0.977	0.989	0.967	**0.997**	0.921	0.931	0.721	0.991
	7	0.961	0.964	0.962	**0.985**	0.971	0.867	0.958	0.742	0.974
	8	**1.000**	**1.000**	**1.000**	0.997	0.966	0.921	0.997	0.784	**1.000**
	9	0.957	0.921	0.949	**0.963**	0.958	0.895	0.928	0.906	0.959
	10	0.995	0.993	**0.996**	0.946	0.993	0.971	0.952	0.741	0.995
	11	0.999	0.998	**1.000**	0.996	**1.000**	0.960	0.998	0.890	0.999
	12	0.970	0.953	0.966	0.898	**0.974**	0.944	0.939	0.721	0.964
	13	0.977	0.981	**0.986**	0.963	0.981	0.978	0.973	0.870	0.983
	14	**0.948**	0.948	0.938	0.876	0.914	0.904	0.935	0.877	0.937
	15	**1.000**	0.998	**1.000**	0.988	**1.000**	0.977	0.985	0.850	**1.000**
	22	0.793	0.811	0.804	**0.880**	0.807	0.754	0.769	0.754	0.845
	23	0.996	0.997	0.995	0.983	**0.997**	0.807	0.972	0.845	**0.997**
	24	0.933	0.904	0.922	**0.952**	0.920	0.738	0.877	0.740	0.939
	25	0.998	0.998	**1.000**	0.977	0.998	0.932	0.954	0.905	0.998
	26	0.895	0.845	0.857	**0.930**	0.894	0.839	0.819	0.873	0.881
	27	0.968	0.883	0.918	**0.982**	0.927	0.847	0.867	0.885	0.948
	28	0.962	0.895	0.939	**0.974**	0.967	0.918	0.913	0.850	0.954

Continued on next page

Table C.1: 5-fold cross validation AUROC of each data descriptor with leave-one-dataset-out optimal hyperparameter values (where applicable).

Dataset	Target class	NND	LNND	LOF	MD	SVM	IF	EIF	SAE	ALP
	29	**1.000**	0.994	**1.000**	0.995	**1.000**	0.998	0.983	0.787	**1.000**
	30	0.992	0.989	0.995	0.990	0.995	0.913	0.974	0.863	**0.998**
	31	**0.999**	0.996	0.997	0.997	0.998	0.922	0.937	0.911	0.997
	32	0.962	0.959	0.966	0.956	0.965	0.852	0.956	0.814	**0.973**
	33	0.922	0.920	0.947	0.933	0.922	**0.958**	0.927	0.792	0.950
	34	**0.999**	0.990	0.996	0.997	0.947	0.805	0.796	0.956	0.997
	35	0.943	**0.945**	0.938	0.903	0.882	0.918	0.876	0.797	0.943
	36	0.994	**1.000**	**1.000**	0.994	0.992	0.945	0.947	0.879	**1.000**
letter	A	1.000	0.910	0.977	0.995	**1.000**	0.941	0.948	0.965	0.980
	B	**0.992**	0.934	0.975	0.987	0.990	0.924	0.932	0.956	0.984
	C	**0.996**	0.916	0.972	0.981	0.993	0.935	0.942	0.945	0.978
	D	**0.995**	0.933	0.978	0.983	0.992	0.907	0.924	0.916	0.986
	E	**0.988**	0.927	0.981	0.980	0.973	0.876	0.885	0.939	0.986
	F	**0.995**	0.917	0.975	0.979	0.990	0.944	0.920	0.928	0.980
	G	**0.991**	0.910	0.967	0.974	0.977	0.890	0.900	0.927	0.978
	H	0.954	0.856	0.951	0.936	0.928	0.844	0.823	0.907	**0.964**
	I	**0.996**	0.829	0.899	0.978	0.991	0.942	0.941	0.970	0.946
	J	**0.997**	0.902	0.961	0.982	0.991	0.951	0.942	0.942	0.970
	K	**0.984**	0.924	0.974	0.969	0.962	0.870	0.869	0.949	0.983
	L	**0.997**	0.884	0.953	0.987	0.985	0.916	0.939	0.952	0.964
	M	**0.995**	0.906	0.967	0.972	0.993	0.908	0.886	0.947	0.985
	N	**0.992**	0.897	0.953	0.986	0.989	0.926	0.924	0.941	0.979
	O	**0.990**	0.933	0.986	0.986	0.990	0.922	0.934	0.956	0.990
	P	**0.995**	0.919	0.970	0.983	0.992	0.945	0.951	0.928	0.977
	Q	0.985	0.941	0.988	0.976	0.975	0.877	0.884	0.938	**0.992**
	R	**0.993**	0.952	0.985	0.983	0.986	0.930	0.934	0.939	0.989
	S	0.980	0.902	0.978	0.966	0.973	0.835	0.828	0.932	**0.983**
	T	**0.996**	0.924	0.973	0.987	0.985	0.944	0.959	0.947	0.983
	U	**0.996**	0.915	0.972	0.989	0.989	0.906	0.900	0.968	0.981
	V	**0.997**	0.937	0.976	0.993	0.995	0.949	0.958	0.978	0.985
	W	**0.999**	0.926	0.986	0.994	0.998	0.965	0.978	0.975	0.989
	X	0.982	0.911	0.973	0.984	0.983	0.889	0.888	0.945	**0.984**
	Y	**0.994**	0.886	0.970	0.977	0.983	0.922	0.936	0.962	0.976
	Z	**0.991**	0.895	0.975	0.977	0.988	0.900	0.899	0.954	0.988
magic	g	0.845	0.808	0.860	0.803	0.787	0.757	0.777	0.777	**0.873**
	h	0.453	0.486	0.483	0.435	**0.491**	0.442	0.406	0.455	0.484
mfeat	0	0.998	0.983	0.997	0.996	0.996	0.988	0.983	**0.999**	0.996
	1	0.995	0.916	0.990	0.995	0.993	0.980	0.972	0.982	**0.997**
	2	**0.999**	0.987	0.999	0.998	0.998	0.993	0.988	0.983	0.999
	3	**0.985**	0.933	0.976	0.983	0.983	0.973	0.970	0.967	0.979
	4	**0.997**	0.976	0.996	0.993	0.994	0.989	0.983	0.986	0.996
	5	**0.989**	0.930	0.985	0.984	0.981	0.967	0.960	0.936	0.983
	6	**0.997**	0.969	0.993	0.997	0.995	0.987	0.987	0.988	0.995
	7	**0.999**	0.982	0.998	0.997	0.998	0.990	0.987	0.995	0.998
	8	0.987	0.953	0.984	**0.988**	0.981	0.968	0.962	0.963	0.985
	9	0.997	0.968	0.995	**0.997**	0.996	0.986	0.983	0.982	0.995

Continued on next page

Table C.1: 5-fold cross validation AUROC of each data descriptor with leave-one-dataset-out optimal hyperparameter values (where applicable).

Dataset	Target class	NND	LNND	LOF	MD	SVM	IF	EIF	SAE	ALP
miniboone	0	0.506	0.590	0.612	0.693	0.717	0.775	0.709	**0.829**	0.600
	1	0.898	0.807	**0.902**	0.865	0.842	0.799	0.784	0.700	0.901
new-thyroid	1	0.972	0.957	0.991	0.987	0.988	0.992	0.987	0.965	**0.992**
	2	0.963	0.937	0.956	0.965	0.966	0.879	0.944	0.960	**0.971**
	3	0.808	0.779	0.863	0.938	0.851	0.918	0.785	**0.978**	0.878
page-blocks	1	0.908	0.899	0.934	0.957	**0.964**	0.924	0.930	0.925	0.940
	2	0.916	0.869	0.916	0.898	0.924	0.875	0.916	0.853	**0.935**
	3	0.909	0.793	0.846	**0.991**	0.934	0.506	0.821	0.852	0.887
	4	**0.962**	0.938	0.946	0.954	0.938	0.946	0.949	0.953	0.961
	5	0.783	0.766	0.831	0.816	0.784	0.760	0.769	0.687	**0.856**
pop-failures	0	0.836	0.862	0.885	0.814	0.903	0.856	**0.914**	0.573	0.903
	1	0.619	0.556	0.645	0.697	0.678	0.658	**0.717**	0.538	0.686
seeds	1	0.919	0.947	0.932	0.950	0.941	0.955	**0.963**	0.840	0.945
	2	0.989	0.968	0.982	0.992	**0.998**	0.988	0.993	0.989	0.987
	3	0.978	0.956	0.981	0.986	**0.990**	0.973	0.977	0.913	0.982
segment	1	0.997	0.993	0.998	0.998	0.996	0.995	0.986	0.996	**0.999**
	2	0.999	0.998	0.999	0.997	0.997	0.995	0.992	**1.000**	0.999
	3	0.906	0.848	0.922	**0.947**	0.918	0.899	0.859	0.933	0.944
	4	0.920	0.841	0.871	**0.945**	0.918	0.896	0.844	0.922	0.913
	5	0.945	0.902	0.920	**0.950**	0.937	0.924	0.855	0.940	0.946
	6	0.997	0.993	0.998	0.999	0.994	0.974	0.944	0.993	**0.999**
	7	0.999	0.992	0.998	0.998	0.997	0.996	0.992	**0.999**	0.998
seismic-bumps	0	**0.743**	0.520	0.572	0.712	0.694	0.712	0.728	0.591	0.602
	1	0.346	0.428	0.415	**0.517**	0.476	0.487	0.378	0.477	0.397
sensorless	1	0.980	0.974	0.987	0.947	0.969	0.897	0.909	0.947	**0.990**
	2	0.973	0.961	0.977	0.926	0.956	0.883	0.909	0.936	**0.980**
	3	0.984	0.978	0.994	0.954	0.974	0.876	0.889	0.946	**0.995**
	4	0.985	0.977	0.992	0.938	0.966	0.893	0.877	0.883	**0.993**
	5	0.972	0.961	0.981	0.890	0.946	0.858	0.867	0.846	**0.984**
	6	0.971	0.960	0.982	0.910	0.946	0.877	0.866	0.917	**0.985**
	7	0.996	0.989	0.997	0.997	0.986	0.941	0.932	0.991	**0.998**
	8	0.970	0.962	0.981	0.893	0.941	0.873	0.852	0.868	**0.985**
	9	0.972	0.963	0.984	0.871	0.939	0.869	0.830	0.854	**0.987**
	10	**0.988**	0.977	0.985	0.963	0.980	0.893	0.950	0.944	0.988
	11	0.997	0.988	0.997	**0.998**	0.990	0.969	0.938	0.990	0.998
shuttle	1	0.990	0.985	0.996	0.940	0.993	0.930	0.902	0.974	**0.998**
	2	0.892	0.866	0.931	0.958	0.962	0.843	0.812	0.930	**0.964**
	3	0.887	0.830	0.900	**0.947**	0.944	0.827	0.783	0.835	0.924
	4	0.988	0.986	0.992	0.986	0.987	0.973	0.956	0.989	**0.996**
	5	0.981	0.982	0.988	0.996	0.990	0.985	0.965	0.996	**0.997**
	6	0.930	0.878	0.871	0.548	0.946	0.754	0.673	0.656	**0.990**
	7	0.865	0.851	0.868	**0.964**	0.926	0.468	0.737	0.871	0.924
skin	1	**1.000**	0.968	0.955	1.000	1.000	0.991	0.999	0.999	1.000

Continued on next page

Table C.1: 5-fold cross validation AUROC of each data descriptor with leave-one-dataset-out optimal hyperparameter values (where applicable).

Dataset	Target class	NND	LNND	LOF	MD	SVM	IF	EIF	SAE	ALP
	2	**0.997**	0.839	0.909	0.889	0.426	0.892	0.916	0.948	0.963
somerville	0	0.473	**0.575**	0.461	0.443	0.454	0.498	0.452	0.419	0.454
	1	0.631	0.567	0.629	0.602	**0.644**	0.619	0.635	0.516	0.627
sonar	M	0.573	0.557	0.641	**0.680**	0.672	0.597	0.520	0.666	0.679
	R	0.680	0.626	0.674	0.591	0.674	0.664	0.675	0.585	**0.705**
spambase	0	0.794	0.576	0.646	0.798	0.756	**0.842**	0.691	0.779	0.681
	1	0.498	0.604	0.676	0.849	0.641	0.710	0.488	**0.851**	0.734
spectf	0	0.842	0.820	**0.847**	0.640	0.841	0.787	0.838	0.712	0.840
	1	0.240	0.283	0.208	0.247	0.290	0.265	0.249	**0.326**	0.249
sportsarticles	objective	**0.848**	0.605	0.638	0.835	0.836	0.839	0.829	0.814	0.651
	subjective	0.224	0.370	0.338	0.244	0.440	**0.636**	0.467	0.288	0.312
texture	2	0.993	0.939	0.987	**0.996**	0.989	0.951	0.964	0.976	0.991
	3	0.999	0.988	0.999	**1.000**	0.999	0.973	0.989	0.986	0.999
	4	1.000	0.998	1.000	**1.000**	1.000	0.995	0.998	0.999	1.000
	6	0.999	0.990	0.999	**1.000**	0.999	0.983	0.993	0.993	0.999
	7	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.998	1.000	1.000	**1.000**
	8	0.982	0.964	0.989	**0.994**	0.985	0.916	0.912	0.967	0.993
	9	0.993	0.963	0.994	**0.996**	0.990	0.941	0.961	0.977	0.994
	10	0.977	0.973	0.991	0.992	0.979	0.918	0.930	0.977	**0.992**
	12	**1.000**	1.000	**1.000**	**1.000**	**1.000**	0.997	1.000	**1.000**	**1.000**
	13	0.997	0.997	1.000	0.998	0.999	0.986	0.995	0.992	**1.000**
	14	0.988	0.939	0.984	**0.998**	0.986	0.922	0.938	0.992	0.987
transfusion	0	0.572	0.523	0.514	0.510	0.537	**0.597**	0.574	0.571	0.562
	1	0.554	0.549	0.556	**0.688**	0.617	0.672	0.660	0.566	0.559
vehicle	bus	0.969	0.922	0.967	**0.978**	0.961	0.836	0.846	0.942	0.974
	opel	0.768	0.665	0.696	**0.850**	0.719	0.714	0.721	0.702	0.736
	saab	0.759	0.719	0.717	**0.890**	0.782	0.737	0.733	0.751	0.773
	van	0.962	0.907	0.931	**0.968**	0.936	0.864	0.887	0.935	0.956
waveform	0	0.821	0.804	0.834	0.844	**0.852**	0.847	0.848	0.775	0.831
	1	0.885	0.864	0.893	0.895	0.906	**0.915**	0.906	0.778	0.891
	2	0.887	0.867	0.894	0.899	0.908	**0.916**	0.906	0.821	0.892
wdbc	B	0.951	0.913	0.948	**0.966**	0.953	0.957	0.953	0.904	0.957
	M	0.672	0.745	0.786	0.664	0.785	**0.874**	0.726	0.633	0.823
wifi	1	0.999	0.995	0.999	0.998	0.999	0.984	0.987	0.922	**0.999**
	2	0.972	0.938	0.968	0.925	**0.973**	0.971	0.929	0.872	0.971
	3	0.992	0.982	0.992	0.992	0.993	0.993	0.992	0.919	**0.994**
	4	0.998	0.994	0.999	0.998	0.999	0.997	0.998	0.873	**0.999**
wilt	n	0.828	0.830	0.923	0.728	0.725	0.490	0.555	0.421	**0.940**
	w	0.933	0.870	0.927	**0.972**	0.947	0.881	0.900	0.862	0.943
wine	1	0.990	0.970	0.994	0.983	0.995	0.980	0.916	0.852	**0.996**
	2	0.925	0.898	0.930	**0.952**	0.945	0.933	0.933	0.860	0.940

Continued on next page

Table C.1: 5-fold cross validation AUROC of each data descriptor with leave-one-dataset-out optimal hyperparameter values (where applicable).

Dataset	Target class	NND	LNND	LOF	MD	SVM	IF	EIF	SAE	ALP
	3	0.999	0.982	0.997	**1.000**	**1.000**	0.989	0.997	0.805	0.997
wisconsin	2	0.993	0.752	0.676	0.987	0.990	**0.995**	0.994	0.970	0.864
	4	0.566	0.848	0.781	0.820	0.903	**0.959**	0.809	0.304	0.881
wpbc	N	0.565	0.593	0.550	0.517	0.528	0.550	0.572	**0.621**	0.532
	R	0.503	0.586	0.577	0.492	0.559	**0.602**	0.543	0.508	0.565
yeast	CYT	**0.758**	0.682	0.725	0.716	0.733	0.702	0.706	0.628	0.737
	ERL	0.839	0.816	0.847	0.705	0.771	0.782	0.704	0.445	**0.867**
	EXC	0.869	0.904	0.925	0.879	0.863	**0.935**	0.921	0.675	0.906
	ME1	0.957	0.886	0.941	0.956	**0.962**	0.867	0.830	0.730	0.958
	ME2	0.732	0.736	0.737	0.706	0.719	**0.748**	0.740	0.629	0.744
	ME3	0.867	0.830	0.885	**0.908**	0.903	0.874	0.870	0.786	0.904
	MIT	0.721	0.681	0.728	**0.730**	0.724	0.729	0.717	0.654	0.707
	NUC	0.614	0.616	0.628	0.650	**0.658**	0.632	0.632	0.612	0.635
	POX	0.564	0.516	0.553	0.595	0.572	**0.661**	0.551	0.640	0.585
	VAC	0.608	0.563	0.606	0.585	0.593	0.592	0.609	0.581	**0.613**

C.2 Missing-indicators

The tables in this section correspond to the experiments in Chapter 10.

Table C.2 contains the mean AUROC across five-fold cross-validation and five random states for each classifier, each dataset, each imputation strategy, without and with missing-indicators. Table C.3 contains the mean AUROC for CART, GBM and ERT with updated hyperparameter values (as discussed in Subsection 10.3). Table C.4 contains the mean AUROC obtained by imputing missing categorical values with the mean, after one-hot encoding (Subsection 10.3).

Table C.2: AUROC, main experiment. **Bold**: higher value (without or with missing-indicators) by at least 0.001.

Classifier	Dataset	Imputation strategy, missing-indicators no/yes					
		Mean/mode		Neighbours		Iterative	
		No	Yes	No	Yes	No	Yes
NN-1	adult	0.857	0.858	0.858	0.858	0.858	0.858
	agaricus-lepiota	1.000	1.000	1.000	1.000	1.000	1.000
	aps-failure	**0.928**	0.926	**0.926**	0.922	**0.928**	0.923
	arrhythmia	0.760	0.760	0.760	0.760	0.760	0.760
	bands	0.836	**0.838**	0.834	**0.847**	0.836	**0.848**
	ckd	**0.998**	0.994	0.991	**0.993**	0.989	**0.991**
	crx	0.908	**0.909**	0.904	**0.908**	0.909	**0.910**
	dress-sales	0.548	**0.555**	0.540	**0.545**	0.527	**0.531**

Continued on next page

Table C.2: AUROC, main experiment. **Bold**: higher value (without or with missing-indicators) by at least 0.001.

Classifier	Dataset	Imputation strategy, missing-indicators no/yes					
		Mean/mode		Neighbours		Iterative	
		No	Yes	No	Yes	No	Yes
	exasens	0.710	**0.726**	0.703	**0.713**	0.717	**0.726**
	hcc	0.699	**0.760**	0.707	**0.745**	0.712	**0.753**
	heart-disease	0.846	0.847	0.841	**0.844**	0.843	**0.846**
	hepatitis	**0.849**	0.841	0.841	**0.850**	0.839	**0.847**
	horse-colic	0.716	**0.733**	**0.738**	0.734	0.726	**0.738**
	mammographic-masses	0.821	**0.827**	0.821	**0.825**	0.824	**0.831**
	mi	0.572	**0.579**	0.564	**0.580**	0.569	**0.579**
	nomao	**0.983**	0.982	0.978	**0.981**	0.983	0.982
	primary-tumor	0.675	**0.687**	0.678	**0.693**	0.676	**0.687**
	secom	0.641	**0.651**	0.641	**0.643**	0.646	**0.653**
	soybean	0.993	0.993	0.992	0.993	0.993	0.993
	thyroid0387	**0.879**	0.877	0.878	0.877	0.875	0.875
NN-2	adult	0.860	0.861	0.861	0.861	0.861	0.860
	agaricus-lepiota	1.000	1.000	1.000	1.000	1.000	1.000
	aps-failure	0.920	**0.922**	0.918	**0.920**	0.921	0.921
	arrhythmia	0.733	0.733	0.734	0.734	0.733	0.733
	bands	0.830	**0.832**	0.818	**0.835**	0.825	**0.836**
	ckd	**0.999**	0.996	0.992	**0.995**	0.991	**0.993**
	crx	0.899	0.900	0.898	**0.899**	0.900	0.901
	dress-sales	**0.554**	0.547	**0.541**	0.539	**0.532**	0.527
	exasens	0.709	**0.716**	0.699	**0.706**	0.712	**0.718**
	hcc	0.690	**0.696**	0.695	**0.709**	0.698	**0.705**
	heart-disease	0.831	**0.835**	0.828	**0.837**	0.829	**0.836**
	hepatitis	**0.861**	0.851	0.846	**0.850**	0.860	**0.862**
	horse-colic	0.684	**0.710**	**0.724**	0.706	0.695	**0.704**
	mammographic-masses	0.820	**0.825**	0.821	**0.824**	0.822	**0.828**
	mi	0.561	**0.563**	0.555	**0.560**	0.563	0.563
	nomao	0.980	**0.982**	0.976	**0.980**	0.980	**0.981**
	primary-tumor	0.667	**0.673**	0.670	**0.675**	0.666	**0.677**
	secom	0.607	**0.612**	0.614	**0.617**	0.607	**0.613**
	soybean	0.986	**0.988**	0.987	0.988	0.986	**0.988**
	thyroid0387	**0.878**	0.877	**0.878**	0.876	0.871	0.871
NN-1-D	adult	0.838	0.838	0.837	**0.839**	0.837	0.838
	agaricus-lepiota	1.000	1.000	1.000	1.000	1.000	1.000
	aps-failure	**0.929**	0.926	**0.927**	0.922	**0.928**	0.923
	arrhythmia	0.764	0.764	0.763	0.763	0.764	0.764
	bands	0.871	**0.875**	0.865	**0.879**	0.870	**0.880**
	ckd	**0.998**	0.994	0.991	**0.993**	0.989	**0.991**
	crx	0.907	**0.908**	0.905	**0.908**	0.908	**0.909**
	dress-sales	0.544	**0.560**	0.538	**0.545**	0.528	**0.535**
	exasens	0.629	**0.641**	0.625	**0.634**	0.632	**0.640**
	hcc	0.728	**0.786**	0.733	**0.772**	0.738	**0.773**
	heart-disease	0.847	**0.848**	0.843	**0.845**	0.843	**0.847**
	hepatitis	**0.857**	0.853	0.841	**0.855**	0.841	**0.853**
	horse-colic	0.743	**0.751**	**0.762**	0.752	0.749	**0.757**
	mammographic-masses	0.802	**0.806**	0.798	**0.805**	0.803	**0.808**
	mi	0.572	**0.580**	0.564	**0.580**	0.569	**0.579**

Continued on next page

Table C.2: AUROC, main experiment. **Bold**: higher value (without or with missing-indicators) by at least 0.001.

Classifier	Dataset	Imputation strategy, missing-indicators no/yes					
		Mean/mode		Neighbours		Iterative	
		No	Yes	No	Yes	No	Yes
	nomao	**0.984**	0.983	0.979	**0.982**	0.984	0.983
	primary-tumor	0.665	**0.676**	0.667	**0.684**	0.665	**0.677**
	secom	0.644	**0.652**	0.644	**0.645**	0.647	**0.655**
	soybean	0.993	0.993	0.992	0.993	0.993	0.993
	thyroid0387	**0.881**	0.879	0.880	0.879	0.877	0.877
NN-2-D	adult	0.842	0.843	0.842	**0.843**	0.842	0.843
	agaricus-lepiota	1.000	1.000	1.000	1.000	1.000	1.000
	aps-failure	0.920	**0.922**	0.918	**0.921**	0.922	0.922
	arrhythmia	0.735	0.736	0.736	0.736	0.735	0.735
	bands	0.859	**0.861**	0.844	**0.863**	0.850	**0.863**
	ckd	**0.999**	0.996	0.992	**0.995**	0.991	**0.993**
	crx	0.898	0.899	0.898	**0.900**	0.900	0.901
	dress-sales	0.548	0.548	**0.543**	0.538	**0.534**	0.532
	exasens	0.628	**0.635**	0.623	**0.629**	0.629	**0.634**
	hcc	0.710	**0.723**	0.716	**0.737**	0.719	**0.729**
	heart-disease	0.833	**0.838**	0.830	**0.839**	0.831	**0.839**
	hepatitis	**0.862**	0.856	0.847	**0.852**	0.859	**0.865**
	horse-colic	0.712	**0.731**	**0.745**	0.730	0.719	**0.729**
	mammographic-masses	0.802	**0.805**	0.799	**0.804**	0.802	**0.807**
	mi	0.560	**0.563**	0.556	**0.560**	0.564	0.565
	nomao	0.981	**0.983**	0.977	**0.981**	0.981	**0.982**
	primary-tumor	0.659	**0.666**	0.660	**0.667**	0.657	**0.669**
	secom	0.606	**0.610**	0.612	**0.615**	0.606	**0.611**
	soybean	0.986	**0.988**	0.987	0.988	0.986	**0.988**
	thyroid0387	**0.880**	0.878	**0.879**	0.878	0.872	0.872
SVM-L	adult	0.905	**0.906**	0.905	0.906	0.905	0.906
	agaricus-lepiota	1.000	1.000	1.000	1.000	1.000	1.000
	aps-failure	0.966	**0.969**	0.961	**0.969**	0.963	**0.966**
	arrhythmia	0.818	**0.843**	0.819	**0.843**	0.818	**0.843**
	bands	0.796	**0.817**	0.791	**0.809**	0.760	**0.801**
	ckd	1.000	1.000	0.999	1.000	0.999	1.000
	crx	**0.922**	0.920	0.920	0.920	0.922	0.921
	dress-sales	**0.598**	0.593	**0.594**	0.588	0.591	**0.597**
	exasens	0.762	**0.780**	0.761	**0.769**	0.761	**0.780**
	hcc	**0.757**	0.738	**0.781**	0.756	**0.746**	0.733
	heart-disease	0.866	0.865	0.866	**0.867**	0.867	0.868
	hepatitis	**0.848**	0.824	**0.857**	0.831	**0.856**	0.833
	horse-colic	**0.790**	0.784	**0.798**	0.784	**0.770**	0.762
	mammographic-masses	0.865	**0.867**	0.862	**0.865**	0.864	0.864
	mi	0.641	**0.666**	0.639	**0.669**	0.636	**0.671**
	nomao	0.986	**0.988**	0.986	**0.988**	0.985	**0.988**
	primary-tumor	0.769	0.769	**0.772**	0.770	0.778	0.777
	secom	0.626	**0.629**	**0.671**	0.659	**0.631**	0.628
	soybean	0.999	0.999	0.999	0.999	0.999	0.999
	thyroid0387	0.964	0.964	0.964	0.964	0.954	**0.956**
SVM-G	adult	0.895	**0.897**	0.896	0.896	0.896	0.897
	agaricus-lepiota	1.000	1.000	1.000	1.000	1.000	1.000

Continued on next page

Table C.2: AUROC, main experiment. **Bold**: higher value (without or with missing-indicators) by at least 0.001.

Classifier	Dataset	Imputation strategy, missing-indicators no/yes					
		Mean/mode		Neighbours		Iterative	
		No	Yes	No	Yes	No	Yes
	aps-failure	0.967	0.968	0.960	**0.965**	0.965	**0.966**
	arrhythmia	0.848	0.848	0.848	0.848	0.848	0.848
	bands	0.855	**0.865**	0.858	**0.870**	0.857	**0.869**
	ckd	1.000	1.000	1.000	1.000	1.000	1.000
	crx	0.926	**0.927**	0.924	**0.927**	0.926	**0.928**
	dress-sales	0.618	**0.620**	**0.620**	0.619	0.607	**0.612**
	exasens	0.772	**0.780**	0.767	**0.780**	0.773	**0.780**
	hcc	0.778	**0.790**	0.785	**0.793**	0.770	**0.783**
	heart-disease	**0.865**	0.864	0.863	**0.864**	0.864	0.864
	hepatitis	0.893	0.892	**0.888**	0.887	**0.893**	0.890
	horse-colic	0.768	**0.771**	0.784	**0.786**	0.767	**0.769**
	mammographic-masses	0.840	**0.845**	0.838	**0.841**	0.839	**0.842**
	mi	0.635	**0.643**	0.637	**0.645**	0.639	**0.648**
	nomao	0.991	0.992	0.988	**0.991**	0.989	**0.991**
	primary-tumor	0.762	**0.765**	0.764	**0.767**	0.766	0.767
	secom	**0.699**	0.694	**0.702**	0.698	**0.689**	0.685
	soybean	0.999	0.999	0.999	0.999	0.999	0.999
	thyroid0387	0.978	0.978	0.978	0.977	0.969	0.970
LR	adult	0.905	**0.906**	0.906	0.906	0.906	0.906
	agaricus-lepiota	1.000	1.000	1.000	1.000	1.000	1.000
	aps-failure	0.971	**0.979**	0.971	**0.980**	0.967	**0.978**
	arrhythmia	0.860	0.860	0.860	0.860	0.859	0.860
	bands	0.819	**0.833**	0.811	**0.830**	0.808	**0.828**
	ckd	1.000	1.000	1.000	1.000	1.000	1.000
	crx	0.924	0.923	0.923	0.923	0.924	0.924
	dress-sales	0.620	0.620	0.619	**0.624**	0.614	**0.620**
	exasens	0.774	**0.783**	0.768	**0.775**	0.773	**0.782**
	hcc	**0.778**	0.760	**0.796**	0.774	**0.772**	0.755
	heart-disease	0.867	**0.868**	0.867	**0.869**	0.867	**0.869**
	hepatitis	**0.863**	0.856	**0.871**	0.862	**0.870**	0.862
	horse-colic	**0.789**	0.786	**0.793**	0.786	**0.769**	0.764
	mammographic-masses	0.866	**0.868**	0.863	**0.865**	0.865	0.865
	mi	0.654	**0.685**	0.645	**0.685**	0.650	**0.688**
	nomao	0.986	**0.988**	0.986	**0.988**	0.985	**0.988**
	primary-tumor	0.773	**0.776**	0.772	**0.775**	0.780	**0.783**
	secom	**0.686**	0.678	**0.687**	0.680	**0.676**	0.673
	soybean	0.999	0.999	0.999	0.999	0.999	0.999
	thyroid0387	0.974	0.975	0.975	0.975	0.973	0.973
MLP	adult	0.890	0.890	**0.891**	0.889	**0.891**	0.890
	agaricus-lepiota	1.000	1.000	1.000	1.000	1.000	1.000
	aps-failure	0.928	**0.942**	0.931	**0.943**	0.931	**0.942**
	arrhythmia	0.831	**0.846**	0.831	**0.845**	0.831	**0.845**
	bands	0.871	**0.879**	0.873	**0.885**	0.868	**0.882**
	ckd	1.000	1.000	1.000	1.000	0.999	1.000
	crx	0.902	**0.906**	0.901	**0.905**	0.900	**0.905**
	dress-sales	0.549	**0.553**	0.560	**0.561**	0.544	0.545
	exasens	0.759	**0.762**	0.746	**0.755**	0.757	**0.763**

Continued on next page

Table C.2: AUROC, main experiment. **Bold**: higher value (without or with missing-indicators) by at least 0.001.

Classifier	Dataset	Imputation strategy, missing-indicators no/yes					
		Mean/mode		Neighbours		Iterative	
		No	Yes	No	Yes	No	Yes
	hcc	0.778	**0.781**	0.791	**0.796**	0.777	**0.781**
	heart-disease	**0.819**	0.815	**0.816**	0.811	**0.818**	0.816
	hepatitis	0.861	0.861	**0.870**	0.865	**0.872**	0.866
	horse-colic	0.714	**0.744**	0.727	**0.756**	0.719	**0.734**
	mammographic-masses	**0.845**	0.840	**0.841**	0.836	**0.847**	0.840
	mi	0.659	**0.695**	0.656	**0.697**	0.660	**0.697**
	nomao	0.991	0.991	0.987	**0.990**	0.990	0.991
	primary-tumor	0.768	**0.782**	0.765	**0.778**	0.769	**0.785**
	secom	0.693	**0.701**	0.699	**0.704**	0.686	**0.697**
	soybean	0.999	0.999	0.999	0.999	0.999	0.999
	thyroid0387	0.988	0.988	0.988	0.987	0.986	0.985
CART	adult	0.776	0.775	0.776	0.775	**0.776**	0.774
	agaricus-lepiota	1.000	1.000	1.000	1.000	1.000	1.000
	aps-failure	0.855	**0.858**	**0.858**	0.857	0.854	**0.857**
	arrhythmia	**0.712**	0.710	**0.712**	0.702	**0.714**	0.702
	bands	**0.716**	0.713	0.697	**0.716**	0.706	**0.717**
	ckd	**0.965**	0.964	0.979	0.978	**0.972**	0.970
	crx	**0.818**	0.812	**0.813**	0.810	**0.815**	0.809
	dress-sales	0.524	**0.548**	0.526	**0.529**	**0.534**	0.532
	exasens	**0.618**	0.616	**0.618**	0.608	0.621	**0.626**
	hcc	0.593	**0.603**	**0.619**	0.617	**0.614**	0.601
	heart-disease	0.702	0.703	0.701	0.700	0.703	**0.706**
	hepatitis	**0.660**	0.657	**0.691**	0.673	**0.703**	0.700
	horse-colic	**0.695**	0.673	**0.700**	0.663	**0.680**	0.676
	mammographic-masses	**0.748**	0.744	0.747	0.746	0.744	**0.746**
	mi	0.572	0.572	0.549	**0.574**	0.557	**0.571**
	nomao	0.935	0.935	0.922	**0.925**	0.926	0.927
	primary-tumor	0.621	0.621	0.625	**0.627**	0.622	**0.623**
	secom	0.547	**0.552**	0.555	**0.558**	**0.542**	0.538
	soybean	0.975	**0.977**	0.973	0.974	0.971	**0.973**
	thyroid0387	**0.897**	0.888	**0.875**	0.871	**0.886**	0.883
RF	adult	0.890	0.890	0.890	0.891	0.891	0.890
	agaricus-lepiota	1.000	1.000	1.000	1.000	1.000	1.000
	aps-failure	0.988	0.989	0.988	0.989	0.988	0.988
	arrhythmia	0.883	**0.884**	0.885	0.885	**0.886**	0.883
	bands	0.893	**0.896**	0.886	**0.898**	0.896	0.896
	ckd	1.000	1.000	1.000	1.000	1.000	1.000
	crx	0.932	0.931	**0.934**	0.932	0.931	0.931
	dress-sales	0.591	**0.606**	0.583	**0.602**	0.582	**0.597**
	exasens	0.701	0.701	0.689	**0.694**	0.698	**0.701**
	hcc	0.803	**0.816**	0.813	0.813	0.794	**0.806**
	heart-disease	0.861	**0.864**	0.862	**0.866**	0.864	**0.866**
	hepatitis	0.882	**0.887**	**0.890**	0.887	**0.888**	0.886
	horse-colic	**0.800**	0.791	**0.811**	0.809	0.793	0.792
	mammographic-masses	0.812	**0.821**	0.815	**0.819**	0.812	**0.820**
	mi	0.687	0.687	0.676	**0.681**	**0.687**	0.679
	nomao	0.994	0.994	0.991	**0.992**	0.993	0.993

Continued on next page

Table C.2: AUROC, main experiment. **Bold**: higher value (without or with missing-indicators) by at least 0.001.

Classifier	Dataset	Imputation strategy, missing-indicators no/yes					
		Mean/mode		Neighbours		Iterative	
		No	Yes	No	Yes	No	Yes
	primary-tumor	0.749	**0.758**	0.730	**0.761**	0.748	**0.761**
	secom	**0.722**	0.710	**0.719**	0.713	**0.722**	0.710
	soybean	0.999	0.999	0.999	0.999	0.999	0.999
	thyroid0387	0.994	0.994	**0.993**	0.992	**0.995**	0.992
ERT	adult	0.846	0.847	0.847	0.847	0.846	0.847
	agaricus-lepiota	1.000	1.000	1.000	1.000	1.000	1.000
	aps-failure	0.989	0.989	0.989	0.988	0.989	0.989
	arrhythmia	0.885	**0.889**	0.881	**0.885**	0.881	**0.885**
	bands	0.889	0.890	0.874	**0.890**	0.885	**0.892**
	ckd	1.000	1.000	1.000	1.000	1.000	1.000
	crx	**0.913**	0.911	0.916	0.915	**0.912**	0.910
	dress-sales	0.572	**0.600**	0.563	**0.594**	0.560	**0.589**
	exasens	**0.633**	0.632	0.622	**0.626**	0.624	**0.630**
	hcc	0.783	**0.799**	0.776	**0.804**	0.771	**0.796**
	heart-disease	0.858	**0.861**	0.862	**0.865**	0.861	0.861
	hepatitis	**0.871**	0.861	0.876	**0.877**	**0.882**	0.871
	horse-colic	**0.793**	0.780	**0.818**	0.796	**0.790**	0.780
	mammographic-masses	0.793	**0.801**	0.791	**0.800**	0.793	**0.801**
	mi	**0.689**	0.683	0.661	**0.683**	0.676	**0.686**
	nomao	0.994	0.993	0.991	**0.992**	0.993	0.993
	primary-tumor	0.702	**0.718**	0.698	**0.717**	0.704	**0.721**
	secom	**0.718**	0.713	**0.716**	0.705	0.706	**0.716**
	soybean	0.999	0.999	0.999	0.999	0.999	0.999
	thyroid0387	**0.979**	0.976	0.980	0.979	0.975	**0.977**
ABT	adult	0.915	0.915	0.915	0.915	0.915	0.915
	agaricus-lepiota	1.000	1.000	1.000	1.000	1.000	1.000
	aps-failure	0.987	0.987	0.987	0.987	0.986	0.987
	arrhythmia	**0.634**	0.632	0.634	0.633	**0.634**	0.632
	bands	0.806	0.806	0.793	**0.809**	0.805	**0.807**
	ckd	1.000	1.000	0.998	**0.999**	0.998	**1.000**
	crx	0.905	0.906	0.907	0.906	**0.909**	0.905
	dress-sales	**0.590**	0.582	**0.584**	0.578	0.587	**0.589**
	exasens	0.720	0.720	0.705	**0.717**	**0.713**	0.711
	hcc	0.715	**0.724**	**0.739**	0.735	**0.708**	0.687
	heart-disease	0.860	0.860	0.857	**0.861**	**0.861**	0.858
	hepatitis	0.797	**0.804**	0.824	**0.830**	0.805	**0.814**
	horse-colic	**0.753**	0.752	**0.749**	0.742	**0.735**	0.729
	mammographic-masses	0.856	**0.857**	0.855	0.856	0.854	0.855
	mi	0.555	**0.572**	0.572	**0.586**	0.573	0.572
	nomao	0.987	0.987	0.985	0.986	0.986	0.986
	primary-tumor	**0.661**	0.660	**0.670**	0.668	0.668	**0.671**
	secom	0.670	0.670	0.661	0.661	0.663	0.663
	soybean	0.863	**0.871**	0.777	**0.850**	0.855	**0.865**
	thyroid0387	0.685	0.685	0.666	0.666	0.674	0.674
GBM	adult	0.921	0.921	0.921	0.921	0.921	0.921
	agaricus-lepiota	1.000	1.000	1.000	1.000	1.000	1.000
	aps-failure	0.989	0.988	0.988	0.989	0.988	0.988

Continued on next page

Table C.2: AUROC, main experiment. **Bold**: higher value (without or with missing-indicators) by at least 0.001.

Classifier	Dataset	Imputation strategy, missing-indicators no/yes					
		Mean/mode		Neighbours		Iterative	
		No	Yes	No	Yes	No	Yes
	arrhythmia	0.873	**0.874**	**0.880**	0.875	**0.879**	0.878
	bands	0.869	0.870	0.857	**0.871**	0.870	**0.873**
	ckd	1.000	1.000	0.998	0.998	0.998	0.999
	crx	0.932	0.932	0.930	0.931	0.929	**0.931**
	dress-sales	**0.612**	0.606	0.597	**0.601**	**0.612**	0.609
	exasens	0.725	0.725	0.720	**0.724**	0.723	**0.725**
	hcc	0.759	**0.780**	0.762	**0.773**	**0.747**	0.742
	heart-disease	0.872	0.872	0.869	0.870	0.873	0.872
	hepatitis	**0.837**	0.828	0.837	**0.838**	0.854	0.854
	horse-colic	**0.793**	0.789	**0.794**	0.789	**0.798**	0.789
	mammographic-masses	0.850	**0.853**	0.847	**0.851**	0.846	**0.853**
	mi	**0.664**	0.663	0.659	**0.663**	0.654	**0.661**
	nomao	0.991	0.991	0.989	**0.990**	0.991	0.991
	primary-tumor	0.760	**0.763**	0.762	0.762	**0.754**	0.752
	secom	0.708	**0.710**	**0.717**	0.716	0.708	**0.711**
	soybean	0.999	0.999	0.998	0.999	0.998	0.998
	thyroid0387	**0.916**	0.914	0.896	0.896	0.903	**0.905**

Table C.3: AUROC, additional experiment for mean/mode imputation and classifiers with adjusted hyperparameter values. **Bold**: higher value (without or with missing-indicators) by at least 0.001.

Dataset	Classifier, missing-indicators no/yes					
	CART		GBM		ERT	
	No	Yes	No	Yes	No	Yes
adult	0.844	0.844	0.927	0.927	0.847	0.847
agaricus-lepiota	0.991	0.992	1.000	1.000	1.000	1.000
aps-failure	0.859	0.859	0.988	0.988	0.991	0.991
arrhythmia	**0.749**	0.748	0.850	**0.852**	0.897	**0.899**
bands	0.749	**0.759**	0.855	**0.857**	0.890	0.890
ckd	0.968	0.967	0.998	0.998	1.000	1.000
crx	0.897	0.897	0.934	0.933	0.914	0.914
dress-sales	0.568	**0.570**	0.608	**0.614**	0.572	**0.602**
exasens	0.723	**0.732**	0.755	**0.757**	0.626	0.626
hcc	0.577	**0.588**	0.737	**0.745**	0.791	**0.808**
heart-disease	0.777	0.777	0.870	**0.871**	0.861	**0.862**
hepatitis	**0.626**	0.578	**0.812**	0.809	**0.877**	0.873
horse-colic	**0.742**	0.724	**0.789**	0.783	**0.799**	0.782
mammographic-masses	0.823	0.823	0.857	**0.859**	0.795	**0.802**
mi	0.586	**0.592**	**0.650**	0.639	**0.702**	0.695
nomao	0.916	0.916	0.994	0.994	0.994	0.994
primary-tumor	0.703	**0.707**	0.766	0.767	0.705	**0.714**
secom	0.500	0.500	**0.684**	0.677	0.746	**0.747**
soybean	0.990	0.991	0.999	0.999	0.999	0.999
thyroid0387	0.909	0.909	0.913	**0.923**	0.987	0.987

Table C.4: AUROC, additional experiment for imputation of categorical attributes (mode imputation or mean imputation after one-hot encoding). **Bold**: higher value by at least 0.001.

Classifier	Dataset	Without missing-indicators		With missing-indicators	
		Mode	Mean	Mode	Mean
NN-1	adult	0.857	0.858	0.858	0.858
	agaricus-lepiota	1.000	1.000	1.000	1.000
	bands	0.836	**0.839**	0.838	**0.843**
	crx	0.908	**0.909**	0.909	0.909
	dress-sales	**0.548**	0.533	**0.555**	0.539
	horse-colic	0.716	**0.737**	0.733	**0.737**
	mammographic-masses	0.821	**0.831**	0.827	**0.828**
	nomao	0.983	0.984	0.982	0.982
	primary-tumor	0.675	**0.679**	0.687	**0.693**
	soybean	0.993	0.993	0.993	0.993
	thyroid0387	0.879	0.879	0.877	0.877
NN-2	adult	0.860	0.861	0.861	0.861
	agaricus-lepiota	1.000	1.000	1.000	1.000
	bands	0.830	0.829	0.832	**0.834**
	crx	0.899	0.898	0.900	0.900
	dress-sales	**0.554**	0.548	**0.547**	0.531
	horse-colic	0.684	**0.688**	0.710	**0.719**
	mammographic-masses	0.820	**0.824**	0.825	0.825
	nomao	0.980	0.981	0.982	0.982
	primary-tumor	0.667	**0.669**	0.673	0.674
	soybean	0.986	0.986	0.988	0.988
	thyroid0387	0.878	0.879	**0.877**	0.876
NN-1-D	adult	0.838	0.838	0.838	0.838
	agaricus-lepiota	1.000	1.000	1.000	1.000
	bands	0.871	**0.874**	0.875	0.876
	crx	0.907	0.908	0.908	0.908
	dress-sales	**0.544**	0.537	**0.560**	0.544
	horse-colic	0.743	**0.763**	0.751	**0.756**
	mammographic-masses	0.802	**0.810**	0.806	0.807
	nomao	0.984	0.985	0.983	0.983
	primary-tumor	0.665	**0.669**	0.676	**0.681**
	soybean	0.993	0.993	0.993	0.993
	thyroid0387	0.881	0.880	0.879	0.879
NN-2-D	adult	0.842	0.843	0.843	0.843
	agaricus-lepiota	1.000	1.000	1.000	1.000
	bands	**0.859**	0.857	0.861	**0.862**
	crx	0.898	0.898	0.899	0.900
	dress-sales	**0.548**	0.543	**0.548**	0.535
	horse-colic	0.712	**0.716**	0.731	**0.739**
	mammographic-masses	0.802	**0.806**	0.805	0.806
	nomao	0.981	0.982	0.983	0.983
	primary-tumor	0.659	**0.661**	0.666	0.667
	soybean	0.986	0.986	0.988	0.988
	thyroid0387	0.880	0.880	**0.878**	0.877
SVM-L	adult	0.905	0.905	0.906	0.906
	agaricus-lepiota	1.000	1.000	1.000	1.000
	bands	0.796	**0.797**	0.817	0.817

Continued on next page

Table C.4: AUROC, additional experiment for imputation of categorical attributes (mode imputation or mean imputation after one-hot encoding). **Bold**: higher value by at least 0.001.

Classifier	Dataset	Without missing-indicators		With missing-indicators	
		Mode	Mean	Mode	Mean
	crx	0.922	0.921	0.920	0.920
	dress-sales	**0.598**	0.590	0.593	0.593
	horse-colic	0.790	**0.794**	0.784	0.784
	mammographic-masses	0.865	0.866	0.867	0.867
	nomao	**0.986**	0.984	0.988	0.988
	primary-tumor	0.769	0.769	0.769	0.769
	soybean	0.999	0.999	0.999	0.999
	thyroid0387	0.964	0.965	0.964	0.964
SVM-G	adult	0.895	**0.896**	0.897	0.897
	agaricus-lepiota	1.000	1.000	1.000	1.000
	bands	0.855	0.856	0.865	**0.867**
	crx	0.926	0.925	0.927	0.927
	dress-sales	**0.618**	0.609	**0.620**	0.614
	horse-colic	0.768	**0.774**	0.771	**0.774**
	mammographic-masses	0.840	**0.843**	**0.845**	0.843
	nomao	0.991	0.991	0.992	0.992
	primary-tumor	0.762	**0.764**	0.765	0.766
	soybean	0.999	0.999	0.999	0.999
	thyroid0387	0.978	0.978	0.978	0.978
LR	adult	0.905	0.906	0.906	0.906
	agaricus-lepiota	1.000	1.000	1.000	1.000
	bands	**0.819**	0.814	0.833	0.832
	crx	0.924	0.924	0.923	0.924
	dress-sales	**0.620**	0.611	0.620	0.620
	horse-colic	**0.789**	0.788	0.786	0.787
	mammographic-masses	0.866	0.867	0.868	0.868
	nomao	**0.986**	0.984	0.988	0.988
	primary-tumor	0.773	0.773	0.776	0.776
	soybean	0.999	0.999	0.999	0.999
	thyroid0387	0.974	0.974	0.975	0.975
MLP	adult	0.890	**0.891**	0.890	0.890
	agaricus-lepiota	1.000	1.000	1.000	1.000
	bands	0.871	**0.874**	0.879	**0.882**
	crx	0.902	0.902	0.906	0.906
	dress-sales	**0.549**	0.540	**0.553**	0.549
	horse-colic	0.714	**0.727**	0.744	**0.749**
	mammographic-masses	0.845	0.844	0.840	0.841
	nomao	0.991	0.991	0.991	0.991
	primary-tumor	0.768	**0.769**	**0.782**	0.781
	soybean	0.999	0.999	0.999	0.999
	thyroid0387	0.988	0.989	0.988	0.988
CART	adult	0.844	0.844	0.844	0.844
	agaricus-lepiota	0.991	0.991	0.992	0.991
	bands	**0.749**	0.744	**0.759**	0.757
	crx	0.897	**0.899**	0.897	**0.899**
	dress-sales	0.568	0.568	**0.570**	0.568
	horse-colic	**0.742**	0.728	**0.724**	0.723

Continued on next page

Table C.4: AUROC, additional experiment for imputation of categorical attributes (mode imputation or mean imputation after one-hot encoding). **Bold**: higher value by at least 0.001.

Classifier	Dataset	Without missing-indicators		With missing-indicators	
		Mode	Mean	Mode	Mean
	mammographic-masses	0.823	0.822	**0.823**	0.821
	nomao	0.916	0.916	0.916	0.916
	primary-tumor	0.703	**0.739**	0.707	**0.738**
	soybean	0.990	**0.995**	0.991	**0.995**
	thyroid0387	0.909	0.909	0.909	0.909
RF	adult	0.890	0.891	0.890	0.890
	agaricus-lepiota	1.000	1.000	1.000	1.000
	bands	0.893	**0.895**	**0.896**	0.890
	crx	0.932	0.933	0.931	0.930
	dress-sales	**0.591**	0.589	**0.606**	0.589
	horse-colic	0.800	**0.802**	0.791	**0.795**
	mammographic-masses	0.812	**0.823**	0.821	**0.822**
	nomao	0.994	0.994	0.994	0.994
	primary-tumor	0.749	**0.753**	0.758	0.759
	soybean	0.999	0.999	0.999	0.999
	thyroid0387	0.994	0.994	0.994	0.993
ERT	adult	0.847	**0.848**	0.847	0.847
	agaricus-lepiota	1.000	1.000	1.000	1.000
	bands	0.890	**0.893**	**0.890**	0.889
	crx	0.914	0.914	0.914	0.914
	dress-sales	0.572	**0.589**	**0.602**	0.591
	horse-colic	0.799	**0.806**	0.782	**0.785**
	mammographic-masses	0.795	**0.804**	0.802	0.801
	nomao	0.994	0.994	0.994	0.994
	primary-tumor	0.705	**0.711**	0.714	0.713
	soybean	0.999	0.999	0.999	0.999
	thyroid0387	0.987	0.987	0.987	0.987
ABT	adult	0.915	0.915	0.915	0.915
	agaricus-lepiota	1.000	1.000	1.000	1.000
	bands	0.806	0.806	**0.806**	0.805
	crx	0.905	0.906	**0.906**	0.904
	dress-sales	**0.590**	0.582	**0.582**	0.579
	horse-colic	0.753	**0.763**	0.752	**0.764**
	mammographic-masses	0.856	0.857	0.857	0.858
	nomao	0.987	0.987	0.987	0.987
	primary-tumor	**0.661**	0.640	**0.660**	0.639
	soybean	**0.863**	0.859	0.871	**0.873**
	thyroid0387	0.685	0.685	0.685	0.685
GBM	adult	0.927	0.927	0.927	0.927
	agaricus-lepiota	1.000	1.000	1.000	1.000
	bands	0.855	0.855	**0.857**	0.854
	crx	0.934	0.934	0.933	0.934
	dress-sales	**0.608**	0.606	**0.614**	0.608
	horse-colic	0.789	**0.792**	0.783	**0.788**
	mammographic-masses	0.857	0.857	**0.859**	0.858
	nomao	0.994	0.994	0.994	0.994
	primary-tumor	0.766	**0.770**	0.767	**0.769**

Continued on next page

Table C.4: AUROC, additional experiment for imputation of categorical attributes (mode imputation or mean imputation after one-hot encoding). **Bold**: higher value by at least 0.001.

Classifier	Dataset	Without missing-indicators		With missing-indicators	
		Mode	Mean	Mode	Mean
	soybean	0.999	0.999	0.999	0.999
	thyroid0387	0.913	**0.914**	0.923	0.923

C.3 Polar encoding

The tables in this section correspond to the experiments in Chapter 12.

Table C.5 (distance-based classifiers) and Table C.6 (decision tree classifiers) contain the mean AUROC across five-fold cross-validation and five random states for each classifier and each dataset, for mean imputation with missing-indicators as well as for polar encoding (our proposal).

Table C.5: AUROC obtained with mean imputation and missing-indicators (MI) and polar encoding (PE) for distance-based classifiers. **Bold**: higher value.

		Boscovich		Euclidean	
		MI	PE	MI	PE
Dataset	dataset				
FRNN	adult	0.872	**0.878**	0.863	**0.867**
	agaricus-lepiota	**1.000**	**1.000**	**1.000**	**1.000**
	aps-failure	0.943	**0.952**	0.962	**0.968**
	arrhythmia	**0.889**	0.887	0.868	**0.875**
	bands	0.832	**0.852**	0.819	**0.833**
	ckd	0.996	**0.999**	0.996	**0.999**
	crx	0.918	**0.921**	0.918	**0.920**
	dress-sales	**0.592**	0.577	**0.586**	0.572
	exasens	0.719	**0.745**	0.736	**0.749**
	hcc	0.784	**0.792**	0.769	**0.780**
	heart-disease	0.858	**0.863**	0.848	**0.854**
	hepatitis	0.882	**0.884**	0.879	**0.880**
	horse-colic	0.760	**0.772**	0.766	**0.772**
	mammographic-masses	0.816	**0.838**	0.824	**0.837**
	mi	0.674	**0.687**	0.670	**0.678**
	nomao	0.986	**0.990**	0.987	**0.989**
	primary-tumor	**0.794**	0.790	**0.791**	0.784
	secom	0.642	**0.673**	0.596	**0.609**
	soybean	**0.997**	0.997	0.997	**0.997**
	thyroid0387	0.886	**0.906**	0.892	**0.902**
NN	adult	0.846	**0.846**	0.846	**0.846**
	agaricus-lepiota	**1.000**	**1.000**	**1.000**	**1.000**
	aps-failure	**0.910**	0.909	0.902	**0.904**

Continued on next page

Table C.5: AUROC obtained with mean imputation and missing-indicators (MI) and polar encoding (PE) for distance-based classifiers. **Bold**: higher value.

		Boscovich		Euclidean	
		MI	PE	MI	PE
Dataset	dataset				
	arrhythmia	0.757	**0.757**	**0.733**	0.723
	bands	0.800	**0.824**	0.794	**0.810**
	ckd	0.986	**0.998**	0.985	**0.998**
	crx	0.910	**0.912**	0.910	**0.911**
	dress-sales	**0.560**	0.552	**0.560**	0.548
	exasens	0.717	**0.719**	0.713	**0.717**
	hcc	**0.751**	0.717	**0.733**	0.699
	heart-disease	0.833	**0.841**	0.827	**0.832**
	hepatitis	0.815	**0.818**	0.815	**0.828**
	horse-colic	0.723	**0.728**	0.727	**0.730**
	mammographic-masses	**0.831**	0.830	**0.830**	0.830
	mi	**0.591**	0.575	**0.584**	0.583
	nomao	0.980	**0.982**	0.978	**0.980**
	primary-tumor	**0.719**	0.687	**0.718**	0.697
	secom	0.590	**0.617**	0.522	**0.548**
	soybean	0.988	**0.992**	0.987	**0.990**
	thyroid0387	0.835	**0.836**	0.828	**0.828**
NN-D	adult	0.826	**0.828**	**0.827**	0.827
	agaricus-lepiota	**1.000**	**1.000**	**1.000**	**1.000**
	aps-failure	**0.911**	0.910	0.903	**0.905**
	arrhythmia	0.759	**0.760**	**0.735**	0.726
	bands	0.824	**0.851**	0.808	**0.825**
	ckd	0.987	**0.999**	0.987	**0.999**
	crx	0.906	**0.909**	0.906	**0.909**
	dress-sales	**0.564**	0.552	**0.563**	0.548
	exasens	0.636	**0.637**	0.632	**0.634**
	hcc	**0.762**	0.738	**0.744**	0.720
	heart-disease	0.837	**0.843**	0.832	**0.837**
	hepatitis	**0.823**	0.821	0.819	**0.827**
	horse-colic	0.747	**0.754**	0.745	**0.749**
	mammographic-masses	0.808	**0.808**	0.808	**0.808**
	mi	**0.592**	0.577	**0.586**	0.585
	nomao	0.981	**0.983**	0.979	**0.981**
	primary-tumor	**0.703**	0.679	**0.704**	0.688
	secom	0.594	**0.624**	0.526	**0.549**
	soybean	0.988	**0.992**	0.987	**0.990**
	thyroid0387	0.837	**0.838**	0.830	**0.830**
SVM-G	adult			0.893	**0.900**
	agaricus-lepiota			**1.000**	**1.000**
	aps-failure			0.942	**0.974**
	arrhythmia			0.872	**0.878**
	bands			0.833	**0.843**
	ckd			1.000	**1.000**
	crx			0.920	**0.922**
	dress-sales			**0.632**	0.614
	exasens			0.768	**0.774**

Continued on next page

Table C.5: AUROC obtained with mean imputation and missing-indicators (MI) and polar encoding (PE) for distance-based classifiers. **Bold**: higher value.

		Boscovich		Euclidean	
		MI	PE	MI	PE
Dataset	dataset				
	hcc			**0.800**	0.789
	heart-disease			0.861	**0.869**
	hepatitis			0.857	**0.858**
	horse-colic			0.776	**0.788**
	mammographic-masses			**0.845**	0.835
	mi			0.648	**0.655**
	nomao			0.990	**0.991**
	primary-tumor			0.781	**0.789**
	secom			0.678	**0.696**
	soybean			0.999	**0.999**
	thyroid0387			0.891	**0.917**

Table C.6: AUROC obtained with mean imputation and missing-indicators (MI) and polar encoding (PE) for decision tree classifiers. **Bold**: higher value.

Dataset	ABT		CART		ERT		GBM		RF	
	MI	PE	MI	PE	MI	PE	MI	PE	MI	PE
Dataset	ABT		CART		ERT		GBM		RF	
	MI	PE	MI	PE	MI	PE	MI	PE	MI	PE
adult	0.915	**0.915**	**0.844**	**0.844**	0.847	**0.856**	**0.927**	0.927	0.890	**0.897**
agaricus-lepiota	**1.000**	**1.000**	0.992	**0.992**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
aps-failure	**0.987**	0.987	**0.859**	**0.859**	0.991	**0.991**	**0.988**	0.987	0.989	**0.989**
arrhythmia	0.634	**0.634**	**0.748**	0.745	**0.899**	0.899	**0.852**	0.851	**0.887**	0.885
bands	0.806	**0.813**	0.759	**0.768**	0.890	**0.904**	0.857	**0.859**	**0.896**	0.894
ckd	1.000	**1.000**	**0.967**	0.965	**1.000**	**1.000**	0.998	**0.998**	**1.000**	**1.000**
crx	0.906	**0.908**	0.896	**0.897**	0.914	**0.916**	0.933	**0.933**	0.931	**0.931**
dress-sales	0.581	**0.583**	0.570	**0.574**	**0.602**	0.575	**0.614**	0.608	**0.606**	0.576
exasens	0.720	**0.722**	0.732	**0.743**	0.626	**0.627**	0.757	**0.757**	0.702	**0.707**
hcc	0.725	**0.729**	0.588	**0.590**	**0.808**	0.803	0.745	**0.751**	**0.816**	0.813
heart-disease	**0.860**	0.859	**0.777**	0.774	**0.862**	0.861	**0.871**	0.869	**0.864**	0.858
hepatitis	0.807	**0.809**	0.578	**0.596**	**0.873**	0.857	**0.810**	0.798	**0.886**	0.875
horse-colic	0.756	**0.756**	**0.723**	0.718	0.782	**0.796**	**0.784**	0.783	0.792	**0.798**
mammographic-masses	**0.857**	0.856	**0.823**	0.822	0.802	**0.805**	**0.859**	0.856	0.822	**0.825**
mi	0.572	**0.582**	0.592	**0.607**	0.695	**0.709**	0.637	**0.646**	0.686	**0.696**
nomao	0.987	**0.987**	**0.916**	**0.916**	0.994	**0.994**	**0.994**	0.994	0.994	**0.994**
primary-tumor	**0.660**	0.648	0.707	**0.739**	**0.714**	0.712	**0.767**	0.767	**0.758**	0.756
secom	**0.668**	0.663	**0.500**	**0.500**	0.746	**0.747**	0.679	**0.680**	0.709	**0.722**
soybean	0.870	**0.892**	0.991	**0.993**	0.999	**0.999**	0.999	**0.999**	0.999	**0.999**
thyroid0387	0.685	**0.685**	**0.909**	0.908	0.987	**0.990**	0.923	**0.934**	**0.994**	0.993

Summary

Fuzzy rough nearest neighbour classification (FRNN) is an algorithm that calculates a score for each decision class based on the similarity of a test record with the records in the decision class (upper approximation) and its dissimilarity with the records not in the decision class (lower approximation). We propose three modificiations that make FRNN more practical and turn it into a true nearest neighbour algorithm. We then show that this outperforms classical nearest neighbour classification and determine how datasets should be scaled, which distance measure should be used and how many neighbours should be included in the calculation.

Next, we investigate how FRNN can be applied to very large datasets. We demonstrate this can be achieved through distributed computing, but conclude that this approach has only limited potential. We then show that the computational complexity of FRNN can be reduced substantially by using approximate nearest neighbour searches, allowing us to perform cross-validation on datasets with ten million records.

Upper and lower approximations can be considered to be examples of so-called data descriptors, algorithms that express similarity with a target dataset, a prediction task known as one-class classification. We introduce our own data descriptor, average localised proximity (ALP), and show that it performs better than other data descriptors. We then investigate how the hyperparameters of data descriptors can best be optimised, and apply this in the fuzzy rough one-class ensemble, a generalisation of FRNN where the upper and lower approximations are calculated with different data descriptors than the default. However, we find that this obtains worse classification performance than ordinary FRNN, except for large multiclass datasets with not too many classes.

In the final part of the thesis, we evaluate three approaches towards missing values. We find that including the information from missing-values in the from of indicator attributes increases overall classification performance for a range of algorithms. For distance-based and decision tree algorithms, we present an alternative representation of missing values (polar encoding) that can be applied when numerical values are scaled to $[0, 1]$, and which does not require imputation, and show

that it leads to results that are as good or better than those obtained with missing-indicators. For FRNN, it also performs better than an alternative proposal that represents the uncertainty from missing values with interval-valued fuzzy sets.

Samenvatting

Fuzzy rough nearest neighbours (FRNN) is een classificatie-algoritme dat een score berekent voor iedere beslissingklasse op basis van de mate waarin een testgeval lijkt op de objecten in die beslissingingsklasse (de *bovenbenadering*) en de mate waarin het testgeval niet lijkt op de objecten in de *andere* beslissingklassen (de *onderbenadering*). We stellen drie aanpassingen voor die FRNN beter praktisch toepasbaar maken, en het veranderen in een werkelijk naasteburenalgoritme. We laten dan zien dat dit beter presteert dan klassieke naasteburenclassificatie, en bepalen hoe datasets geschaald dienen te worden, welke afstandsmaat men dient te gebruiken en hoeveel buren men dient te betrekken in de berekening.

Vervolgens onderzoeken we hoe FRNN toegepast kan worden op zeer grote datasets. We laten zien dat dit bereikt kan worden door middel van *distributed computing*, maar concluderen dat het potentieel van deze aanpak slechts beperkt is. We laten dan zien dat de computationele complexiteit van FRNN substantieel verminderd kan worden door een algoritme te gebruiken dat naaste buren slechts bij benadering vindt, hetgeen ons toestaat om kruisvalidatie toe te passen op datasets met tien miljoen regels.

De boven- en onderbenadering kunnen worden beschouwd als voorbeelden van zogenaamde *data-descriptoren*, algoritmes die gelijkenis met een doelklasse ten uitdrukking brengen, een voorspellingsopdracht die éénklasseclassificatie genoemd wordt. We introduceren onze eigen data-descriptor, *average localised proximity* (ALP), en laten zien dat deze beter presteert dan andere data-descriptoren. We onderzoeken vervolgens hoe de hyperparameters van data-descriptoren het beste geoptimaliseerd dienen te worden, en passen dit toe in een vaagruw-éénklasse-ensemble, een veralgemening van FRNN waarbij boven- en onderbenadering berekend worden met andere data-descriptoren dan oorspronkelijk. Echter, we verkrijgen hiermee slechtere classificatieresultaten dan met gewone FRNN, behalve voor grote meerklassedatasets met niet te veel klassen.

In het laatste deel van de thesis evalueren we drie manieren om met ontbrekende waarden om te gaan. We stellen vast dat het toevoegen van de informatie uit ontbrekende waarden in de vorm van indicatoren

de classificatieresultaten over het algemeen verbetert voor een reeks aan algoritmes. Voor algoritmes gebaseerd op afstand en beslisbomen stellen we ook een alternatieve representatie voor van ontbrekende waarden (*polar encoding*) die toegepast kan worden wanneer numerieke waarden naar $[0, 1]$ geschaald worden, en die geen imputatie behoeft, en we laten zien dat dit tot resultaten leidt die even goed of beter zijn dan de resultaten verkregen met indicatoren. Voor FRNN presteert deze aanpak ook beter dan een alternatief voorstel dat de onzekerheid van ontbrekende waarden representeert met behulp van intervalwaardige vaagverzamelingen.

Acknowledgements

I would like to thank, first of all, my two supervisors Chris and Dani, for providing me with guidance and criticism, for allowing me to wander off into the weeds of one-class classification and the representation of missing values when those topics piqued my interest, for indulging my insistence on using the phrase *Boscovich distance*[1], for hiring me in the first place despite my not being quite so fresh out of college, and most of all — for being really nice people to work with.

Thanks also to my direct colleagues Marko and Olha, with whom I could discuss theory and occasionally share frustration, and to my direct predecessor Sarah, who very generously provided me with code and explanations, and whose thesis (Vluymans 2018) was a very helpful resource.

I would also like to thank Yvan for letting me join his weekly machine learning group, and the other members for presenting many interesting updates, in particular Arne, Jonathan and Maxim.

My papers have profited from a number of useful comments by anonymous reviewers, often bringing literature to my attention that I had missed, like the work of Jensen & Shen (2009) and Couso & Dubois (2011), proposals that are similar to Chapter 11. I would in particular like to thank James McDermott for a number of useful comments and suggestions on Lenz et al (2021b), adapted here as Chapter 7, and the members of the examination committee for their comments on the initial version of this thesis.

I am also grateful to the many volunteers who have built the Python and LaTeX packages I have used, and those who have shared their knowledge on Stack Overflow and TeX.StackExchange.

I have had a great time at our department, even if we had to work from home for long stretches of time, due to the very nice group of

[1] It is nice to credit people by naming things after them. *1-distance* is perhaps the most descriptive name, but it doesn't appear to be used much. L^1, L_1, L1, ℓ^1, ℓ_1 are all used, but I haven't been able to figure out which letter shape to use, where to place the 1, and whether these shouldn't be preserved for function spaces anyway. *Manhattan* distance is used, but it seems to be a relatively recent substitute for *city-block* distance, while Wikipedia sticks to *taxicab geometry*, and *rectilinear* and *right-angle* can also be found in the literature.

people who work there. At the risk of forgetting anyone, these are my office mates and lunch companions throughout the years, Adnan, Camila, Christophe, Elke, Fatemeh, Georgi, Hans, Henri, Hege, Jeroen, Josephina, Kelly, Koen, Maria, Marko, Martina, Milan, Muluneh, Olha, Oliver, Pawel, Slađa, Stijn, Thang, William, Wout and Yao, as well as the people with whom I have shared many coffee breaks (and whom I have occasionally joined for lunch), Alexis, Asmus, Bart, Charlotte, Felix, Heidi, Jonathan, Nico, Niko, Rien, Robbert, Roy, Steven and Toon. Thanks also to the other professors, always approachable and sociable, despite their large workloads, and to our friendly administrative and technical staff.

I remain slightly frustrated by the enduring, unspoken (yet absolutely cordial) division between the two groups of doctoral students in our department[2], and I quietly exulted when on two occasions, in the week before the *Gentse Feesten* last summer and on the first day of office this January, there were so few people present that we ended up going to the *resto* as one group.[3]

Outside of the office, I have greaty enjoyed playing tennis with Camila, Marko and Oliver, climbing with Georgi, Martina and Pawel, playing board games and having dinner with Camila, Marko, Martina, Olha, Pawel, Slađa, Viacheslav and Vitalii, and going for coffee walks with Pawel in the period when we were still mostly working from home.

Thanks also to my friends who have undertaken the journey to Ghent to visit us — every time a joyous occasion — and to my brother David for his continuous supply of interesting and funny stories.

Finally, my greatest appreciation goes to Joanne, my wife and personal authority on succesfully completing a doctorate, and our son Ivar. They have made working from home for months on end tolerable, and have been very accomodating whenever I have had to work long days and nights, especially in these final few months.

[2] 'Computer scientists' and 'statisticians', although these labels do not fit perfectly. There is actually a third group, the students of Yvan Saeys, many of whom spend most of their time at the VIB.

[3] There are of course entirely practical reasons for having two lunch groups — it is hard enough as it is to find a free table for up to a dozen people.

Publications

Journal papers

Lenz OU, Peralta D, Cornelis C (2020b)

Scalable approximate FRNN-OWA classification. IEEE Transactions on Fuzzy Systems, vol 28, no 5, pp 929–938.

———, Peralta D, Cornelis C (2021b)

Average Localised Proximity: a new data descriptor with good default one-class classification performance. Pattern Recognition, vol 118, p 107991.

———, Peralta D, Cornelis C (2022c)

Optimised one-class classification performance. Machine Learning, vol 111, no 8, pp 2863–2883.

Theerens A, Lenz OU, Cornelis C (2022)

Choquet-based fuzzy rough sets. International Journal of Approximate Reasoning, vol 146, pp 62–78.

Conference papers

Lenz OU, Peralta D, Cornelis C (2019)

A scalable approach to fuzzy rough nearest neighbour classification with ordered weighted averaging operators. IJCRS 2019: Proceedings of the International Joint Conference on Rough Sets. Lecture Notes in Artificial Intelligence 11499. Debrecen, Hungary: Springer, pp 197–209.

———, Peralta D, Cornelis C (2020a)

fuzzy-rough-learn 0.1: a Python library for machine learning with fuzzy rough sets. IJCRS 2020: Proceedings of the International Joint Conference on Rough Sets. Lecture Notes in Artificial Intelligence 12179. Springer, pp 491–499.

———, Peralta D, Cornelis C (2021a)

Adapting fuzzy rough sets for classification with missing values. IJCRS 2021: Proceedings of the International Joint Conference on Rough Sets. Lecture Notes in Artificial Intelligence 12872. Springer, pp 192–200.

———, Cornelis C, Peralta D (2022a)

fuzzy-rough-learn 0.2: a Python library for fuzzy rough set algorithms and one-class classification. FUZZ-IEEE 2022: Proceedings of the IEEE International Conference on Fuzzy Systems. IEEE.

Unpublished manuscripts

Lenz OU, Peralta D, Cornelis C (2022b)
No imputation without representation. arXiv preprint 2206.14254.

———, Peralta D, Cornelis C (2022d)
Representing missing values through polar encoding. arXiv preprint 2210.01905.

Bibliography

Abbas N, Chibani Y, Belhadi Z, Hedir M (2013)

A DSmT based combination scheme for multi-class classification. Proceedings of the 16th International Conference on Information Fusion. IEEE, pp 1950–1957.

Abdallah L, Badarna M, Khalifa W, Yousef M (2021)

MultiKOC: multi-one-class classifier based k-means clustering. Algorithms, vol 14, no 5, p 134.

Aeberhard S, Coomans D, Vel O de (1992)

Extensions To Regularised Discriminant Analysis. Technical report 92-1. Townsville: James Cook University, Department of Computer Science.

Agarwal S, Sureka A (2015)

Using KNN and SVM based one-class classifier for detecting online radicalization on Twitter. ICDCIT 2015: Proceedings of the 11th International Conference on Distributed Computing and Internet Technology. Lecture Notes in Computer Science 8956. Springer, pp 431–442.

Akaike H (1971)

Information theory and an extension of the maximum likelihood principle. Proceedings of the 2nd International Symposium on Information Theory. Akadémiai Kiadó, pp 267–281.

Allardice RE (1891)

The barycentric calculus of Möbius. Proceedings of the Edinburgh Mathematical Society, vol 10, pp 2–21.

Allison PD (2001)

Missing Data. Thousand Oaks, California: Sage Publications.

——— (2010)

Missing data. Handbook of Survey Research. Ed. by PV Marsden, JD Wright. Second. Bingley, England: Emerald Group Publishing. Chap. 20, pp 631–657.

Amiri M, Jensen R (2016)

Missing data imputation using fuzzy-rough methods. Neurocomputing, vol 205, pp 152–164.

Anderson AB, Basilevsky A, Hum DPJ (1983)
Missing data: a review of the literature. Handbook of Survey Research. Ed. by PH Rossi, JD Wright, AB Anderson. Quantitive Studies in Social Relations. New York: Academic Press. Chap. 12, pp 415–494.

Anderson E (1928)
The problem of species in the northern blue flags, Iris versicolor L. and Iris virginica L. Annals of the Missouri Botanical Garden, vol 15, no 3, pp 241–332.

——— (1935)
The irises of the Gaspé Peninsula. Bulletin of the American Iris Society, vol 59, pp 2–5.

Andoni A, Indyk P (2017)
Nearest neighbors in high-dimensional spaces. Handbook of discrete and computational geometry. Ed. by JE Goodman, J O'Rourke, CD Tóth. Third. Boca Raton: Chapman and Hall/CRC. Chap. 43, pp 1135–1155.

Antal B, Hajdu A (2014)
An ensemble-based system for automatic screening of diabetic retinopathy. Knowledge-Based Systems, vol 60, pp 20–27.

Antal M, Szabó LZ (2015)
An evaluation of one-class and two-class classification algorithms for keystroke dynamics authentication on mobile devices. CSCS 2015: Proceedings of the 20th International Conference on Control Systems and Computer Science. IEEE, pp 343–350.

Asfoor HM, Srinivasan R, Vasudevan G, Verbiest N, Cornelis C, Tolentino M, Teredesai A, De Cock M (2014)
Computing fuzzy rough approximations in large scale information systems. Proceedings of the Second IEEE International Conference on Big Data.

——— (2015)
Fuzzy rough set approximations in large scale information systems. MA thesis. University of Washington.

Aste M, Boninsegna M, Freno A, Trentin E (2015)
Techniques for dealing with incomplete data: a tutorial and survey. Pattern Analysis and Applications, vol 18, no 1, pp 1–29.

Babu MSP, Ramana BV, Kumar BRS (2010)
New automatic diagnosis of liver status using bayesian classification. ICINC 2010: Proceedings of the International Conference on Intelligent Network and Computing. Vol. 2. IEEE, pp 385–388.

Bailey T, Jain A (1978)
A note on distance-weighted k-nearest neighbor rules. IEEE Transactions on Systems, Man, and Cybernetics, vol 8, no 4, pp 311–313.

Baldi P, Sadowski P, Whiteson D (2014)

Searching for exotic particles in high-energy physics with deep learning. Nature Communications, vol 5, p 4308.

———, Cranmer K, Faucett T, Sadowski P, Whiteson D (2016)

Parameterized neural networks for high-energy physics. European Physical Journal C, vol 76, no 5, p 235.

Ban T, Abe S (2006)

Implementing multi-class classifiers by one-class classification methods. IJCNN 2006: Proceedings of the IEEE International Joint Conference on Neural Networks. IEEE, pp 327–332.

Bayer C, Enge-Rosenblatt O, Bator M, Mönks U (2013)

Sensorless drive diagnosis using automated feature extraction, significance ranking and reduction. ETFA 2013: Proceedings of the 18th IEEE International Conference on Emerging Technologies & Factory Automation (ETFA).

Bekker J, Davis J (2020)

Learning from positive and unlabeled data: a survey. Machine Learning, vol 109, no 4, pp 719–760.

Bellman R, Kalaba R, Zadeh L (1964)

Abstraction and pattern classification. Technical report RM-4307-PR. Santa Monica, California: The Rand Corporation.

Benavoli A, Corani G, Mangili F (2016)

Should we really use post-hoc tests based on mean-ranks? Journal of Machine Learning Research, vol 17, no 5, pp 152–161.

Bennett KP, Mangasarian OL (1990)

Neural network training via linear programming. vol, no 948.

———, Mangasarian OL (1991)

Robust Linear Programming Discrimination of Two Linearly Inseparable Sets. Technical report 1054. University of Wisconsin – Madison, Department of Computer Sciences.

——— (1992)

Decision tree construction via linear programming. Technical report 1067. University of Wisconsin – Madison, Department of Computer Sciences.

———, Mangasarian OL (1992)

Multicategory discrimination via linear programming. Technical report 1127. University of Wisconsin – Madison, Department of Computer Sciences.

——— (1993)

Machine learning via mathematical programming. Doctoral thesis. Technical Report 1167. University of Wisconsin – Madison, Department of Computer Sciences.

Bentley JL (1975)

Multidimensional binary search trees used for associative searching. Communications of the ACM, vol 18, no 9, pp 509–517.

Bergstra J, Bardenet R, Bengio Y, Kégl B (2011)

Algorithms for hyper-parameter optimization. NIPS 2011: Proceedings of the 25th Annual Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems 24. NIPS, pp 2546–2554.

———, Bengio Y (2012)

Random search for hyper-parameter optimization. Journal of Machine Learning Research, vol 13, no 10, pp 281–305.

Bernhardsson E (2018)

New approximate nearest neighbor benchmarks. URL: `https://erikbern.com/2018/06/17/new-approximate-nearest-neighbor-benchmarks.html`.

Betrò B (1991)

Bayesian methods in global optimization. Journal of Global Optimization, vol 1, no 1, pp 1–14.

Bezdek JC, Harris JD (1978)

Fuzzy partitions and relations; an axiomatic basis for clustering. Fuzzy Sets and Systems, vol 1, no 2, pp 111–127.

———, Keller JM, Krishnapuram R, Kuncheva LI, Pal NR (1999)

Will the real iris data please stand up? IEEE Transactions on Fuzzy Systems, vol 7, no 3, pp 368–369.

Bhatt RB, Dhall A, Sharma G, Chaudhury S (2009)

Efficient skin region segmentation using low complexity fuzzy decision tree model. INDICON 2009: Proceedings of the Annual IEEE India Conference, pp 1–4.

Blank J, Deb K (2020)

Pymoo: multi-objective optimization in Python. IEEE Access, vol 8, pp 89497–89509.

Bock RK, Chilingarian A, Gaug M, Hakl F, Hengstebeck T, Jiřina M, Klaschka J, Kotrč E, Savický P, Towers S, Vaiciulis A, Wittek W (2003)

Methods for multidimensional event classification: a case study. Technical report 887. Prague: Academy of Sciences of the Czech Republic, Institute of Computer Science.

Boscovich RJ (1757)

De litteraria expeditione per pontificiam ditionem. De bononiensi scientarium et artium instituto atque academia commentarii, vol 4, pp 353–396 (opuscula).

——— (1760)

De recentissimis graduum dimensionibus, et figura, ac magnitudine terræ inde derivanda. Philosophiæ recentioris, *B. Stay*. Vol. 2. Rome: Nicolaus et Marcus Palearini, pp 406–426.

Boyer CB (1956)
History of analytic geometry. New York: Scripta Mathematica. Chap. 9. The Golden Age, pp 242–243.

Boytsov L, Naidan B (2013)
Engineering efficient and effective non-metric space library. SISAP 2013: Proceedings of the 6th International Conference on Similarity Search and Applications. Lecture Notes in Computer Science 8199, pp 280–293.

Bradley AP (1997)
The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition, vol 30, no 7, pp 1145–1159.

Breiman L, Friedman JH, Olshen RA, Stone CJ (1984)
Classification and Regression Trees. The Wadsworth statistics/probability series. Monterey, California: Wadsworth.

——— (2001)
Random forests. Machine Learning, vol 45, no 1, pp 5–32.

Van Breukelen M, Duin RPW, Tax DMJ, Den Hartog JE (1997)
Combining classifiers for the recognition of handwritten digits. Proceedings of the 1st IAPR TC1 Workshop on Statistical Techniques in Pattern Recognition, pp 13–18.

Breunig MM, Kriegel HP, Ng RT, Sander J (2000)
LOF: identifying density-based local outliers. SIGMOD 2000: Proceedings of the ACM international conference on Management of data. SIGMOD Record vol 29, no 2. ACM, pp 93–104.

Brochu E, Cora VM, Freitas N de (2009)
A Tutorial on Bayesian Optimization of Expensive Cost Functions, with Application to Active User Modeling and Hierarchical Reinforcement Learning. Technical report UBC TR-2009-023. University of British Columbia, Department of Computer Science.

Brodatz P (1966)
Textures: a photographic album for artists and designers. New York: Dover Publications.

Buscema M (1998)
MetaNet: the theory of independent judges. Substance Use & Misuse, vol 33, no 2, pp 439–461.

Van Buuren S, Groothuis-Oudshoorn K (2011)
mice: multivariate imputation by chained equations in R. Journal of Statistical Software, vol 45, no 3, pp 1–67.

Candillier L, Lemaire V (2012)
Design and analysis of the nomao challenge: active learning in the real-world. ECML-PKDD 2012: Active Learning in Real-world Applications Workshop.

Cao VL, Nicolau M, McDermott J (2019)
Learning neural representations for network anomaly detection. IEEE Transactions on Cybernetics, vol 49, no 8, pp 3074–3087.

Catlett J (1991)
Megainduction: a test flight. ML91: Proceedings of the Eighth International Workshop on Machine Learning. Morgan Kaufmann, pp 596–599.

Cattral R, Oppacher F, Deugo D (2002)
Evolutionary data mining with automatic rule generalization. Recent Advances in Computers, Computing and Communications, vol 1, no 1, pp 296–300.

Cestnik B, Kononenko I, Bratko I (1987)
ASSISTANT 86: a knowledge-elicitation tool for sophisticated users. EWSL 87: Proceedings of the 2nd European Working Session on Learning. Sigma Press, pp 31–45.

Charytanowicz M, Niewczas J, Kulczycki P, Kowalski PA, Łukasik S, Żak S (2010)
Complete gradient clustering algorithm for features analysis of X-ray images. Information technologies in biomedicine. Ed. by E Piętka, J Kawa. Springer, pp 15–24.

Chaudhuri K, Dasgupta S (2014)
Rates of convergence for nearest neighbor classification. Advances in Neural Information Processing Systems, vol 27.

Chen T, Guestrin C (2016)
XGBoost: a scalable tree boosting system. KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 785–794.

Chen WC, Schmidt JN, Yan D, Vohra YK, Chen CC (2021)
Machine learning and evolutionary prediction of superhard bcn compounds. npj Computational Materials, vol 7, no 114.

Chow WK (1979)
A look at various estimators in logistic models in the presence of missing values. Technical report N-1324-HEW. Santa Monica, California: The Rand Corporation.

CIA World Factbook (2022)
GDP — composition, by sector of origin. URL: `https://www.cia.gov/the-world-factbook/field/gdp-composition-by-sector-of-origin/`.

Cohen J (1968)
Multiple regression as a general data-analytic system. Psychological Bulletin, vol 70, no 6, pp 426–443.

———, Cohen P (1975)
Applied multiple regression/correlation analysis for the behavioral sciences. Hills-

dale, New Jersey: Lawrence Erlbaum Associates. Chap. 7. Missing Data, pp 265–290.

Coomans D, Jonckheer MH, Massart DL, Broeckaert I, Blockx P (1978)

The application of linear discriminant analysis in the diagnosis of thyroid diseases. Analytica Chimica Acta, vol 103, no 4, pp 409–415.

Cornelis C, Verbiest N, Jensen R (2010)

Ordered weighted average based fuzzy rough sets. RSKT 2010: Proceedings of the 5th International Conference on Rough Set and Knowledge Technology. Lecture Notes in Artificial Intelligence 6401. Springer, pp 78–85.

Cortes C, Vapnik V (1995)

Support-vector networks. Machine Learning, vol 20, no 3, pp 273–297.

Couso I, Dubois D (2011)

Rough sets, coverings and incomplete information. Fundamenta Informaticae, vol 108, no 3-4, pp 223–247.

Cover T, Hart P (1967)

Nearest neighbor pattern classification. IEEE Transactions on Information Theory, vol 13, no 1, pp 21–27.

Cox DR (1966)

Some procedures connected with the logistic qualitative response curve. in (fn david, ed.) research papers in statistics: essays in honour of j. neyman's 70th birthday. Research Papers in Statistics: Festschrift for J. Neyman. Ed. by FN David. London: John Wiley & Sons, pp 55–71.

Dai J (2013)

Rough set approach to incomplete numerical data. Information Sciences, vol 241, pp 43–57.

Das S, Datta S, Chaudhuri BB (2018)

Handling data irregularities in classification: foundations, trends, and future challenges. Pattern Recognition, vol 81, pp 674–693.

De Stefano C, Maniaci M, Fontanella F, Scotto di Freca A (2018)

Reliable writer identification in medieval manuscripts through page layout features: the "Avila" Bible case. Engineering Applications of Artificial Intelligence, vol 72, pp 99–110.

D'eer L, Verbiest N, Cornelis C, Godo L (2015)

A comprehensive study of implicator–conjunctor-based and noise-tolerant fuzzy rough sets: definitions, properties and robustness analysis. Fuzzy Sets and Systems, vol 275, pp 1–38.

Demšar J (2006)

Statistical comparisons of classifiers over multiple data sets. Journal of Machine Learning Research, vol 7, no 1, pp 1–30.

Detrano R, Yiannikas J, Salcedo EE, Rincon G, Go RT, Williams G, Leatherman J (1984)

Bayesian probability analysis: a prospective demonstration of its clinical utility in diagnosing coronary disease. Circulation, vol 69, no 3, pp 541–547.

———, Janosi A, Steinbrunn W, Pfisterer M, Schmid JJ, Sandhu S, Guppy KH, Lee S, Froelicher V (1989)

International application of a new probability algorithm for the diagnosis of coronary artery disease. The American Journal of Cardiology, vol 64, no 5, pp 304–310.

Ding Y, Simonoff JS (2010)

An investigation of missing data methods for classification trees applied to binary response data. Journal of Machine Learning Research, vol 11, no 1, pp 131–170.

Dixon JK (1979)

Pattern recognition with partly missing data. IEEE Transactions on Systems, Man, and Cybernetics, vol 9, no 10, pp 617–621.

Domingues R, Filippone M, Michiardi P, Zouaoui J (2018)

A comparative evaluation of outlier detection algorithms: experiments and analyses. Pattern Recognition, vol 74, pp 406–421.

Dörksen H, Lohweg V (2014)

Combinatorial refinement of feature weighting for linear classification. ETFA '14: Proceedings of the 19th IEEE International Conference on Emerging Technologies and Factory Automation.

———, Mönks U, Lohweg V (2014)

Fast classification in industrial big data environments. ETFA '14: Proceedings of the 19th IEEE International Conference on Emerging Technologies and Factory Automation.

Draper BA, Collins RT, Brolio J, Hanson AR, Riseman EM (1989)

The Schema System. International Journal of Computer Vision, vol 2, no 3, pp 209–250.

Dua D, Graff C (2019)

UCI Machine Learning Repository. URL: `http://archive.ics.uci.edu/ml`.

Dubois D, Prade H (1987)

Twofold fuzzy sets and rough sets — some issues in knowledge representation. Fuzzy Sets and Systems, vol 23, no 1, pp 3–18.

———, Prade H (1990)

Rough fuzzy sets and fuzzy rough sets. International Journal of General Systems, vol 17, no 2-3, pp 191–209.

———, Prade H (2005)

Interval-valued fuzzy sets, possibility theory and imprecise probability. EUSFLAT-LFA 2005: Proceedings of the Joint 4th Conference of the European Society

for Fuzzy Logic and Technology and the 11th Rencontres Francophones sur la Logique Floue et ses Applications, pp 314–319.

Dudani SA (1973)

An experimental study of moment methods for automatic identification of three-dimensional objects from television images. Doctoral thesis. The Ohio State University.

——— (1976)

The distance-weighted k-nearest-neighbor rule. IEEE Transactions on Systems, Man, and Cybernetics, vol 6, no 4, pp 325–327.

Dunn JC (1974)

A fuzzy relative of the ISODATA process and its use in detecting compact well-separated clusters. Journal of Cybernetics, vol 3, no 3, pp 32–57.

Efron B, Gong G (1981)

Statistical theory and the computer. Computer Science and Statistics: Proceedings of the 13th Symposium on the Interface. Springer, pp 3–7.

Eirola E (2014)

Machine learning methods for incomplete data and variable selection. Doctoral thesis. Aalto University, Espoo.

Eisenhart C (1961)

Boscovich and the combination of observations. Roger Joseph Boscovich, S.J., F.R.S., 1711–1787: Studies of his Life and Work on the 250th Anniversary of his Birth. Ed. by LL Whyte. London: George Allen & Unwin. Chap. 9, pp 200–212.

Elter M, Schulz-Wendtland R, Wittenberg T (2007)

The prediction of breast cancer biopsy outcomes using two CAD approaches that both emphasize an intelligible decision process. Medical Physics, vol 34, no 11, pp 4164–4172.

Enders CK (2010)

Applied Missing Data Analysis. Methodology in the Social Sciences. New York: The Guilford Press.

Esposito F, Malerba D, Semeraro G, Annese E, Scafuro G (1990)

An experimental page layout recognition system for office document automatic classification: an integrated approach for inductive generalization. Proceedings of the 10th International Conference on Pattern Recognition. Vol. 1. IEEE, pp 557–562.

———, Malerba D, Semeraro G (1995)

A knowledge-based approach to the layout analysis. ICDAR '95: Proceedings of Third International Conference on Document Analysis and Recognition. Vol. I. IEEE Computer Society Press, pp 466–471.

Estrela da Silva J, Marques de Sá JP, Jossinet J (2000)

Classification of breast tissue by electrical impedance spectroscopy. Medical and Biological Engineering and Computing, vol 38, no 1, pp 26–30.

Evans B, Fisher D (1994)

Overcoming process delays with decision tree induction. IEEE Expert, vol 9, no 1, pp 60–66.

Evett IW, Spiehler EJ (1987)

Rule induction in forensic science. Proceedings of the KBS in Government conference. Online Publications, pp 107–118.

Ferguson DJ, Meier P (1976)

Results of the treatment of mammary cancer at the university of chicago, 1960–1969. Surgical Clinics of North America, vol 56, no 1, pp 103–109.

Ferreira Costa C, Nascimento MA (2016)

IDA 2016 industrial challenge: using machine learning for predicting failures. IDA 2016: Proceedings of the 15th International Symposium on Intelligent Data Analysis. Lecture Notes in Computer Science 9897. Springer, pp 381–386.

Fisher RA (1936)

The use of multiple measurements in taxonomic problems. Annals of Eugenics, vol 7, no 2, pp 179–188.

Fix E, Hodges Jr J (1951)

Discriminatory Analysis — Nonparametric Discrimination: Consistency Properties. Technical report 21-49-004. Randolph Field, Texas: USAF School of Aviation Medicine.

Forina M, Lanteri S (1984)

Data analysis in food chemistry. Chemometrics. Springer, pp 305–349.

———, Armanino C, Castino M, Ubigli M (1986)

Multivariate data analysis as a discriminating method of the origin of wines. Vitis, vol 25, no 3, pp 189–201.

Fragoso RC, Cavalcanti GD, Pinheiro RH, Oliveira LS (2021)

Dynamic selection and combination of one-class classifiers for multi-class classification. Knowledge-Based Systems, vol 228, p 107290.

Freund Y, Schapire RE (1995)

A desicion-theoretic generalization of on-line learning and an application to boosting. EuroCOLT '95: Proceedings of the Second European Conference on Computational Learning Theory. Lecture Notes in Computer Science 904. Springer, pp 23–37.

Frey PW, Slate DJ (1991)

Letter recognition using Holland-style adaptive classifiers. Machine Learning, vol 6, no 2, pp 161–182.

Friedman JH, Bentley JL, Finkel RA (1977)

An algorithm for finding best matches in logarithmic expected time. ACM Transactions on Mathematical Software, vol 3, no 3, pp 209–226.

——— (2001)

Greedy function approximation: a gradient boosting machine. The Annals of Statistics, vol 29, no 5, pp 1189–1232.

Fukushima K (1969)

Visual feature extraction by a multilayered network of analog threshold elements. IEEE Transactions on Systems Science and Cybernetics, vol 5, no 4, pp 322–333.

García S, Luengo J, Herrera F (2015)

Data preprocessing in data mining. Intelligent Systems Reference Library 72. Cham, Zug: Springer. Chap. 4. Dealing with Missing Values.

Gautam C, Tiwari A, Ravindran S (2016)

Construction of multi-class classifiers by extreme learning machine based one-class classifiers. 2016 International Joint Conference on Neural Networks (IJCNN). IEEE, pp 2001–2007.

Geurts P, Ernst D, Wehenkel L (2006)

Extremely randomized trees. Machine Learning, vol 63, no 1, pp 3–42.

Giacinto G, Perdisci R, Del Rio M, Roli F (2008)

Intrusion detection in computer networks by a modular ensemble of one-class classifiers. Information Fusion, vol 9, no 1, pp 69–82.

Glorot X, Bengio Y (2010)

Understanding the difficulty of training deep feedforward neural networks. AISTATS 2010: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research 9. JMLR Workshop and Conference Proceedings, pp 249–256.

Goh KS, Chang EY, Li B (2005)

Using one-class and two-class svms for multiclass image annotation. IEEE Transactions on Knowledge and Data Engineering, vol 17, no 10, pp 1333–1346.

Golovenkin SE, Bac J, Chervov A, Mirkes EM, Orlova YV, Barillot E, Gorban AN, Zinovyev A (2020)

Trajectories, bifurcations, and pseudo-time in large clinical datasets: applications to myocardial infarction and diabetes data. GigaScience, vol 9, no 11, giaa128.

Gorman RP, Sejnowski TJ (1988)

Analysis of hidden units in a layered network trained to classify sonar targets. Neural Networks, vol 1, no 1, pp 75–89.

Graham JW (2009)

Missing data analysis: making it work in the real world. Annual Review of Psychology, vol 60, pp 549–576.

Grzymala-Busse JW (1988)
Knowledge acquisition under uncertainty—a rough set approach. Journal of Intelligent and Robotic Systems, vol 1, no 1, pp 3–16.

———, Hu M (2000)
A comparison of several approaches to missing attribute values in data mining. RSCTC 2000: Proceedings of the Second International Conference on Rough Sets and Current Trends in Computing. Lecture Notes in Artificial Intelligence 2005. Springer, pp 378–385.

——— (2006)
Rough set strategies to data with missing attribute values. Foundations and novel approaches in data mining. Springer, pp 197–212.

Güvenir HA, Acar B, Demiröz G, Çekin A (1997)
A supervised machine learning algorithm for arrhythmia analysis. Proceedings of the 24th Annual Meeting of Computers in Cardiology. Computers in Cardiology 24. IEEE, pp 433–436.

———, Demiröz G, İlter N (1998)
Learning differential diagnosis of erythemato-squamous diseases using voting feature intervals. Artificial Intelligence in Medicine, vol 13, no 3, pp 147–165.

Haberman SJ (1976)
Generalized residuals for log-linear models. Proceedings of the 9th International Biometrics Conference. Invited Papers I. The Biometric Society, pp 104–122.

Hadjadji B, Chibani Y, Guerbai Y (2014)
Multiple one-class classifier combination for multi-class classification. 2014 22nd International Conference on Pattern Recognition. IEEE, pp 2832–2837.

———, Chibani Y, Guerbai Y (2017)
Combining diverse one-class classifiers by means of dynamic weighted average for multi-class pattern classification. Intelligent Data Analysis, vol 21, no 3, pp 515–535.

———, Chibani Y (2018)
Two combination stages of clustered one-class classifiers for writer identification from text fragments. Pattern Recognition, vol 82, pp 147–162.

———, Chibani Y, Nemmour H (2019)
Hybrid one-class classifier ensemble based on fuzzy integral for open-lexicon handwritten arabic word recognition. Pattern Analysis and Applications, vol 22, no 1, pp 99–113.

Hajj N, Rizk Y, Awad M (2019)
A subjectivity classification framework for sports articles using improved cortical algorithms. Neural Computing and Applications, vol 31, no 11, pp 8069–8085.

Haldemann J, Ksoll V, Walter D, Alibert Y, Klessen RS, Benz W, Koethe U, Ardizzone L, Rother C (2022)

Exoplanet Characterization using Conditional Invertible Neural Networks. arXiv preprint 2202.00027.

Hall M, Frank E, Holmes G, Pfahringer B, Reutemann P, Witten IH (2009)
The WEKA data mining software: an update. ACM SIGKDD Explorations Newsletter, vol 11, no 1, pp 10–18.

Hamming RW (1950)
Error detecting and error correcting codes. The Bell System Technical Journal, vol 29, no 2, pp 147–160.

Hand DJ, Till RJ (2001)
A simple generalisation of the area under the ROC curve for multiple class classification problems. Machine Learning, vol 45, no 2, pp 171–186.

Hanley JA, McNeil BJ (1982)
The meaning and use of the area under a receiver operating characteristic (ROC) curve. Radiology, vol 143, no 1, pp 29–36.

Hariri S, Carrasco Kind M, Brunner RJ (2021)
Extended Isolation Forest. IEEE Transactions on Knowledge and Data Engineering, vol 33, no 4, pp 1479–1489.

Hayashi T, Fujita H (2021)
One-class ensemble classifier for data imbalance problems. Applied Intelligence, vol, pp 1–17.

Heidke P (1926)
Berechnung des erfolges und der güte der windstärkevorhersagen im sturmwarnungsdienst. Geografiska Annaler, vol 8, no 4, pp 301–349.

Van der Heijden GJMG, Donders ART, Stijnen T, Moons KGM (2006)
Imputation of missing values is superior to complete case analysis and the missing-indicator method in multivariable diagnostic research: a clinical example. Journal of Clinical Epidemiology, vol 59, no 10, pp 1102–1109.

Ho TK, Basu M, Law MHC (2006)
Measures of geometrical complexity in classification problems. Data complexity in pattern recognition. Springer, pp 1–23.

Holm S (1979)
A simple sequentially rejective multiple test procedure. Scandinavian Journal of Statistics, vol 6, no 2, pp 65–70.

Hong TP, Tseng LH, Chien BC (2010)
Mining from incomplete quantitative data by fuzzy rough sets. Expert Systems with Applications, vol 37, no 3, pp 2644–2653.

Hooke R, Jeeves TA (1961)
''direct search'' solution of numerical and statistical problems. Journal of the ACM, vol 8, no 2, pp 212–229.

Horton P, Nakai K (1996)
A probabilistic classification system for predicting the cellular localization sites of proteins. ISMB-96: Proceedings of the Fourth International Conference on Intelligent Systems for Molecular Biology. AAAI, pp 109–115.

Hu Q, Zhang L, Zhou Y, Pedrycz W (2018)
Large-scale multimodality attribute reduction with multi-kernel fuzzy rough sets. IEEE Transactions on Fuzzy Systems, vol 26, no 1, pp 226–238.

Huang C, Rice DR, Steffen JH (2022)
MAGRATHEA: an open-source spherical symmetric planet interior structure code. Monthly Notices of the Royal Astronomical Society, vol 513, no 4, pp 5256–5269.

Hutcheson Jr JD, Prather JE (1981)
Interpreting the effects of missing data in survey research. Southeastern Political Review, vol 9, no 2, pp 129–143.

Indyk P, Motwani R (1998)
Approximate nearest neighbors: towards removing the curse of dimensionality. STOC '98: Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing. ACM Press, pp 604–613.

Ipsen N, Mattei PA, Frellsen J (2020)
How to deal with missing data in supervised deep learning? Artemiss 2020: First ICML Workshop on the Art of Learning with Missing Values.

Jain AK, Duin RPW, Mao J (2000)
Statistical pattern recognition: a review. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 22, no 1, pp 4–37.

Janssens JHM, Flesch I, Postma EO (2009)
Outlier detection with one-class classifiers from ML and KDD. ICMLA 2009: Proceedings of the Eighth International Conference on Machine Learning and Applications. IEEE, pp 147–153.

Jégou H, Douze M, Schmid C (2011)
Product quantization for nearest neighbor search. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 33, no 1, pp 117–128.

Jensen R, Cornelis C (2008)
A new approach to fuzzy-rough nearest neighbour classification. RSCTC 2008: Proceedings of the 6th International Conference on Rough Sets and Current Trends in Computing. Lecture Notes in Artificial Intelligence 5306. Springer, pp 310–319.

———, Cornelis C, Shen Q (2009)
Hybrid fuzzy-rough rule induction and feature selection. Proceedings of the 2009 IEEE International Conference on Fuzzy Systems. IEEE, pp 1151–1156.

———, Shen Q (2009)

Interval-valued fuzzy-rough feature selection in datasets with missing values. FUZZ-IEEE 2009: Proceedings of the 18th IEEE International Conference on Fuzzy Systems. IEEE, pp 610–615.

——— (2010)

Fuzzy-rough data mining with Weka. URL: `http://users.aber.ac.uk/rkj/Weka.pdf`.

———, Cornelis C (2011)

Fuzzy-rough nearest neighbour classification and prediction. Theoretical Computer Science, vol 412, no 42, pp 5871–5884.

———, Mac Parthaláin N (2015)

Towards scalable fuzzy–rough feature selection. Information Sciences, vol 323, pp 1–15.

Jiang Y, He X, Lee MLT, Rosner B, Yan J (2020)

Wilcoxon rank-based tests for clustered data with R package clusrank. Journal of Statistical Software, vol 96, no 6, pp 1–26.

Jirina M, Jirina Jr M (2011)

Classifiers based on inverted distances. New fundamental technologies in data mining. Ed. by K Funatsu, K Hasegawa. Rijeka: InTech. Chap. 19, pp 369–387.

Johnson BA, Tateishi R, Xie Z (2012)

Using geographically weighted variables for image classification. Remote Sensing Letters, vol 3, no 6, pp 491–499.

———, Tateishi R, Hoan NT (2013)

A hybrid pansharpening approach and multiscale object-based image analysis for mapping diseased pine and oak trees. International Journal of Remote Sensing, vol 34, no 20, pp 6969–6982.

Johnson J, Douze M, Jégou H (2021)

Billion-scale similarity search with gpus. IEEE Transactions on Big Data, vol 7, no 3, pp 535–547.

Jones DR (2001)

A taxonomy of global optimization methods based on response surfaces. Journal of Global Optimization, vol 21, no 4, pp 345–383.

Jones MP (1996)

Indicator and stratification methods for missing explanatory variables in multiple linear regression. Journal of the American Statistical Association, vol 91, no 433, pp 222–230.

Josse J, Prost N, Scornet E, Varoquaux G (2020)

On the consistency of supervised learning with missing values. arXiv preprint 1902.06931.

Jossinet J (1996)

Variability of impedivity in normal and pathological breast tissue. Medical and Biological Engineering and Computing, vol 34, no 5, pp 346–350.

Jović A, Brkić K, Bogunović N (2014)

An overview of free software tools for general data mining. Proceedings of the 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO 2014). IEEE, pp 1112–1117.

Kang S, Cho S, Kang P (2015)

Multi-class classification via heterogeneous ensemble of one-class classifiers. Engineering Applications of Artificial Intelligence, vol 43, pp 35–43.

Kapelner A, Bleich J (2015)

Prediction with missing data via Bayesian additive regression trees. Canadian Journal of Statistics, vol 43, no 2, pp 224–239.

Karau H, Konwinski A, Wendell P, Zaharia M (2015)

Learning spark: lightning-fast big data analysis. O'Reilly Media.

Kassab A (2021)

A sequential multi-stage one-class classification model in network intrusion detection systems. MA thesis. American University of Beirut.

Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, Ye Q, Liu TY (2017)

LightGBM: a highly efficient gradient boosting decision tree. NIPS 2017: Proceedings of the Thirty-first Conference on Neural Information Processing Systems. Advances in neural information processing systems 30. NIPS Foundation, pp 3146–3154.

Kim JO, Curry J (1977)

The treatment of missing data in multivariate analysis. Sociological Methods & Research, vol 6, no 2, pp 215–240.

Kim M, Choi JK, Baek SK (2021)

Win-stay-lose-shift as a self-confirming equilibrium in the iterated prisoner's dilemma. Proceedings of the Royal Society B, vol 288, no 1953, 20211021.

King DE (2009)

Dlib-ml: a machine learning toolkit. Journal of Machine Learning Research, vol 10, no 60, pp 1755–1758.

——— (2017)

A Global Optimization Algorithm Worth Using. `http://blog.dlib.net/2017/12/a-global-optimization-algorithm-worth.html`. Last accessed 6 Jan 2021.

Kingma DP, Ba JL (2015)

Adam: a method for stochastic optimization. ICLR 2015: 3rd International Conference on Learning Representations.

Klikowski J, Woźniak M (2020)

Employing one-class svm classifier ensemble for imbalanced data stream classification. International conference on computational science. Springer, pp 117–127.

Knorr EM, Ng RT (1997)

A unified notion of outliers: properties and computation. KDD-97: Proceedings of the Third International Conference on Knowledge Discovery and Data Mining. AAAI, pp 219–222.

Koczkodaj WW, Li F, Wolny–Dominiak A (2018)

RatingScaleReduction package: stepwise rating scale item reduction without predictability loss. The R Journal, vol 10, no 1, pp 43–55.

Kohavi R (1996)

Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. KDD-96: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining. AAAI Press, pp 202–207.

Krawczyk B, Woźniak M (2012)

Combining diverse one-class classifiers. International Conference on Hybrid Artificial Intelligence Systems. Springer, pp 590–601.

——— (2013)

Combining one-class support vector machines for microarray classification. 2013 Federated Conference on Computer Science and Information Systems. IEEE, pp 83–89.

———, Filipczuk P (2014)

Cytological image analysis with firefly nuclei detection and hybrid one-class classification decomposition. Engineering Applications of Artificial Intelligence, vol 31, pp 126–135.

———, Jeleń Ł, Krzyżak A, Fevens T (2014a)

One-class classification decomposition for imbalanced classification of breast cancer malignancy data. International Conference on Artificial Intelligence and Soft Computing. Springer, pp 539–550.

———, Woźniak M (2014a)

Diversity measures for one-class classifier ensembles. Neurocomputing, vol 126, pp 36–44.

———, Woźniak M (2014b)

Hypertension type classification using hierarchical ensemble of one-class classifiers for imbalanced data. International Conference on ICT Innovations. Springer, pp 341–349.

———, Woźniak M, Cyganek B (2014b)

Clustering-based ensemble of one-class classifiers for hyperspectral image segmentation. International Conference on Hybrid Artificial Intelligence Systems. Springer, pp 678–688.

———, Woźniak M, Cyganek B (2014c)
Clustering-based ensembles for one-class classification. Information Sciences, vol 264, pp 182–195.

———, Woźniak M, Herrera F (2015)
On the usefulness of one-class classifier ensembles for decomposition of multi-class problems. Pattern Recognition, vol 48, no 12, pp 3969–3982.

———, Cyganek B (2017)
Selecting locally specialised classifiers for one-class classification ensembles. Pattern Analysis and Applications, vol 20, no 2, pp 427–439.

———, Galar M, Woźniak M, Bustince H, Herrera F (2018)
Dynamic ensemble selection for multi-class classification with one-class classifiers. Pattern Recognition, vol 83, pp 34–51.

Kryszkiewicz M (1998)
Rough set approach to incomplete information systems. Information Sciences, vol 112, no 1-4, pp 39–49.

Kurgan LA, Cios KJ, Tadeusiewicz R, Ogiela M, Goodenday LS (2001)
Knowledge discovery approach to automated cardiac SPECT diagnosis. Artificial Intelligence in Medicine, vol 23, no 2, pp 149–169.

Kushner HJ (1962)
A versatile stochastic model of a function of unknown and time varying form. Journal of Mathematical Analysis and Applications, vol 5, no 1, pp 150–167.

——— (1964)
A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. Journal of Basic Engineering, vol 86, no 1, pp 97–106.

Lantz B (2013)
Machine learning with R. Birmingham: Packt Publishing. Chap. 3. Lazy Learning — Classification Using Nearest Neighbors, pp 65–87.

Le Morvan M, Josse J, Scornet E, Varoquaux G (2021)
What's a good imputation to predict with missing values? NeurIPS 2021: Proceedings of the Thirty-fifth Conference on Neural Information Processing Systems. Advances in neural information processing systems 34. NIPS Foundation, pp 11530–11540.

Lee T, Richards JA (1984)
Piecewise linear classification using seniority logic committee methods, with application to remote sensing. Pattern Recognition, vol 17, no 4, pp 453–464.

Lessmeier C, Enge-Rosenblatt O, Bayer C, Zimmer D (2014)
Data acquisition and signal analysis from measured motor currents for defect detection in electromechanical drive systems. PHME '14: Proceedings of the Second European Conference of the Prognostics and Health Management Society.

Lincoff GH (1981)

The Audubon Society Field Guide to North American Mushrooms. New York: Alfred A Knopf.

Lindqvist P, Peetre J (2000)

p-arclength of the q-circle. Preprint 2000:21 LUNFMA-5014-2000. Lund University, Centre for Mathematical Sciences.

Liu FT, Ting KM, Zhou ZH (2008)

Isolation Forest. ICDM 2008: Proceedings of the Eighth IEEE International Conference on Data Mining. IEEE, pp 413–422.

Liu J, Song J, Miao Q, Cao Y (2013)

Fenoc: an ensemble one-class learning framework for malware detection. 2013 Ninth International Conference on Computational Intelligence and Security. IEEE, pp 523–527.

Lohweg V, Hoffmann JL, Dörksen H, Hildebrand R, Gillich E, Hofmann J, Schaede J (2013)

Banknote authentication with mobile devices. MWSF 2013: Proceedings of the Media Watermarking, Security, and Forensics Conference. Proceedings of SPIE 8665. SPIE, pp 47–60.

Lucas DD, Klein R, Tannahill J, Ivanova D, Brandon S, Domyancic D, Zhang Y (2013)

Failure analysis of parameter-induced simulation crashes in climate models. Geoscientific Model Development, vol 6, no 4, pp 1157–1171.

Luengo J, García S, Herrera F (2012a)

On the choice of the best imputation methods for missing values considering three groups of classification methods. Knowledge and Information Systems, vol 32, no 1, pp 77–108.

———, Sáez JA, Herrera F (2012b)

Missing data imputation for fuzzy rule-based classification systems. Soft Computing, vol 16, no 5, pp 863–881.

Lyon RJ, Stappers B, Cooper S, Brooke JM, Knowles JD (2016)

Fifty years of pulsar candidate selection: from simple filters to a new principled real-time classification approach. Monthly Notices of the Royal Astronomical Society, vol 459, no 1, pp 1104–1123.

MacDonald MG, Feil L, Quinn T, Rice D (2022)

Confirming the 3:2 resonance chain of K2-138. The Astronomical Journal, vol 163, no 4, 162, p 162.

Macleod JE, Luk A, Titterington DM (1987)

A re-examination of the distance-weighted k-nearest neighbor classification rule. IEEE Transactions on Systems, Man, and Cybernetics, vol 17, no 4, pp 689–696.

Mahalanobis PC (1936)
On the generalized distance in statistics. Proceedings of the National Institute of Sciences of India, vol 2, no 1, pp 49–55.

Maillo J, Luengo J, García S, Herrera F, Triguero I (2017a)
Exact fuzzy k-nearest neighbor classification for big datasets. FUZZ-IEEE 2017: Proceedings of the IEEE International Conference on Fuzzy Systems.

———, Ramírez S, Triguero I, Herrera F (2017b)
kNN-IS: an iterative spark-based design of the k-nearest neighbors classifier for big data. Knowledge-Based Systems, vol 117, pp 3–15.

Malherbe C, Vayatis N (2017)
Global optimization of Lipschitz functions. ICML 2017: Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research 70, pp 2314–2323.

Malkov YA, Ponomarenko A, Logvinov A, Krylov V (2014)
Approximate nearest neighbor algorithm based on navigable small world graphs. Information Systems, vol 45, pp 61–68.

———, Yashunin DA (2020)
Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 42, no 4, pp 824–836.

Mangasarian OL, Wolberg WH (1990)
Cancer diagnosis via linear programming. Technical report 958. University of Wisconsin – Madison, Department of Computer Sciences.

———, Street WN, Wolberg WH (1994)
Breast cancer diagnosis and prognosis via linear programming. Proceedings of the AAAI Spring Symposium on Artificial Intelligence in Medicine: Interpreting Clinical Data. Spring Symposium Series Technical Reports 94-01. AAAI, pp 83–86.

Mansouri K, Ringsted T, Ballabio D, Todeschini R, Consonni V (2013)
Quantitative structure–activity relationship models for ready biodegradability of chemicals. Journal of Chemical Information and Modeling, vol 53, no 4, pp 867–878.

Marchand A, Van Lente F, Galen RS (1983)
The assessment of laboratory tests in the diagnosis of acute appendicitis. American Journal of Clinical Pathology, vol 80, no 3, pp 369–374.

Marlin BM (2008)
Missing data problems in machine learning. Doctoral thesis. University of Toronto.

McCann M, Li Y, Maguire L, Johnston A (2008)
Causality challenge: benchmarking relevant signal components for effective monitor-

ing and process control. NIPS 2008: Proceedings of Workshop on Causality. Proceedings of Machine Learning Research 6. JMLR Workshop and Conference Proceedings, pp 277–288.

McLeish M, Cecile M (1990)

Enhancing medical expert systems with knowledge obtained from statistical data. Annals of Mathematics and Artificial Intelligence, vol 2, no 1–4, pp 261–276.

Michalski RS, Chilausky RL (1980)

Learning by being told and learning from examples: an experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis. International Journal of Policy Analysis and Information Systems, vol 4, no 2, pp 125–161.

Michie D, Spiegelhalter DJ, Taylor CC (1994), eds.

Machine learning, neural and statistical classification. Artificial intelligence. Hemel Hempstead: Ellis Horwood.

Minkowski H (1896)

Geometrie der Zahlen. I. Lieferung. Leipzig: B.G. Teubner.

Möbius AF (1827)

Der barycentrische Calcul: ein Hülfsmittel zur analytischen Behandlung der Geometrie. Leipzig: Verlag von Johann Ambrosius Barth.

Molter F, Thomas AW, Huettel SA, Heekeren HR, Mohr PN (2022)

Gaze-dependent evidence accumulation predicts multi-alternative risky choice behaviour. PLoS Computational Biology, vol 18, no 7, e1010283.

Muja M, Lowe DG (2014)

Scalable nearest neighbor algorithms for high dimensional data. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol 36, no 11, pp 2227–2240.

Muller RH (1944a)

Verification of short-range weather forecasts (a survey of the literature) i. Bulletin of the American Meteorological Society, vol 25, no 1, pp 18–27.

——— (1944b)

Verification of short-range weather forecasts (a survey of the literature) ii (continued). Bulletin of the American Meteorological Society, vol 25, no 2, pp 47–53.

——— (1944c)

Verification of short-range weather forecasts (a survey of the literature) iii (conclusion)(concluded from february bulletin). Bulletin of the American Meteorological Society, vol 25, no 3, pp 88–95.

Mygdalis V, Alexandros I, Tefas A, Pitas I (2015)

Large-scale classification by an approximate least squares one-class support vector machine ensemble. 2015 IEEE Trustcom/BigDataSE/ISPA. Vol. 2. IEEE, pp 6–10.

Nadkarni P (2016)
Clinical research computing: a practitioner's handbook. London: Academic Press. Chap. 10. Core Technologies: Data Mining and "Big Data", pp 187–204.

Narayanan SJ, Bhatt RB, Perumal B (2016)
Improving the accuracy of fuzzy decision tree by direct back propagation with adaptive learning rate and momentum factor for user localization. IMCIP 2016: Proceedings of the Twelfth International Multi-Conference on Information Processing. Procedia Computer Science 89. Elsevier, pp 506–513.

Neal RM (2006)
Some notes for the BIRS Workshop on Statistical Inference for High Energy Physics. URL: https://glizen.com/radfordneal/BIRS-hep/notes1.pdf.

Nelder JA, Mead R (1965)
A simplex method for function minimization. The Computer Journal, vol 7, no 4, pp 308–313.

Da Rocha Neto AR (2006)
SINPATCO — sistema inteligente para diagnóstico de patologias da coluna vertebral. Doctoral thesis. Universidade Federal do Ceará, Fortaleza.

Ng CG, Yusoff MSB (2011)
Missing values in data analysis: ignore or impute? Education in Medicine Journal, vol 3, no 1.

Nguyen G, Dlugolinsky S, Bobák M, Tran V, García ÁL, Heredia I, Malík P, Hluch L (2019)
Machine learning and deep learning frameworks and libraries for large-scale data mining: a survey. Artificial Intelligence Review, vol 52, no 1, pp 77–124.

Nolan AM, Wachsman ED, Mo Y (2021)
Computation-guided discovery of coating materials to stabilize the interface between lithium garnet solid electrolyte and high-energy cathodes for all-solid-state lithium batteries. Energy Storage Materials, vol 41, pp 571–580.

Omohundro SM (1989)
Five Balltree Construction Algorithms. Technical report TR-89-063. Berkeley, California: International Computer Science Institute.

Orme JG, Reis J (1991)
Multiple regression with missing data. Journal of Social Service Research, vol 15, no 1–2, pp 61–91.

Paleyes A, Pullin M, Mahsereci M, Lawrence N, González J (2019)
Emulation of physical processes with Emukit. NeurIPS 2019: Workshop on Machine Learning and the Physical Sciences. NeurIPS.

Paschke F, Bayer C, Bator M, Mönks U, Dicks A, Enge-Rosenblatt O, Lohweg V (2013)
Sensorlose zustandsüberwachung an synchronmotoren. Proceedings of the 23rd

Workshop on Computational Intelligence. Schriftenreihe des Instituts für angewandte Informatik / Automatisierungstechnik am Karlsruher Institut für Technologie 46. KIT Scientific Publishing, pp 211–226.

Patrício M, Pereira J, Crisóstomo J, Matafome P, Gomes M, Seiça R, Caramelo F (2018)

Using resistin, glucose, age and BMI to predict the presence of breast cancer. BMC Cancer, vol 18, no 29.

Pawlak Z (1981)

Rough sets. Report 431. ICS PAS.

——— (1982)

Rough sets. International Journal of Computer & Information Sciences, vol 11, no 5, pp 341–356.

Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay É (2011)

Scikit-learn: machine learning in Python. Journal of Machine Learning Research, vol 12, no 85, pp 2825–2830.

Pelillo M (2014)

Alhazen and the nearest neighbor rule. Pattern Recognition Letters, vol 38, pp 34–37.

Pereira Barata A, Takes FW, Herik HJ van den, Veenman CJ (2019)

Imputation methods outperform missing-indicator for data missing completely at random. ICDM 2019: Proceedings of the Workshops. IEEE, pp 407–414.

Perez-Lebel A, Varoquaux G, Le Morvan M, Josse J, Poline JB (2022)

Benchmarking missing-values approaches for predictive models on health databases. GigaScience, vol 11, no 1, giac013.

Pigott TD (2001)

A review of methods for missing data. Educational Research and Evaluation, vol 7, no 4, pp 353–383.

Powell MJD (2004)

The NEWUOA software for unconstrained optimization without derivatives. Technical report NA2004/08. University of Cambridge, Department of Applied Mathematics and Theoretical Physics.

——— (2009)

The BOBYQA algorithm for bound constrained optimization without derivatives. Technical report NA2009/06. University of Cambridge, Department of Applied Mathematics and Theoretical Physics.

Qian Y, Wang Q, Cheng H, Liang J, Dang C (2015)

Fuzzy-rough feature selection accelerator. Fuzzy Sets and Systems, vol 258, pp 61–78.

Quinlan JR (1986)
Induction of decision trees. Machine Learning, vol 1, no 1, pp 81–106.

———, Compton PJ, Horn KA, Lazarus L (1986)
Inductive knowledge acquisition: a case study. Proceedings of the Second Australian Conference on Applications of Expert Systems. Turing Institute Press, pp 157–173.

——— (1987)
Simplifying decision trees. International Journal of Man-Machine Studies, vol 27, no 3, pp 221–234.

——— (1989)
Unknown attribute values in induction. Proceedings of the Sixth International Workshop on Machine Learning. Morgan Kaufmann, pp 164–168.

Ramana BV, Babu MSP, Venkateswarlu NB (2012)
A critical comparative study of liver patients from USA and India: an exploratory analysis. International Journal of Computer Science Issues, vol 9, no 3, pp 506–516.

Ramentol E, Vluymans S, Verbiest N, Caballero Y, Bello R, Cornelis C, Herrera F (2015)
IFROWANN: imbalanced fuzzy-rough ordered weighted average nearest neighbor classification. IEEE Transactions on Fuzzy Systems, vol 23, no 5, pp 1622–1637.

Ribeiro RP, Pereira P, Gama J (2016)
Sequential anomalies: a study in the railway industry. Machine Learning, vol 105, no 1, pp 127–153.

Ridder D de, Tax DMJ, Duin RPW (1998)
An experimental comparison of one-class classification methods. ASCI'98: Proceedings of the Fourth Annual Conference of the Advanced School for Computing and Imaging. ASCI, pp 213–218.

Riza LS, Janusz A, Bergmeir C, Cornelis C, Herrera F, Ślzak D, Benítez JM (2014)
Implementing algorithms of rough set theory and fuzzy rough set theory in the R package "RoughSets". Information Sciences, vol 287, pp 68–89.

Rizk Y, Awad M (2012)
Syntactic genetic algorithm for a subjectivity analysis of sports articles. Proceedings of the 11th IEEE International Conference on Cybernetic Intelligent Systems. IEEE, pp 93–98.

Rodríguez-Ruiz J, Mata-Sánchez JI, Monroy R, Loyola-Gonzalez O, López-Cuevas A (2020)
A one-class classification approach for bot detection on Twitter. Computers & Security, vol 91, p 101715.

Roe BP, Yang HJ, Zhu J, Liu Y, Stancu I, McGregor G (2005)
Boosted decision trees as an alternative to artificial neural networks for particle

identification. Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment, vol 543, no 2–3, pp 577–584.

Rohra JG, Perumal B, Narayanan SJ, Thakur P, Bhatt RB (2016)

User localization in an indoor environment using fuzzy hybrid of particle swarm optimization & gravitational search algorithm with neural networks. SocPros 2016: Proceedings of the Sixth International Conference on Soft Computing for Problem Solving. Advances in Intelligent Systems and Computing 546. Springer, pp 286–295.

Rosenblatt F (1961)

Principles of neurodynamics — Perceptrons and the theory of brain mechanisms. Technical report VG-1196-G-8. Buffalo, New York: Cornell Aeronautical Laboratory.

Rosner B, Glynn RJ, Lee MLT (2006)

The Wilcoxon signed rank test for paired comparisons of clustered data. Biometrics, vol 62, no 1, pp 185–192.

Rossiev DA, Golovenkin SE, Shulman V, Matjushin G (1995)

Neural networks for forecasting of myocardial infarction complications. Proceedings of the Second International Symposium on Neuroinformatics and Neurocomputers. IEEE, pp 292–298.

Rossum G van, Boer J de (1991)

Interactively testing remote servers using the Python programming language. CWI Quarterly, vol 4, no 4, pp 283–303.

Rousseeuw PJ, Croux C (1993)

Alternatives to the median absolute deviation. Journal of the American Statistical Association, vol 88, no 424, pp 1273–1283.

Rubin DB (1976)

Inference and missing data. Biometrika, vol 63, no 3, pp 581–592.

Rubini LJ, Eswaran P (2015)

Generating comparative analysis of early stage prediction of chronic kidney disease. International Journal of Modern Engineering Research, vol 5, no 7, pp 49–55.

Ruspini EH (1969)

A new approach to clustering. Information and Control, vol 15, no 1, pp 22–32.

Saltzer JH (2020)

The origin of the "MIT license". IEEE Annals of the History of Computing, vol 42, no 4, pp 94–98.

Sangma JW, Rani Y, Pal V, Kumar N, Kushwaha R (in press)

FHC-NDS: fuzzy hierarchical clustering of multiple nominal data streams. IEEE Transactions on Fuzzy Systems, vol. DOI: 10.1109/TFUZZ.2022.3189083.

Santos MS, Abreu PH, García-Laencina PJ, Simão A, Carvalho A (2015)
A new cluster-based oversampling method for improving survival prediction of hepatocellular carcinoma patients. Journal of Biomedical Informatics, vol 58, pp 49–59.

Schafer JL (1997)
Analysis of Incomplete Multivariate Data. Monographs on Statistics and Applied Probability 72. London: Chapman & Hall.

———, Graham JW (2002)
Missing data: our view of the state of the art. Psychological Methods, vol 7, no 2, pp 147–177.

Schlimmer JC (1987)
Concept acquisition through representational adjustment. Doctoral thesis. Technical report 87-19. University of California, Irvine.

Schölkopf B, Platt JC, Shawe-Taylor J, Smola AJ, Williamson RC (1999)
Estimating the support of a high-dimensional distribution. Technical report MSR-TR-99-87. Redmond, Washington: Microsoft Research.

———, Platt JC, Shawe-Taylor J, Smola AJ, Williamson RC (2001)
Estimating the support of a high-dimensional distribution. Neural Computation, vol 13, no 7, pp 1443–1471.

Shelupsky D (1959)
A generalization of the trigonometric functions. The American Mathematical Monthly, vol 66, no 10, pp 879–884.

Sigillito VG, Wing SP, Hutton LV, Baker KB (1989)
Classification of radar returns from the ionosphere using neural networks. Johns Hopkins APL Technical Digest, vol 10, no 3, pp 262–266.

Sikora M, Wróbel Ł (2010)
Application of rule induction algorithms for analysis of data collected by seismic hazard monitoring systems in coal mines. Archives of Mining Sciences, vol 55, no 1, pp 91–114.

——— (2011)
Induction and pruning of classification rules for prediction of microseismic hazards in coal mines. Expert Systems with Applications, vol 38, no 6, pp 6748–6758.

Silva PFB (2013)
Development of a system for automatic plant species recognition. MA thesis. Universidade do Porto.

Śmieja M, Struski Ł, Tabor J, Zieliński B, Spurek P (2018)
Processing of missing data by neural networks. NeurIPS 2018: Proceedings of the Thirty-second Annual Conference on Neural Information Processing Systems. Advances in neural information processing systems 31. NIPS Foundation, pp 689–696.

———, Struski Ł, Tabor J, Marzec M (2019)
Generalized RBF kernel for incomplete data. Knowledge-Based Systems, vol 173, pp 150–162.

Sokolov A, Paull EO, Stuart JM (2016)
One-class detection of cell states in tumor subtypes. PSB 2016: Proceedings of the 21st Pacific Symposium on Biocomputing. World Scientific, pp 405–416.

Soltani Zarrin P, Röckendorf N, Wenger C (2020)
In-vitro classification of saliva samples of COPD patients and healthy controls using machine learning tools. IEEE Access, vol 8, pp 168053–168060.

Spendley W, Hext GR, Himsworth FR (1962)
Sequential application of simplex designs in optimisation and evolutionary operation. Technometrics, vol 4, no 4, pp 441–461.

Sperrin M, Martin GP, Sisk R, Peek N (2020)
Missing data should be handled differently for prediction than for description or causal explanation. Journal of Clinical Epidemiology, vol 125, pp 183–187.

Stemplinger S, Prévost S, Zemb T, Horinek D, Dufrêche JF (2021)
Theory of ternary fluids under centrifugal fields. The Journal of Physical Chemistry B, vol 125, no 43, pp 12054–12062.

Stephenson W, Frangella Z, Udell M, Broderick T (2021)
Can we globally optimize cross-validation loss? quasiconvexity in ridge regression. NeurIPS 2021: Proceedings of the Thirty-fifth Conference on Neural Information Processing Systems. Advances in Neural Information Processing Systems 34. NeurIPS, pp 24352–24364.

Stigler SM (1986)
The history of statistics: the measurement of uncertainty before 1900. Cambridge, Massachusetts: The Belknap Press of Harvard University Press. Chap. 1. Least Squares and the Combination of Observations, pp 46–47.

Street WN (1991)
Toward automated cancer diagnosis: an interactive system for cell feature extraction. Technical report 1052. University of Wisconsin – Madison, Department of Computer Sciences.

———, Wolberg WH, Mangasarian OL (1992)
Nuclear Feature Extraction for Breast Tumor Diagnosis. Technical report 1131. University of Wisconsin – Madison, Department of Computer Sciences.

——— (1994)
Cancer diagnosis and prognosis via linear-programming-based machine learning. Doctoral thesis. Technical Report 94-14. University of Wisconsin – Madison, Department of Computer Sciences, Mathematical Programming Group.

———, Mangasarian OL, Wolberg WH (1995)
An inductive learning approach to prognostic prediction. ML95: Proceedings

of the Twelfth International Conference on Machine Learning. Morgan Kaufmann, pp 522–530.

———, Mangasarian OL, Wolberg WH (1996)

Individual and collective prognostic prediction. Technical report 96-01. University of Wisconsin – Madison, Department of Computer Sciences, Mathematical Programming Group.

Stumpf SA (1978)

A note on handling missing data. Journal of Management, vol 4, no 1, pp 65–73.

Suits DB (1957)

Use of dummy variables in regression equations. Journal of the American Statistical Association, vol 52, no 280, pp 548–551.

Swersky L, Marques HO, Sander J, Campello RJGB, Zimek A (2016)

On the evaluation of outlier detection and one-class classification methods. DSAA 2016: Proceedings of the 3rd IEEE International Conference on Data Science and Advanced Analytics. IEEE, pp 1–10.

Tax DMJ, Duin RPW (1998)

Outlier detection using classifier instability. SSPR/SPR 1998: Proceedings of the Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition and Structural and Syntactic Pattern Recognition. Lecture Notes in Computer Science 1451. Springer, pp 593–601.

———, Duin RPW (1999a)

Data domain description using support vectors. ESANN 1999: Proceedings of the Seventh European Symposium on Artificial Neural Networks. D-Facto, pp 251–256.

———, Duin RPW (1999b)

Support vector domain description. Pattern Recognition Letters, vol 20, no 11–13, pp 1191–1199.

——— (2001)

One-class classification: concept learning in the absence of counter-examples. Doctoral thesis. Technische Universiteit Delft.

———, Duin RPW (2004)

Support vector data description. Machine Learning, vol 54, no 1, pp 45–66.

Tchebyshev PL (1854)

Théorie des mécanismes connus sous le nom de parallélogrammes. Mémoires présentés à l'Académie impériale des Sciences de St. Petersbourg par divers Savants et lus dans ses Assamblées, vol 7, pp 537–568.

——— (1859)

Sur les questions de minimima qui se rattachent à la representation approximative des fonctions. Mémoires de l'Académie impériale des Sciences de St. Petersbourg,

sixième Série, première Partie: Sciences mathématiques et physiques, vol 7, pp 199–291.

Thangavel K, Pethalakshmi A (2009)

Dimensionality reduction based on rough set theory: a review. Applied Soft Computing, vol 9, no 1, pp 1–12.

Tian ZP, Nie RX, Wang JQ, Long RY (2021)

Adaptive consensus-based model for heterogeneous large-scale group decision-making: detecting and managing noncooperative behaviors. IEEE Transactions on Fuzzy Systems, vol 29, no 8, pp 2209–2223.

Tipping ME (2001)

Sparse Bayesian learning and the relevance vector machine. Journal of Machine Learning Research, vol 1, pp 211–244.

Todhunter I (1873)

A history of the mathematical theories of attraction and the figure of the earth, from the time of Newton to that of Laplace. Vol. 1. London: Macmillan. Chap. 14. Boscovich and Stay, pp 331–332.

Tohmé M, Lengellé R (2011)

Maximum margin one class support vector machines for multiclass problems. Pattern Recognition Letters, vol 32, no 13, pp 1652–1658.

Tönsmann M, Ewald DT, Scharfer P, Schabel W (2021)

Surface tension of binary and ternary polymer solutions: experimental data of poly(vinyl acetate), poly(vinyl alcohol) and polyethylene glycol solutions and mixing rule evaluation over the entire concentration range. Surfaces and Interfaces, vol 26, no 101352.

Torczon VJ (1989)

Multidirectional search: a direct search algorithm for parallel machines. Doctoral thesis. Rice University.

Tresp V, Neuneier R, Ahmad S (1994)

Efficient methods for dealing with missing data in supervised learning. NIPS-94: Proceedings of the Eighth Annual Conference on Neural Information Processing Systems. Advances in neural information processing systems 7. MIT Press, pp 689–696.

Troyanskaya O, Cantor M, Sherlock G, Brown P, Hastie T, Tibshirani R, Botstein D, Altman RB (2001)

Missing value estimation methods for DNA microarrays. Bioinformatics, vol 17, no 6, pp 520–525.

Twala BE, Jones M, Hand DJ (2008)

Good methods for coping with missing data in decision trees. Pattern Recognition Letters, vol 29, no 7, pp 950–956.

Twala B (2009)

An empirical comparison of techniques for handling incomplete data using decision trees. Applied Artificial Intelligence, vol 23, no 5, pp 373–405.

Unwin A, Kleinman K (2021)

The iris data set: in search of the source of virginica. Significance, vol 18, no 6, pp 26–29.

Utgoff PE, Brodley CE (1991)

Linear machine decision trees. COINS Technical Report 91-10. Amherst, Massachusetts: University of Massachusetts, Department of Computer and Information Science.

Vamplew P, Adams A (1992)

Missing values in a backpropagation neural net. ACNN '92: Proceedings of the Third Australian Conference on Neural Networks. Sydney University Electrical Engineering, pp 64–66.

Verbiest N, Cornelis C, Jensen R (2012)

Fuzzy rough positive region based nearest neighbour classification. FUZZ-IEEE 2012: Proceedings of the IEEE International Conference on Fuzzy Systems.

———, Cornelis C, Herrera F (2013)

OWA-FRPS: a prototype selection method based on ordered weighted average fuzzy rough set theory. RSFDGrC 2013: Proceedings of the 14th International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing. Lecture Notes in Computer Science 8170. Springer, pp 180–190.

——— (2014)

Fuzzy rough and evolutionary approaches to instance selection. Doctoral thesis. Universiteit Gent.

Vigna S (2015)

A weighted correlation index for rankings with ties. WWW '15: Proceedings of the 24th international conference on World Wide Web, pp 1166–1176.

Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, van der Walt SJ, Brett M, Wilson J, Millman KJ, Mayorov N, Nelson ARJ, Jones E, Kern R, Larson E, Carey CJ, Polat İ, Feng Y, Moore EW, VanderPlas J, Laxalde D, Perktold J, Cimrman R, Henriksen I, Quintero EA, Harris CR, Archibald AM, Ribeiro AH, Pedregosa F, van Mulbregt P, SciPy 1.0 Contributors (2020)

SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, vol 17, no 3, pp 261–272.

Vluymans S, Asfoor H, Saeys Y, Cornelis C, Tolentino M, Teredesai A, De Cock M (2015a)

Distributed fuzzy rough prototype selection for big data regression. NAFIPS 2015: Proceedings of the Annual Meeting of the North American Fuzzy Information Processing Society.

———, D'eer L, Saeys Y, Cornelis C (2015b)

Applications of fuzzy rough set theory in machine learning: a survey. Fundamenta Informaticae, vol 142, no 1-4, pp 53–86.

———, Sánchez Tarragó D, Saeys Y, Cornelis C, Herrera F (2016)

Fuzzy rough classifiers for class imbalanced multi-instance data. Pattern Recognition, vol 53, pp 36–45.

——— (2018)

Dealing with imbalanced and weakly labelled data in machine learning using fuzzy and rough set methods. Doctoral thesis. Universiteit Gent.

———, Cornelis C, Herrera F, Saeys Y (2018a)

Multi-label classification using a fuzzy rough neighborhood consensus. Information Sciences, vol 433, pp 96–114.

———, Fernández A, Saeys Y, Cornelis C, Herrera F (2018b)

Dynamic affinity-based classification of multi-class imbalanced data with one-versus-one decomposition: a fuzzy rough set approach. Knowledge and Information Systems, vol 56, no 1, pp 55–84.

———, Mac Parthaláin N, Cornelis C, Saeys Y (2019)

Weight selection strategies for ordered weighted average based fuzzy rough sets. Information Sciences, vol 501, pp 155–171.

Wang F, Yu J, Liu Z, Kong M, Wu Y (2021)

Study on offshore seabed sediment classification based on particle size parameters using XGBoost algorithm. Computers & Geosciences, vol 149, no 104713.

Wang Z, Liu K, Li J, Zhu Y, Zhang Y (in press)

Various frameworks and libraries of machine learning and deep learning: a survey. Archives of Computational Methods in Engineering, vol. DOI: 10.1007/s11831-018-09312-w.

Weiss SM, Kulikowski CA (1991)

Computer systems that learn: classification and prediction methods from statistics, neural nets, machine learning, and expert systems. San Mateo, California: Morgan Kaufmann Publishers.

Wettschereck D (1994)

A study of distance-based machine learning algorithms. Doctoral thesis. Oregon State University.

Wilcoxon F (1945)

Individual comparisons by ranking methods. Biometrics Bulletin, vol 1, no 6, pp 80–83.

Wilk T, Wozniak M (2010)

Combination of one-class classifiers for multiclass problems by fuzzy logic. Neural Network World, vol 20, no 7, p 853.

———, Wozniak M (2012)

Soft computing methods applied to combination of one-class classifiers. Neurocomputing, vol 75, no 1, pp 185–193.

Wolberg WH, Mangasarian OL (1990)

Multisurface method of pattern separation for medical diagnosis applied to breast cytology. Proceedings of the National Academy of Sciences of the United States of America, vol 87, no 23, pp 9193–9196.

———, Street WN, Heisey DM, Mangasarian OL (1995)

Computerized breast cancer diagnosis and prognosis from fine-needle aspirates. Archives of Surgery, vol 130, no 5, pp 511–516.

Wright MH (1995)

Direct search methods: once scorned, now respectable. Numerical analysis 1995: Proceedings of the 16th Dundee Biennial Conference on Numerical Analysis. Pitman Research Notes in Mathematics Series 344. Longman, pp 191–208.

Xing HJ, Liu WT (2020)

Robust adaboost based ensemble of one-class support vector machines. Information Fusion, vol 55, pp 45–58.

Yager RR (1988)

On ordered weighted averaging aggregation operators in multicriteria decisionmaking. IEEE Transactions on Systems, Man, and Cybernetics, vol 18, no 1, pp 183–190.

Yeh CY, Lee ZY, Lee SJ (2009)

Boosting one-class support vector machines for multi-class classification. Applied Artificial Intelligence, vol 23, no 4, pp 297–315.

Yeh IC, Yang KJ, Ting TM (2009)

Knowledge discovery on RFM model using bernoulli sequence. Expert Systems with Applications, vol 36, no 3, pp 5866–5871.

Yöntem MK, Adem K, İlhan T, Kılıçarslan S (2019)

Divorce prediction using correlation based feature selection and artificial neural networks. Nevşehir Hacı Bektaş Veli Üniversitesi SBE Dergisi, vol 9, no 1, pp 259–273.

Yu CD, Huang J, Austin W, Xiao B, Biros G (2015)

Performance optimization for the k-nearest neighbors kernel on x86 architectures. SC15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. 7. Austin, TX.

Zadeh LA (1965)

Fuzzy sets. Information and Control, vol 8, no 3, pp 338–353.

Zavrel J (1997)

An empirical re-examination of weighted voting for k-NN. BENELEARN-97: Proceedings of the 7th Belgian-Dutch Conference on Machine Learning. Tilburg University, pp 139–145.

Zeiler MD (2012)
ADADELTA: An Adaptive Learning Rate Method. arXiv preprint 1212.5701.

Zeng A, Li T, Liu D, Zhang J, Chen H (2015)
A fuzzy rough set approach for incremental feature selection on hybrid information systems. Fuzzy Sets and Systems, vol 258, pp 39–60.

———, Li T, Hu J, Chen H, Luo C (2017)
Dynamical updating fuzzy rough approximations for hybrid data under the variation of attribute values. Information Sciences, vol 378, pp 363–388.

Zhang Y, Zhang B, Coenen F, Xiao J, Lu W (2014)
One-class kernel subspace ensemble for medical image classification. EURASIP Journal on Advances in Signal Processing, vol 2014, no 1, pp 1–13.

Zhao R, Mao K (2018)
Fuzzy bag-of-words model for document representation. IEEE Transactions on Fuzzy Systems, vol 26, no 2, pp 794–804.

Zhu J, Zou H, Rosset S, Hastie T (2009)
Multi-class AdaBoost. Statistics and Its Interface, vol 2, no 3, pp 349–360.