Outlier Selection and One-Class Classification

Jeroen H.M. Janssens

Outlier Selection and One-Class Classification

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan Tilburg University,
op gezag van de rector magnificus,
prof. dr. Ph. Eijlander,
in het openbaar te verdedigen ten overstaan van een
door het college voor promoties aangewezen commissie
in de aula van de Universiteit
op dinsdag 11 juni 2013 om 14:15 uur

door

Jeroen Henricus Marinus Janssens

doctorandus in de artificiële intelligentie,
geboren op 6 juni 1983 te Helmond, Nederland

Promotores:

Prof. dr. E.O. Postma

Prof. dr. H.J. van den Herik

Beoordelingscommissie:

Prof. dr. E.J. Krahmer

Prof. dr. H.X. Lin

Prof. dr. J-J.Ch. Meyer

Prof. dr. A. Plaat

Dr. D.W. Aha

The research reported in this thesis has been carried out as part of the Poseidon project under the responsibility of the Embedded Systems Institute, the Netherlands. This project is partially supported by Agentschap NL under the BSIK03021 program.

SIKS Dissertation Series No. 2013-18
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

TiCC Dissertation Series No. 27

ISBN: 978–90–820273–1–0

Printed by Wöhrmann Print Service, Zutphen
Published by Van Lankveld Uitgeverij, Maastricht

Typeset by the author using XeLaTeX. The figures in the thesis were created using Ti*k*Z and MATLAB®. Thesis statistics: 216 pages; 39,625 words; 51 figures; 13 tables.

Preface

This thesis is the product of excellent supervision, infinite support, and great perseverance (with even a few anomalies and outliers along the way). The research presented has been conducted as part of a bigger project called Poseidon. Thales Nederland BV is Poseidon's main industrial partner. They offered an interesting, real-world problem to be investigated: to improve maritime safety and security. The problem involved many research challenges that have been taken up by different university teams, including visualising complex data, levering heterogeneous knowledge sources, and ensuring fault-tolerant distributed systems.

In 2007, prof.dr. Eric Postma was asked by the Embedded Systems Institute to form a Maastricht University team (which later migrated to the Tilburg University) to take on the research challenge of detecting maritime anomalies. I still consider it a privilege that Eric offered me the opportunity to continue my academic career as a Ph.D. student within the Poseidon project. It was not a difficult choice to accept the position, considering that Eric has been already my supervisor for both my bachelor's thesis in 2006 at University College Maastricht and my master's thesis in 2007 at Maastricht University. Shortly thereafter, the team would be strengthened by the addition of two other members: dr. Ildiko Flesh as post-doc and Prof.dr. Jaap van den Herik as my second supervisor.

Although the research presented in the thesis is domain-agnostic, it was guided by the challenge of maritime anomaly detection. Many possible research directions were possible to take on this challenge. Eventually, I converged to focus on outlier selection and one-class classification—with emphasis on density-based algorithms. This formed the basis of Chapter 3. Later, while studying the t-SNE algorithm (a non-linear dimensionality reduction technique by Laurens van der Maaten and Geoffrey Hinton), I came up with Stochastic Outlier Selection algorithm, which is now Chapter 4. After many experiments in which multiple algorithms were compared on multiple data sets, I wished to know which algorithm would be applied best to a given dataset; this is the contents of Chapters 5 and 6.

Without the help of many organisations and people, this thesis would not have been possible. I would like to express my gratitude to the following organisations: Tilburg Center for Communication and Cognition (TiCC), Department of Communication and Information Sciences (DCI), Tilburg School of Humanities (TSH), Tilburg University (TiU), Embedded Systems Institute (ESI), Thales Nederland BV, Noldus Information

Technology, School for Information and Knowledge Systems (SIKS), Agentschap NL, Maastricht University (UM), Department of Knowledge Engineering (DKE), Eindhoven University of Technology (TU/e) Machine Learning Summer School (MLSS09), University of Cambridge (CU), Computational and Biological Learning lab (CBL), and New York University (NYU). The number of people whose support have been crucial for the completion of the thesis is so large, that they deserve special acknowledgements (see page 197).

Jeroen Janssens January 2013, New York, NY

Contents

List of figures

List of tables

List of abbreviations

AIS	Automatic Identification System
ARA	Average Relative AUC
AUC	Area Under the ROC Curve
ESI	Embedded Systems Institute
KDD	Knowledge Discovery in Databases
KDE	Kernel Density Estimation
KNNDD	K-Nearest Neighbour Data Description
LOCI	Local Correlation Integral
LOF	Local Outlier Factor
LSOD	Least Squares Outlier Detection
MAD	Maritime Anomaly Detection
MDEF	Multi-Granularity Deviation Factor
ML	Machine Learning
MSS	Maritime Safety and Security
NN	Nearest Neighbour
OCC	One-Class Classification
PCA	Principal Component Analysis
Presto	Poseidon Research Tool
PWDD	Parzen Window Data Description
RFE	Recursive Feature Elimination
SNG	Stochastic Neighbour Graph

SOS . . . . . Stochastic Outlier Selection

SVDD . . . . . Support Vector Data Description

SVM . . . . . Support Vector Machines

TiU . . . . . Tilburg University

UOS . . . . . Unsupervised Outlier Selection

Chapter 1
Introduction

Contents:

What is common in a terrorist attack, a forged painting, and a rotten apple? The answer is: all three are anomalies; they are real-world observations that deviate from what is considered to be normal. Detecting anomalies is of utmost importance because an undetected anomaly can be dangerous or expensive. A human domain expert may suffer from three cognitive limitations: fatigue, information overload, and emotional bias. The cognitive limitations will hamper the detection of anomalies. Outlier-selection algorithms are capable of automatically classifying data points as outliers in large amounts of data. In the thesis, we study whether a domain expert can effectively be supported by such algorithms. This chapter introduces the problem statement and the three accompanying research questions. Subsequently, we state our research methodology and provide an outline of the remainder of the thesis.

Outline:

1.1 Anomaly detection. **1.2** Outlier selection. **1.3** Problem statement. **1.4** Research questions. **1.5** Research methodology. **1.6** Structure of the thesis.

1.1 Anomaly detection

The world around us seems to be quite normal. Laws of physics dictate how an apple falls, rules teach people not to steal, and schedules tell us when the next train is due. So, usually we know what to expect. However, sometimes, unexpected things happen. For those occasions we use the term anomaly. In the thesis we define an anomaly as follows.

Definition 1.1 (Anomaly). *An anomaly is an observation or event that deviates qualitatively from what is considered to be normal, according to a domain expert.*

In awkward situations, it transpires that the world around us is full of anomalies. Yet, we have to live with them and we, as researchers and domain experts, are given the task to investigate them. That is one of the aims of this study.

In Subsection 1.1.1, we describe the Poseidon project in which our research has been performed. Subsection 1.1.2 discusses anomaly detection in the maritime domain, and Subsection 1.1.3 provides examples of anomaly detection in other domains.

1.1.1 The Poseidon project

In 2007, a multi-disciplinary research project called 'Poseidon' was initiated by the Embedded Systems Institute (ESI) with Thales Nederland B.V. as the carrying industrial partner. Tilburg University (TiU), one of Poseidon's academic partners, was involved in the main research activity, which was to improve the safety and security in the maritime domain.[1] In particular, TiU was responsible for developing computer programs or algorithms that automatically detect maritime anomalies. This thesis describes the research that was conducted during the Poseidon project at TiU. Below we take a closer look at anomaly detection in the maritime domain.

1.1.2 Anomaly detection in the maritime domain

Maintaining safety and security of maritime traffic is of great importance for preventing or responding to accidents, terrorist attacks, or piracy. Human operators (i.e., domain

[1] Here we note that safety and security are different concepts. Safety refers to protection against accidental events. In addition, security refers to protection against intentional damages. In our research both concepts are involved. The nature of our research does not necessitate that we deal with both concepts separately. Therefore we speak of maintaining safety and security of maritime traffic.

experts) monitoring maritime safety and security typically watch a large visual display on which all vessel movements in a coastal region are plotted. With the help of the visual display, unexpected deviations from normality, i.e., anomalies, should be detected by the human operator. Despite the visual aid, anomalies often remain undetected. Below we mention three cognitive limitations that may underlie the shortcoming of human anomaly detection.

- Fatigue: human operators are bad at maintaining vigilance for a sustained period of time (Davies and Parasuraman, 1982).
- Information overload: the amount of information displayed is too much for a single human operator to process.
- Emotional bias: human operators suffer from an emotional bias, e.g., towards an outcome that is in their best interest.

We note that computers do not suffer from these three limitations. In fact, maintaining vigilance and processing large volumes of data are the hallmarks of computers. However, a computer may be biased from the start (e.g., it is wrongly designed), but so far it certainly has no emotional bias. Of course, in comparison with human operators, computers fall short in *understanding* maritime situations. The situation awareness of experienced operators relies largely on knowledge and familiarity with vessels, sea lanes, rules and regulations, the weather, and so forth. Obviously, for the time being, computers have no common sense or expert knowledge in the form of intuition that they can use.

An important lesson from the early days of artificial intelligence is that such common sense of expert knowledge is rather difficult to program into a computer algorithm, if it is possible at all (Russel and Norvig, 2010). Simply specifying all maritime knowledge in terms of rules leads to a system that is—again we admit—too rigid and that cannot deal with the probabilistic variations in the real world. Therefore, the best way to proceed is (1) to let the computer algorithm take care of the tasks requiring (1a) vigilance and (1b) cognitive processing power and (2) to leave the interpretation of the maritime situation largely to the expert.

1.1.3 Anomaly detection in other domains

Although the Poseidon project is targeted towards the maritime domain, it is not the sole application domain in which anomalies occur. To underline the general importance

of automatic anomaly detection, we mention three studies that apply anomaly detection algorithms to different problems:

1. detecting seizures in humans (Gardner, Krieger, Vachtsevanos, and Litt, 2006),
2. detecting credit card fraud (Whitrow, Hand, Juszczak, Weston, and Adams, 2009), and
3. change detection in satellite images (Robin, Moisan, and Le Hégarat-Mascle, 2010).

1.2 Outlier selection

The challenging task that we are facing in this study is: how can an algorithm support the expert in the detection of anomalies? We conjecture that the qualities of men and machines are complementary. Currently, a computer algorithm cannot work with the real-world observations directly. Therefore, the real-world observations need to be recorded and represented as data. In general, we may say that the expert's goal is to label real-world observations. For this purpose, we assume two possible labels, 'anomaly' and 'normality'.

At the computer side we see the following. The algorithm's goal is to classify the corresponding data points. We assume two possible classifications, namely 'outlier' and 'inlier'. An exact definition of an outlier depends on the implementation of the outlier-selection algorithm at hand. Below we provide a general definition of an outlier.

Definition 1.2 (Outlier)**.** *An outlier is a data point that deviates quantitatively from the majority of the data points, according to an outlier-selection algorithm.*

An inlier is a data point that is not an outlier. So, we distinguish between the labels of the expert and the classifications of the algorithm. As a direct consequence of this distinction we note that they may not always agree with each other.

The labels given by the expert are regarded as ground truth.[2] In other words, the expert is always right. The challenge is to let the algorithm classify the data points such that the agreement is maximised. The expert is the only one who may label the real-world observations as an anomaly or as a normality. The algorithm has no expert knowledge

[2] For sake of simplicity, we assume labels created by one expert only. Establishing ground truth from multiple, possibly disagreeing, experts is an interesting research problem on its own. In human computation, or crowdsourcing, for example, the challenge is to combine the labels provided by hundreds or thousands of people (see Janssens, 2010; Welinder, Branson, Belongie, and Perona, 2010).

and no common sense, and so has no notion of what is considered normal. The algorithm has only at its disposition the data points that represent the real world observations. The data points are a numerical representation of the real-world observations.

Figure 1.1 shows such a representation. For a certain number of vessels, two features are recorded or measured. The two features are 'speed over ground' and 'rate of turn'. The corresponding two-dimensional feature space is shown in Figure 1.1. The x-axis corresponds to the feature speed over ground, and the y-axis corresponds to the feature rate of turn.

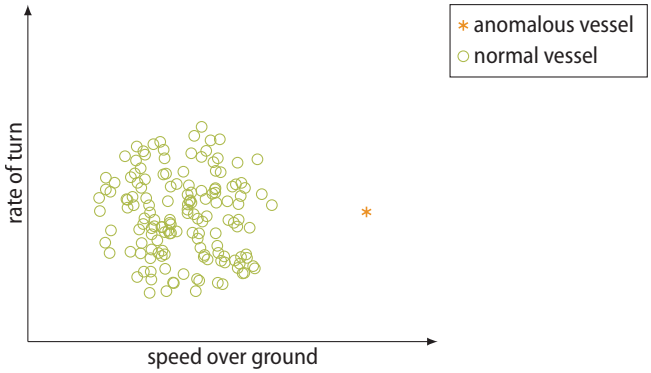


Figure 1.1 Example of a two-dimensional feature space where the data points represent real-world observations.

Let us consider that we observe a number of vessels sailing the sea. Of each vessel, we record and measure (1) the speed over ground and (2) the rate of turn. The resulting data set is plotted in Figure 1.1.

The expert labels one vessel as anomalous, because it is sailing too fast. The remaining vessels are labelled as normalities. The anomalous vessel is denoted by an orange asterisk and the normal vessels are denoted by green circles. Since speed over ground is a feature that is measured, the anomalous vessel lies outside the cluster of normal vessels. Because of the clear separation between the anomaly and the normalities, we may expect that an outlier-selection algorithm performs its task perfectly, i.e, classifying the data point that corresponds to the anomalous vessel as an outlier, and the remaining data points as inliers.

If the speed over ground was not measured and our feature representation would only comprise of the feature rate of turn, then the anomalous vessel would lie inside the cluster

of normal vessels. As a consequence, the algorithm would classify the corresponding data point as an inlier, which would result in a disagreement.

On the one hand, data are often cheap, especially when they come from sensors that automatically record real-world observations. On the other hand, labels are often expensive, especially when they are created by experts. In practical situations, we cannot assume that the expert labels every real-world observation that is recorded.

Often, the outlier-selection algorithm must be able to classify using the data points without any labels. When the algorithm classifies *unlabelled* data points, it is called *unsupervised*[3] and falls within the Unsupervised Outlier Selection (UOS) setting.

Algorithms in the One-Class Classification (OCC) setting are called one-class classifiers. A one-class classifier is *semi-supervised* because the labels of the normal data points are provided by the expert. Because anomalies are rare; we do not assume that any data points belong to the anomalous class.[4] They learn a model from only one class, namely the normal class (Tax, 2001); hence its name. Both settings are explained in Chapter 2.

1.3 Problem statement

The importance of automating the detection of real-world anomalies is clear from the above. So we will investigate whether outlier-selection algorithms are able to support the domain expert, such that the three cognitive limitations of the domain expert are mitigated. For this purpose, we formulate the following problem statement.

> **Problem statement:** *To what extent can outlier-selection algorithms support domain experts with real-world anomaly detection?*

The first question that comes to mind is: *when* can we state that a human expert is being supported? In the thesis, we consider a domain expert to be supported by outlier-selection algorithms when the following three conditions are satisfied.

---

[3] Please note that the terms 'unsupervised' and 'semi-supervised' refer to the *input* of the algorithm, and not to the *output*. It is, however, often the case that unsupervised machine learning algorithms do not classify the unlabelled data points. (For example, unsupervised clustering algorithms output clusters and unsupervised dimensionality reduction algorithms output lower-dimensional representations of the unlabelled data set.)

[4] We note that if the data set contains sufficient data points from *both* the normal class and the anomalous class, then the domain expert may be better supported by a supervised binary classifier, which is beyond the scope of the thesis.

1. A domain expert knows how to evaluate and compare the performance of the algorithms.
2. A domain expert has one or more effective algorithms available.
3. A domain expert knows when to apply which algorithm.

In the next section we transform each condition into a research question.

1.4 Research questions

In order to answer the problem statement, we investigate three research questions (RQs). The thee research questions correspond to the three conditions mentioned in Section 1.3. Below, we list the three research questions and provide a motivation for each of them.

Outlier-selection algorithms have been well studied in the research fields of Machine Learning (ML) and Knowledge Discovery in Databases (KDD) (Chandola, Banerjee, and Kumar, 2009). Both fields have produced their own algorithms and corresponding evaluation procedures. Here we note that not all algorithms within each field have been compared with each other under the same circumstances. Our first condition for a domain expert to be supported by outlier-selection algorithms stated that a domain expert should know how the performance of outlier-selection algorithms should be evaluated and compared. Therefore, the first research question reads as follows.

> **Research question 1:** *How should we evaluate and compare the performance of outlier-selection algorithms?*

The performance of an outlier-selection algorithm is mainly determined by the characteristics of the real-word data set at hand. Data characteristics such as cluster overlap may degrade the performance of an outlier-selection algorithm.

As mentioned in Subsection 1.1, each outlier-selection algorithm quantifies, in one way or another, the relationship between all the data points in the data set. For the research domains of clustering and dimensionality reduction, quantifying the relationship among data points plays an important role. Recently, several algorithms within those two domains employed effectively the concept of 'affinity' to fulfil that role (Hinton and Roweis, 2003; Frey and Dueck, 2007; van der Maaten, 2009b). In our study we aim to investigate (1) whether the concept of affinity is also effective for classifying data points as outliers and (2) whether the performance would be less degraded by the two example

data characteristics mentioned above. Hence, our second research question reads as follows.

> **Research question 2:** *Can an effective outlier-selection algorithm be devised that employs the concept of affinity?*

Because each real-world application is different, each real-world data set has different characteristics. For example, in one data set, the data points may form neat clusters, whereas in another data set, there may be hardly any structure. As a result of these varying data characteristics, no single outlier-selection algorithm outperforms all others on all data sets. It would be beneficial for a domain expert to know when to apply which algorithm. Therefore, we aim to understand the relationship between data characteristics and algorithm performance. Rice (1976) formalised this as the algorithm selection problem. In order to address the algorithm selection problem, i.e, in order to support the domain expert, we adopt a meta-learning approach (Smith-Miles, 2008). Our third research question reads as follows.

> **Research question 3:** *To what extent can we solve the one-class classifier selection problem using a meta-learning approach?*

The answers to these three research questions enable us to formulate an answer to the problem statement.

Contributions

Answering the research questions resulted in three main contributions.

- A framework for evaluating both semi-supervised and unsupervised outlier-selection algorithms.
- A novel outlier-selection algorithm called 'Stochastic Outlier Selection' that has a significantly higher performance and is more robust (to data perturbations and varying densities) than four related algorithms.
- As far as we know, the first meta-learning study for one-class classification where we aim to model the performance of 19 one-class classifiers on 255 one-class data sets using 36 meta-features.

Two additional contributions are (1) Presto, an application to create artificial vessel trajectories and (2) the organisation the first international workshop on Maritime Anomaly

Detection (MAD). Presto and the MAD workshop are described in Appendices A and B, respectively.

1.5 Research methodology

Within our research we employ an inductive research methodology, which consists of five stages: (1) review and analyse the scientific literature, (2) design and develop a novel outlier-selection algorithm, (3) implement the outlier-selection algorithm in MATLAB®, (4) perform comparative experiments, and (5) analyse the obtained results. Table 1.1 shows which stages are employed to address each of the three research questions. Below we discuss the five stages.

Table 1.1 Overview of the research methodology and its five stages employed to address the three research questions.

Methodology stages	RQ1	RQ2	RQ3
Review and analyse scientific literature	✓	✓	✓
Design and develop a novel outlier-selection algorithm		✓	
Implement outlier-selection algorithm in MATLAB®	✓	✓	
Perform comparative experiments	✓	✓	✓
Analyse obtained results	✓	✓	✓

Review and analyse scientific literature

We review and analyse the scientific literature for three reasons. First, to identify the current state-of-the-art in outlier selection for RQ1. Second, to design and develop a novel and effective algorithm for RQ2. Third, to establish a well-founded experimental set-up for RQ3. Appropriate literature is found in, but not limited to, the research fields of (1) machine learning, (2) knowledge discovery and databases, (3) information theory, (4) probability theory, and (5) graph theory.

Design and develop a novel outlier-selection algorithm

Investigating RQ2 requires the development of an outlier-selection algorithm that employs the concept of affinity. The algorithm relies on concepts from (1) probability theory, (2) graph theory, and (3) information theory, of which the appropriate literature

has been studied. We examine the intricacies of the novel algorithm, such as the influence of its parameter(s) and its sensitivity to data perturbations.

Implement outlier-selection algorithm in MATLAB®

We implement the developed algorithm in a programming language since we wish to evaluate its performance. The widely-used and freely-available Data Description Toolbox by Tax (2012) provides implementations of a large number of outlier-selection algorithms in the programming language MATLAB®. In order to make use of this toolbox, we decided to implement our algorithm in the programming language MATLAB® as well. For a good comparison, adequate implementations are also required for outlier-selection algorithms that have been developed previously by other authors, of which to our regret no (appropriate) implementation are available. This holds for the outlier-selection algorithms Local Outlier Factor (LOF) and Local Correlation Integral (LOCI) (Breunig, Kriegel, Ng, and Sander, 2000; Papadimitriou, Kitagawa, Gibbons, and Faloutsos, 2003). Our MATLAB® implementations of LOF and LOCI are now part of the Data Description Toolbox.

Perform comparative experiments

Once all necessary implementations are present, we are able to perform comparative experiments. This is required for all research questions. In a typical machine learning experiment, the experimental set-up consists of evaluating, i.e, applying, one or more algorithms to one or more data sets. The same holds for outlier selection and one-class classification. To study the outlier-selection algorithms in greater detail, we may (1) vary the parameters of the algorithms, and (2) vary the characteristics of the data sets. The experimental set-up involves three evaluation methods. First, cross-validation ensures that the performance generalises to an independent test data set. Second, the Area Under the ROC Curve is used as performance measure. Third, the post-hoc Neményi statistical test is employed to test for statistical significance among the performances of the algorithms. These three techniques will be further explained in Chapter 2.

Analyse obtained results

After a comparative experiment has been performed, we continue with the analysis of the obtained results. Typically, in the machine learning literature, the performances are

averaged and the algorithm with the highest average performance is presented as the best. However, the 'No Free Lunch' theorem implies that for each algorithm there exists a data set on which it is outperformed (Wolpert and Macready, 1995). Therefore, we also investigate under which conditions, i.e., data characteristics, an algorithm performs best. As mentioned in the previous section, we aim to understand the relationship between data characteristics and algorithm performance.

1.6 Structure of the thesis

The problem statement and the three accompanying research questions introduced in Sections 1.3 and 1.4 are investigated over the course of seven chapters. Table 1.2 lists which chapters address the problem statement and research questions. Below we provide a brief description of the contents of each chapter.

Table 1.2 The problem statement (PS) and the three accompanying research questions (RQs) are addressed in different chapters throughout the thesis.

Chapter	PS	RQ1	RQ2	RQ3
Chapter 1: Introduction	✓	✓	✓	✓
Chapter 2: Background and experimental set-up		✓	✓	✓
Chapter 3: Evaluating and comparing outlier-selection algorithms		✓		
Chapter 4: Stochastic Outlier Selection			✓	
Chapter 5: Meta-features for one-class data sets				✓
Chapter 6: Meta-learning for one-class classifiers				✓
Chapter 7: Conclusions	✓	✓	✓	✓

Chapter 1: Introduction

In Chapter 1, we define the concepts of an anomaly and an outlier. We illustrate the importance of anomaly detection by listing examples from the maritime domain and three other application domains. Three cognitive limitations of anomaly detection by domain experts are given. We mention recent results that indicate that outlier-selection algorithms can effectively support the domain expert. Based on these results we formulate our problem statement and three accompanying research questions. Moreover, we describe the employed research methodology. The chapter is concluded by an overview of the thesis.

Chapter 2: Background and experimental set-up

In Chapter 2, we introduce the main concepts and explain the experimental set-up that is employed throughout the subsequent chapters. This is partly based on a review and the analysis of the relevant literature concerning anomaly detection, outlier selection, and machine learning. The chapter consists of four parts. First, we present two outlier-selection settings that we consider in the thesis: (1) unsupervised outlier selection and (2) one-class classification. We highlight their similarities and differences, and we make clear when which setting is appropriate. Second, we describe two types of data representations that outlier-selection algorithms can process: (1) feature-vector representation and (2) similarity-matrix representation. Both representations have their uses. Third, we explain in detail the experimental set-up. It relies on statistical techniques such as the (1) Area Under the ROC Curve (AUC) performance measure, (2) the post-hoc Neményi statistical test, and (3) cross validation. Fourth, we explain how to transform ordinary multi-class data sets into one-class data sets so that they can be used for our comparative experiments.

Chapter 3: Evaluating and comparing outlier-selection algorithms

In Chapter 3, we aim to answer RQ1. We describe how anomalies are detected in the fields of ML and KDD. Both fields have their own anomaly-detection algorithms and corresponding evaluation procedures. Two well-known outlier-selection algorithms from the field of KDD are framed into the one-class classification framework so that these can be compared with three algorithms from the field of ML in a statistically valid way. We perform a comparative evaluation of the ML and KDD outlier-selection algorithms on real-world datasets and discuss the results.

Chapter 4: Stochastic Outlier Selection

In Chapter 4, we aim to answer RQ2. We present a novel, unsupervised algorithm for classifying outliers, called Stochastic Outlier Selection (SOS). SOS uses affinity to compute for each data point an outlier probability. The probabilities that are computed by SOS provide several advantages with respect to the unbounded outlier scores as computed by related algorithms. Using outlier-score plots, we illustrate and discuss the qualitative performances of SOS and four related algorithms. Then we evaluate SOS and the four algorithms on eighteen real-world data sets and seven synthetic data sets. The

results obtained on these 25 data sets show that SOS (1) has a significantly higher average performance and (2) is more robust to data perturbations and varying densities than the four related algorithms. From these results we may conclude that SOS is an effective algorithm for classifying data points as outliers. We observe that SOS is, however, not the best-performing algorithm on every data set. This observation is consistent with the No Free Lunch theorem, which is discussed in Chapter 5.

Chapter 5: Meta-features for one-class data sets

The research required to formulate an answer to RQ3 is split over the Chapters 5 and 6. In Chapter 5, we discuss the No Free Lunch theorem, which states that there is no single best one-class classifier. For each one-class data set, there may be a different one-class classifier that performs best. The performance of a one-class classifier is greatly determined by the characteristics, or meta-features, of the one-class data set. We present three methods for preprocessing data that may improve the performance of certain one-class classifiers on certain one-class data sets. The results in this chapter are obtained in three steps. First, we implement 36 meta-features. Second, we apply the meta-features to 255 data sets. Third, we select empirically the most informative meta-features. The selected meta-features are used as input for Chapter 6, where we use meta-learning to relate the meta-features to the one-class classifier performance.

Chapter 6: Meta-learning for one-class classifiers

In Chapter 6, we aim to answer RQ3 by using the output from Chapter 5. First, we describe 19 one-class classifiers. Subsequently, we perform a comparative experiment by applying the 19 one-class classifiers on the 255 one-class data sets from Chapter 5. To answer RQ3, we aim to understand the relationship between the previously computed meta-features and the one-class classifier performances. To this end, we set up a meta-learning experiment, where we aim to solve the one-class classifier selection problem, i.e., predict the most appropriate one-class classifier for a given, unseen, one-class data set. The results indicate that by using the meta-learning approach, we can solve the one-class classifier selection problem with an accuracy of 52%.

Chapter 7: Conclusions

In Chapter 7 we complete the thesis. We discuss the findings of the thesis on a general level. By reviewing the answers to the three research questions, we arrive at three conclusions. From the conclusions, we formulate an answer to the problem statement. Finally, we give recommendations for future research.

Chapter 2

Background and experimental set-up

Contents:

In this chapter we introduce the main concepts for answering the research questions. In the first part of the chapter (from Section 2.2 to Section 2.7) we assume that the outlier selection is unsupervised. The second part of the chapter (Section 2.8) deals with the one-class classification setting. For certain applications, the assessments of the domain expert are available to the algorithm. If that is the case, then the one-class classification setting may be employed.

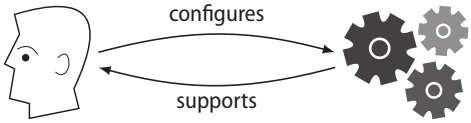
Outline:

2.1 The relationship between the domain expert and the outlier-selection algorithm. **2.2** Representing real-world observations as data points. **2.3** Outlier-selection algorithm. **2.4** Domain expert. **2.5** Hits, misses, correct rejects, and false alarms. **2.6** Evaluating the performance of an outlier-selection algorithm. **2.7** Method for comparing outlier-selection algorithms. **2.8** One-class classification setting. **2.9** Chapter summary.

2.1 The relationship between the domain expert and the outlier-selection algorithm

The domain expert and the outlier-selection algorithm form a symbiotic relationship; they rely on each other in two ways (see Table 2.1). First, the domain expert configures the parameters of the outlier-selection algorithm. Second, the outlier-selection algorithm *should* support the domain expert in order to mitigate his[1] cognitive limitations. We remark that the word 'should' is emphasised because this is essentially the problem statement to which we aim to formulate an answer.[2]

Table 2.1 The symbiotic relationship and parallels between the domain expert and the outlier-selection algorithm.

The	Domain expert	Outlier-selection algorithm
		
employs	Knowledge and experience	Mathematics and statistics
on the	Real world	Data set
to	Detect	Select
those	Observations	Data points
that are	Abnormal	Outlying
as	Anomalies	Outliers

From Table 2.1 we can read the following two sentences. First, the domain expert employs domain knowledge and experience on the real world to detect those observations that are abnormal as anomalies. Second, the outlier-selection algorithm employs techniques from mathematics and statistics on the data set to classify those data points that are outlying as outliers. The words have been put side-by-side in the table in order to emphasise the parallels that exist. The two most important parallels are (i) between (real-world) observations and data points and (ii) between anomalies and outliers.

To illustrate the relationship between the domain expert and the outlier-selection algorithm in more detail, we employ a data flow diagram. The data flow diagram in Figure 2.1 shows the flow of data, i.e., the relationship, between (1) the domain expert and

[1] For brevity, we use 'he' and 'his' whenever 'he or she' and 'his or her' are meant.

[2] We may ask ourselves the converse: *To what extent can the domain expert configure the algorithm?* However, this is beyond the scope of the thesis.

(2) the outlier-selection algorithm, and the three accompanying concepts: (3) the real-world, (4) the data representation, and (5) the data set. As such, the data flow diagram may serve as an overview reference while the five concepts are discussed in the remainder of the chapter.

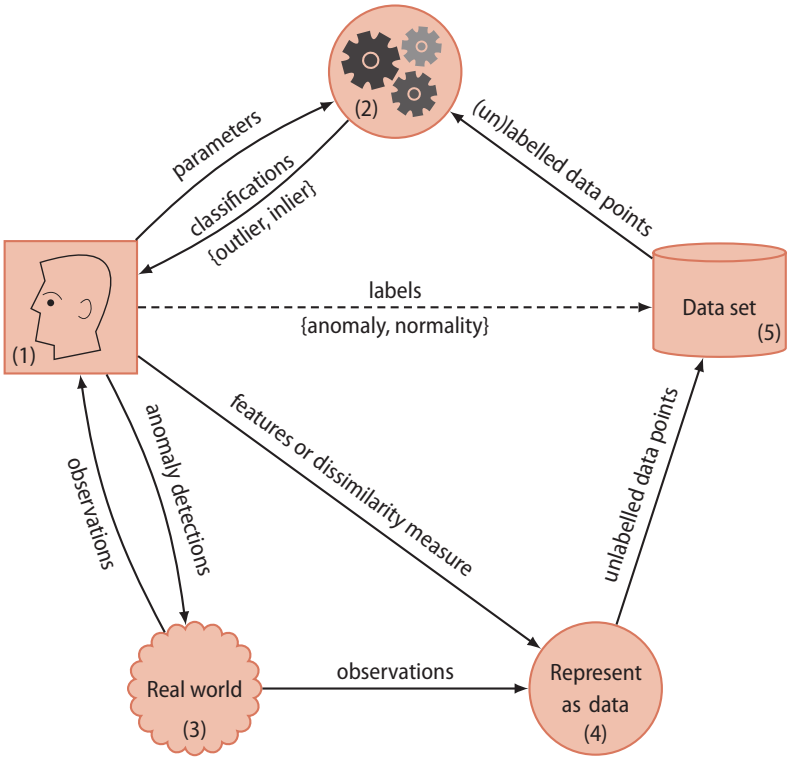


Figure 2.1 Data flow diagram illustrating the relationship between the domain expert (square) and the outlier-selection algorithm (top circle).

We highlight the main flow of data, which starts at the bottom-left corner and proceeds counter-clockwise: *observations* from the *real-world* are *represented as data*, which are subsequently stored in a *data set*. The *outlier-selection algorithm* processes the data set and may classify certain *data points* as *outliers*. The outliers are in turn evaluated by the *domain expert* who may then decide whether the real-world observation should be considered as an *anomaly* and consequently act upon it. The main flow of data and the other supporting flows of data are discussed in the subsequent sections.

In data-flow-diagram terminology, a circle represents a function (i.e., (2) 'outlier-selection algorithm' and (4) 'represent as data'), a square represents an end-user (i.e., (1) 'domain expert') and a cylinder represents a data store (i.e, (5) 'data set'). The (3)

'real world' is represented by a cloud-shape (which is not common in data flow diagram terminology). The arrows represent the data transfer from one concept to another concept.

The remainder of the chapter is organised as follows. We should first note that, in Sections 2.2–2.7, we assume the unsupervised outlier-selection setting. In other words, we assume that the data set has not been labelled by the domain expert.

In Section 2.2 we explain how real-world observations can be represented as data points. In Section 2.3 we give a definition of an outlier-selection algorithm. Section 2.4 discusses the characteristics of the domain expert. Section 2.5 describes four types of (dis)agreements between the domain expert and the outlier-selection algorithm. In Section 2.6 we explain how we evaluate an outlier-selection algorithm. In Section 2.7 we describe how multiple outlier-selections may be compared with each other.

In Section 2.8 we change the setting from unsupervised to semi-supervised. In other words, now we assume that the domain expert has labelled (part of) the data set. We describe the one-class classification setting and introduce the concept of cross-validation, which is needed for the evaluation of one-class classifiers. Finally, we provide a summary of the chapter in Section 2.9.

2.2 Representing real-world observations as data points

From Figure 2.1 we see that the outlier-selection algorithm does not operate on real-world observations directly. The real-world observations need to be represented as data points first. The six Euler diagrams in Figure 2.2 show the relationships between different sets (i.e., data concepts). Below, we discuss the first four Euler diagrams, namely (a) to (d). The discussion of diagrams (e) and (f) is deferred to Sections 2.3 and 2.5, respectively.

The set of all real-world observations of a certain application domain (e.g., vessels from the maritime domain) is denoted by $\mathcal{X}$ (see Figure 2.2(a)). An observation is a generic term and may refer to, for example, events, physical objects, and digital records in a database. We assume that each real-world observation is labelled by a domain expert (e.g., the coastguard operator) as either normal or anomalous. (It includes all the observations, also the ones that the domain expert has not seen.) This is illustrated in Figure 2.2(b), where $\mathcal{A}$ indicates the set of anomalies, which is a subset of the real-world observations, i.e., $\mathcal{A} \subset \mathcal{X}$. Each real-world observation is either recorded as a data point or unrecorded

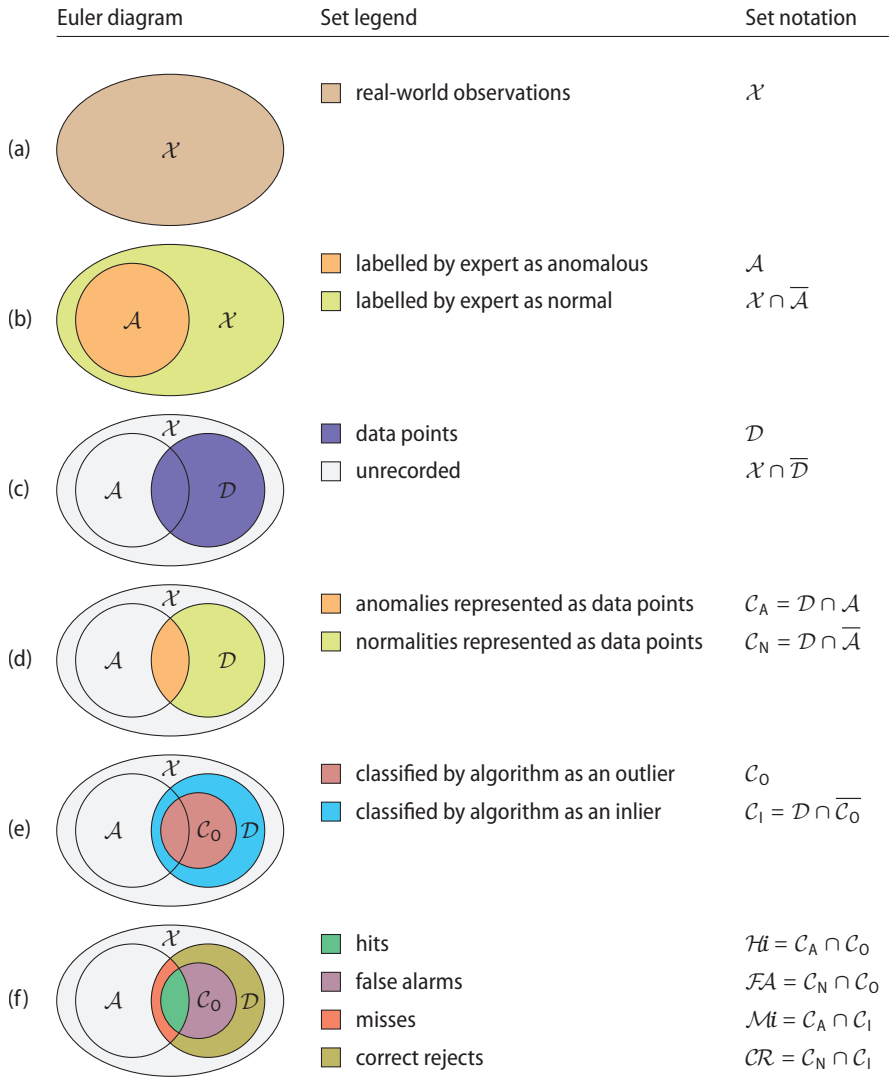


Figure 2.2 Six Euler diagrams illustrating the relationships between the following sets: real-world observations, anomalies, normalities, data set, outliers, inliers, hits, false alarms, misses, and correct rejects.

(see Figure 2.2(c)). Recording occurs when the real-world observation is, for example, caught on camera. All recorded data points are stored in a data set, which is denoted by $\mathcal{D}$. The unrecorded real-world observations cannot be processed by the outlier-selection algorithm. If we combine diagrams (b) and (c), we observe that a data set can contain both normal and anomalous data points (see Figure 2.2(d)).

There are several approaches to represent real-world observations as data (Gärtner, Lloyd, and Flach, 2004). We describe two of them that are employed most commonly: (1) the feature-vector representation (Subsection 2.2.1) and (2) the dissimilarity-matrix representation (Subsection 2.2.2). In Subsection 2.2.3 we explain the difference between the two representations. In Subsection 2.2.4 we describe how a dissimilarity matrix can be obtained from feature vectors.

2.2.1 Feature-vector representation

The first approach to represent real-world observations as data points is through feature vectors (cf. Witten and Frank, 2005). The number of data points in a data set $\mathcal{D}$ is denoted by n. In the thesis we assume that all values are numerical, as opposed to categorical, symbolic, or binary. In a feature-vector representation each data point is represented by an m-dimensional, real-valued, feature vector $\mathbf{x} = [x_1, \ldots, x_m] \in \mathbb{R}^m$. As such, a data point can be regarded as a point in an m-dimensional Euclidean space. The data set $\mathcal{D}$ is represented by a matrix $\mathbf{X}$ of size $n \times m$, i.e., number of data points × number of features. The vector $\mathbf{x}_i$ denotes the i^{th} row of the matrix $\mathbf{X}$.

When the observations are physically embedded in the real world, one or more of its features must be measured in order to obtain a feature vector. Figure 2.3 illustrates an example of the feature-vector representation concerning apples and oranges. Of 42 apples and oranges, two features are measured: width and height. The table in the middle displays the first seven data points. The scatter plot on the right shows all 42 apples and oranges.

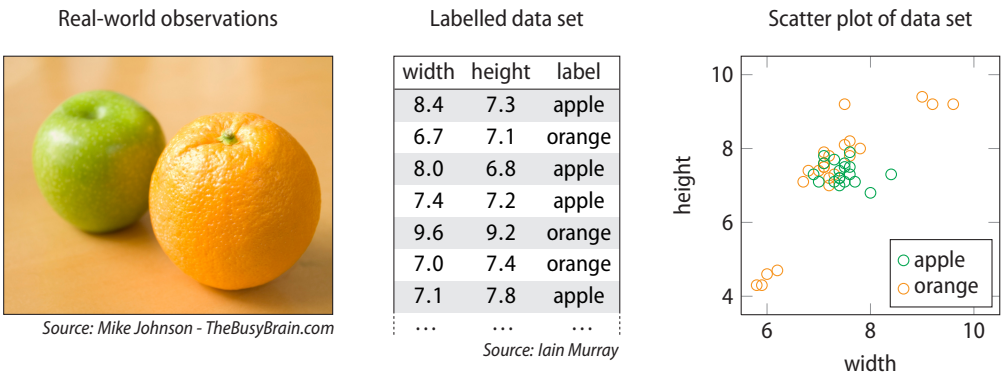


Figure 2.3 Illustration of the feature-vector representation. **Left:** Real-world observations of an apple and an orange. **Middle:** Apples and oranges represented as two-dimensional feature vectors together with a class-label. **Right:** The data set visualised using a scatter plot.

The data set is completely labelled, i.e., the domain expert has indicated for each observation whether it is an apple or an orange. The scatter plot indicates that the features width and height are insufficient to distinguish between apples and oranges. Such a sub-optimal data representation may be improved by (1) measuring more features such as weight, or (2) extracting features from digital images, if they are available (cf. van der Maaten, 2009a).

Here we remark that in application domains with a database, the observations are already represented as data, and may be stored in the correct format. When the records[3] in the database have been entered manually, data cleaning is necessary in cases where the data is blurred and any algorithm processing the data produces unreliable classifications (cf. van Erp, 2010). Each record in the database is considered as a feature vector in the data set.

2.2.2 Dissimilarity-matrix representation

The second approach to represent data is the dissimilarity-matrix representation, which is also known as a featureless representation (Pękalska and Duin, 2005). It is a relative description. Such a relative description of a real-world observation is obtained by measuring its dissimilarity to other real-world observations. The dissimilarity is expressed as a non-negative scalar and is computed by a dissimilarity measure d. If we have n real-world observations, then $n^2 - n$ dissimilarities need to be computed. (A real-world observation need not be compared to itself.) The result is a dissimilarity matrix with size $n \times n$, and 0's (zeros) on the diagonal. The dissimilarity-matrix representation has the advantage that any dissimilarity measure can be used as well as any outlier-selection algorithm that operates in vector spaces (Duin, Loog, Pękalska, and Tax, 2010).

2.2.3 The difference between the representations

Finally we emphasise the differences between the two representations. Because a feature vector has a fixed length m, complex real-world observations may be better represented using a dissimilarity matrix. For example, different moving object trajectories may have different durations and therefore a different number of measurements, i.e., different

[3] In the field of Knowledge Discovery in Databases (KDD), which focusses on the real-world application of machine learning algorithms, the terms 'data set', 'data point', and 'feature' are more commonly known as 'database', 'record', and 'field' (or 'attribute'), respectively (Fayyad, Piatetsky-Shapiro, Smyth, and Uthurusamy, 1996b).

values of m (cf. de Vries, 2012). Other types of complex real-world observations include text-documents and social-network graphs.

A feasible feature-vector representation of real-world observations might be difficult to obtain or insufficient for learning purposes, e.g., when domain experts cannot define features in a straightforward way, when data are high dimensional, or when features consist of both continuous and categorical variables (Pękalska and Duin, 2002).

The main difference between both representations is that in the feature-vector representation the observation is defined by itself, whereas in the dissimilarity-matrix representation, the observation is defined with respect to other observations. We reiterate that feature vectors are an absolute description of real-world observations, and that a dissimilarity matrix is a relative description of real-world observations.

2.2.4 Obtaining a dissimilarity matrix from feature vectors

Feature vectors $\mathbf{X}$ can be transformed into a dissimilarity matrix $\mathbf{D}$ (cf. Duin et al., 2010). Here we illustrate such a transformation by using the Euclidean distance as the dissimilarity measure on the feature vectors displayed in Figure 2.4. The Euclidean distance is defined as follows.

$$d_{ij} = \sqrt{\sum_{k=1}^{m} \left(x_{jk} - x_{ik} \right)^2}, \tag{2.1}$$

where x_{ik} denotes the k^{th} feature value of i^{th} data point, i.e., cell i, k of matrix $\mathbf{X}$. From Equation 2.1 it follows (1) that the dissimilarity measure is symmetric, i.e., $d_{ij} \equiv d_{ji}$, and (2) that the dissimilarity between a data point and itself is zero, i.e., $d_{ii} = 0$. The six data points on the left side of Figure 2.4 are connected by green lines with varying brightness. Both the length and the brightness of these green lines illustrate the dissimilarity between data point $\mathbf{x}_2$ and the other five data points.

The right side of Figure 2.4 shows the matrix $\mathbf{X}$ containing six data points characterised by two features. Next to matrix $\mathbf{X}$ we find a colour bar that maps a range of values to a range of colours. For example, 0 is mapped to white and 8 is mapped to dark blue. The dissimilarity matrix $\mathbf{D}$ is obtained by applying Equation 2.1 to each pair of data points in the matrix $\mathbf{X}$. (The bold, upright letter '$\mathbf{D}$' should not be confused with the calligraphic letter '$\mathcal{D}$' that denotes a data set.) The brightness of each green line is equal to the brightness of the corresponding cell in the second row of $\mathbf{D}$. In fact, because the

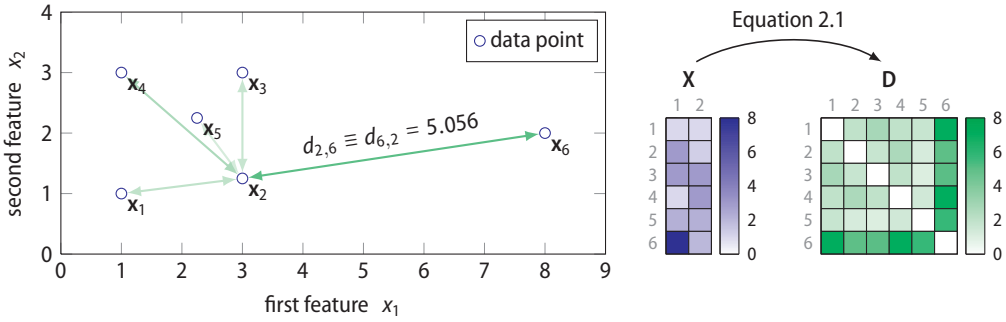


Figure 2.4 Transforming feature vectors into a dissimilarity matrix.

Euclidean distance d is symmetric, the resulting dissimilarity matrix $\mathbf{D}$ is symmetric, meaning that the rows are equal to the columns. Therefore, the brightness of the green lines are also equal to the brightness of the cells of the second column.

2.3 Outlier-selection algorithm

In this section we discuss the concept of an outlier-selection algorithm. Once the real-world observations have been represented as data points, the data points can be processed by an outlier-selection algorithm. In general, an outlier-selection algorithm takes as input a (un)labelled data set and classifies each data point either as an outlier or an inlier. This is illustrated in the Euler diagram in Figure 2.2(e). The classifications are transferred to the domain expert (discussed in Section 2.4). The manner in which data points are classified differs per outlier-selection algorithm. In formal terms, an outlier-selection algorithm is defined as follows.

Definition 2.1 (Outlier-selection algorithm). *An outlier-selection algorithm is a mapping* $f : \mathcal{X} \to \{\text{outlier}, \text{inlier}\}$ *and* $f(\mathbf{x})$ *is said to be the classification of data point* $\mathbf{x} \in \mathcal{X}$.

Each outlier-selection algorithm discussed in the thesis is based on an outlier-scoring algorithm, which outputs an outlier score instead of a classification. An outlier score is a scalar that signifies the degree of 'outlierness' of a data point. An outlier-scoring algorithm is formally defined as follows.

Definition 2.2 (Outlier-scoring algorithm). *An outlier-scoring algorithm is a mapping* $\phi : \mathcal{X} \to \mathbb{R}$ *and* $\phi(\mathbf{x})$ *is said to be the outlier score of data point* $\mathbf{x} \in \mathcal{X}$.

Definitions 2.1 and 2.2 are adapted from the definitions of binary classification and scoring algorithms as formulated by Vanderlooy (2009).

The computed outlier scores are transformed into classifications $\{\text{outlier}, \text{inlier}\}$. A classification of data point $\mathbf{x}$ is obtained by applying a threshold $\theta \in \mathbb{R}$ on the outlier score of data point $\mathbf{x}$ as computed by the outlier-scoring algorithm ϕ.

$$f(\mathbf{x}) = \begin{cases} \text{outlier} & \text{if } \phi(\mathbf{x}) > \theta, \\ \text{inlier} & \text{if } \phi(\mathbf{x}) \leq \theta, \end{cases} \tag{2.2}$$

where the value of the threshold θ is set by the domain expert (see Section 2.4).

Because each outlier-scoring algorithm presented in the thesis, is transformed similarly into an outlier-selection algorithm (namely, by applying a threshold to the outlier scores), we do not distinguish between the outlier-selection algorithm and the underlying outlier-scoring algorithm. In the remainder of the thesis we only use the term outlier-selection algorithm, or algorithm for short, and we may say that it both computes outlier scores and classifies data points as outlier or inlier.

2.4 Domain expert

In this Subsection we discuss the concept of domain expert, and how it relates to the other concepts. The data flow diagram in Figure 2.1 shows four data transfers from the domain expert: (1) parameters to the outlier-selection algorithm, (2) labels: $\{\text{anomaly}, \text{normality}\}$ (of which the line is dashed) to the data set, (3) features or dissimilarity measure to represent the real-world observations as data, and (4) anomaly detections to the real world. We discuss each data transfer below.

Regarding (1), the domain expert configures the parameters of the outlier-selection algorithm. Each application domain, i.e., the corresponding data sets, requires a specific configuration of the parameters for the outlier-selection algorithm to perform optimally. In subsequent chapters we come across the different kinds of parameters.

Regarding (2), when an expert labels a real-world observation as an anomaly, we may also say: (1) that the observation is detected as an anomaly, and (2) that the observation belongs to the anomaly class $\mathcal{C}_A$. This is illustrated in the Euler diagram in Figure 2.2(d). For some application domains, these labels are available to the outlier-selection algorithm. In our comparative experiments, the labels provided by the domain expert are considered to be the ground truth. In other words, the domain expert is always

right. The classifications made by the outlier-selection algorithm are compared to the labels provided by the domain expert (see Section 2.5).

Regarding (3), in Section 2.2 we explained two approaches to represent real-world observations as data points. In a real-world application, the domain expert has the domain knowledge to decide which approach is most appropriate. We assume that the features or the dissimilarity measure are defined by the domain expert.

Regarding (4), the outlier-selection algorithm outputs classifications in order to *support* the domain expert, who has to decide whether the classification is valid and whether further action is required. In the maritime domain, an example action is for the coastguard operator to send a rescue helicopter.

2.5 Hits, misses, correct rejects, and false alarms

From the above we know that a domain expert labels and an outlier-selection algorithm classifies. They may not always agree with each other, as will be explained in this section. Comparing the classifications $\{\text{outlier}, \text{inlier}\}$ with the labels $\{\text{anomaly}, \text{normality}\}$ is necessary for the evaluation of the outlier-selection algorithm (see Section 2.6).

Figure 2.5 shows a confusion matrix (Mitchell, 1997; Witten and Frank, 2005). The confusion matrix shows the four possible outcomes when we compare the label of the real-world observation given by the expert to the classification of the corresponding data point given by the algorithm. The possible outcomes (and the four corresponding sets) are: (1) hit ($\mathcal{H}i$), (2) false alarm ($\mathcal{FA}$), (3) miss ($\mathcal{M}i$), and (4) correct reject ($\mathcal{CR}$).

As we can see in Figure 2.5, if the expert labels an observation as an anomaly and the algorithm classifies the corresponding data point as an outlier, then the outcome is a 'hit'. In this case, the expert and the algorithm agree with each other. The hit square in Figure 2.5 is green colour to illustrate agreement between the expert and the algorithm. The expert and the algorithm also agree with each other when the expert labels a real-world observation as a normality and the algorithm classifies the data point as an inlier. This results in a correct reject, which means that the outlier-selection algorithm has correctly rejected the hypothesis that the observation is anomalous. The corresponding square is coloured olive. The remaining two outcomes (i.e., false alarm and miss) are the result of disagreement between the expert and the algorithm. The squares corresponding to false alarm and miss are located in the top right and bottom left. The two squares are coloured purple and red to indicate disagreement. We remark that the colours used

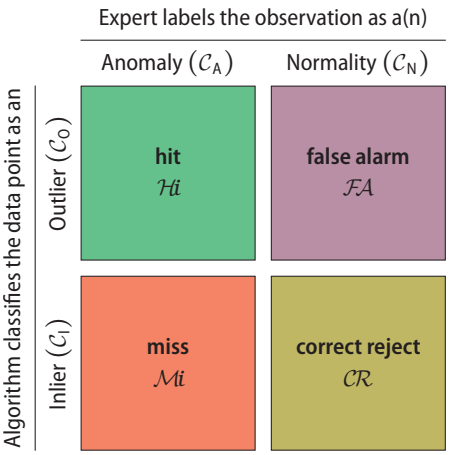


Figure 2.5 Confusion matrix showing the four possible outcomes when the label of the expert concerning a real-world observation is compared with the classification of the algorithm concerning the corresponding data point. The four possible outcomes are: hit, false alarm, miss, and correct rejection.

in Figure 2.5 are employed consistently throughout the thesis to indicate the outcomes. Formally, the four sets corresponding the outcomes are defined as follows.

$$\mathcal{H}i = \left\{\mathbf{x} \in \mathcal{D} \;\middle|\; \mathbf{x} \in \mathcal{C}_A \text{ and } \mathbf{x} \in \mathcal{C}_O\right\} \tag{2.3}$$

$$\mathcal{FA} = \left\{\mathbf{x} \in \mathcal{D} \;\middle|\; \mathbf{x} \in \mathcal{C}_N \text{ and } \mathbf{x} \in \mathcal{C}_O\right\} \tag{2.4}$$

$$\mathcal{M}i = \left\{\mathbf{x} \in \mathcal{D} \;\middle|\; \mathbf{x} \in \mathcal{C}_A \text{ and } \mathbf{x} \in \mathcal{C}_I\right\} \tag{2.5}$$

$$\mathcal{CR} = \left\{\mathbf{x} \in \mathcal{D} \;\middle|\; \mathbf{x} \in \mathcal{C}_N \text{ and } \mathbf{x} \in \mathcal{C}_I\right\} \tag{2.6}$$

The four sets are also illustrated in the Euler diagram in Figure 2.2(f) on page 19. This concludes our discussion of Figure 2.2.

Let us illustrate the four possible outcomes using an example data set as shown in Figure 2.6. Here we note that the four scatters plots show the same example data set; only the colours and the shapes of the data points are different.

The top-left scatter plot shows the example data set as it is available to the outlier-selection algorithm. The data set contains two clusters of data points. The small cluster on the left is much more dense than the big cluster on the right. There are two data points that lie

somewhat outside the clusters. It is up to the outlier-selection algorithm to process the data set and to classify each data point.

The top-right scatter plot shows the labels given by the domain expert. The two data points that lie somewhat outside of the clusters, appear to be anomalies according to the domain expert. The remaining data points are labelled as normalities. The panel inside the scatter plot zooms in on the area around the left anomaly. This is to show that on a smaller scale, the anomaly lies indeed somewhat outside the small, dense cluster. In passing, we note that the labels are not available to the outlier-selection algorithm.

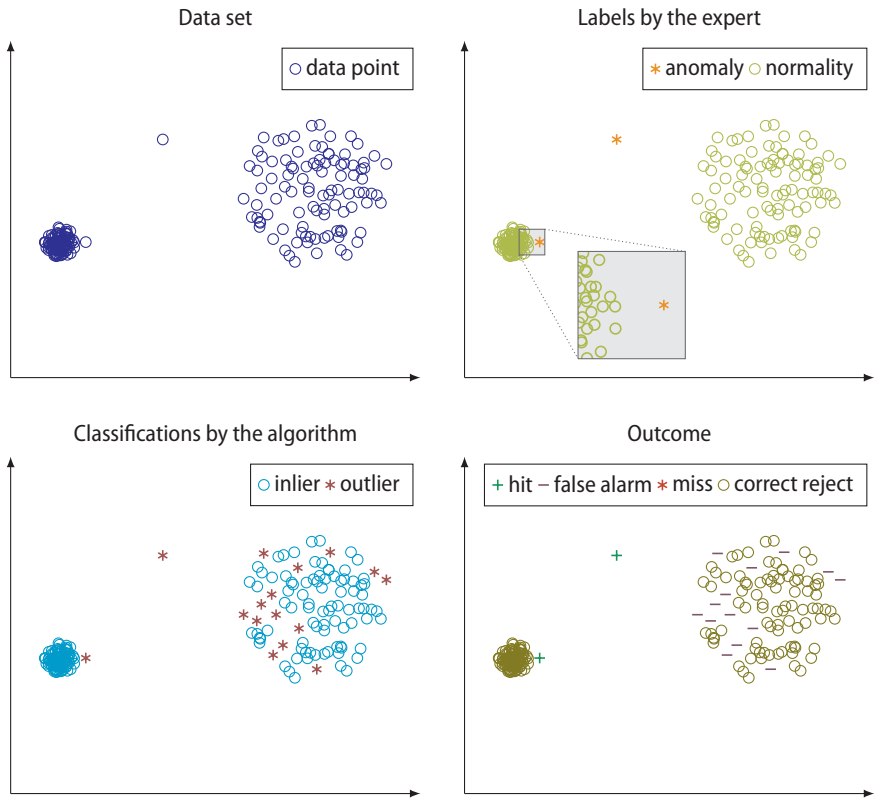


Figure 2.6 An illustration of the four possible outcomes. **Top left:** Real-world observations represented as data points. **Top right:** The domain expert labels two observations as anomalous. **Bottom left:** The outlier-selection algorithm classifies many data points as outliers. **Bottom right:** Comparing the classifications with the labels results in four possible outcomes.

The bottom-left scatter plot shows the classifications made by the outlier-selection algorithm. The algorithm selects as outliers (1) the two anomalies and (2) many data points that are inside the large cluster.

Finally, the bottom-right scatter plot shows the four possible outcomes when we compare the classifications made by the outlier-selection algorithm, and the labels given by the domain expert. In this example, there are no misses because the outlier-selection has classified all anomalies as outliers. However, there are many false alarms, because many normalities in the large cluster have been incorrectly classified as outliers.

The large number of false alarms may indicate that the domain expert has chosen a suboptimal threshold. If the domain expert would have set the threshold to a higher value, then the left-most anomalous data point may have been classified as an inlier, which would result in a miss (cf. Equation 2.2). In Section 2.6 and Subsection 2.6.3 specifically, we further explain choosing appropriate thresholds.

2.6 Evaluating the performance of an outlier-selection algorithm

In this section we discuss the evaluation of the performance of an outlier-selection algorithm. Evaluation comes down to measuring the classification performance of an algorithm on a particular data set. For this, we employ the labels associated with each data point, even though these were not available to the algorithm. When the performances of multiple outlier-selection algorithms are measured, they can be compared (how this precisely takes place is explained in Section 2.7).

Our description of the evaluation of outlier-selection algorithms is structured as follows. In Subsection 2.6.1, we discuss how binary- and multi-classification data sets can be transformed into one-class data sets such that they are usable for evaluating outlier-selection algorithms. In Subsection 2.6.2, we describe a procedure that simulates anomalies using one-class data sets. In Subsection 2.6.3, we explain how the performance measure known as Area Under the ROC Curve (AUC) is computed. In Subsection 2.6.4, we describe the weighted version of AUC, which is appropriate for aggregating over multiple data sets.

2.6.1 Constructing one-class data sets from a multi-class data set

To evaluate the performance of an outlier-selection algorithm, we need data sets where the data points are labelled as normal and anomalous. Such a data set is called a one-class data set $\mathcal{D}$. In order to obtain a good sense of the characteristics of an outlier-selection algorithm we need a large number of varied one-class data sets.

One-class data sets are not as abundant as multi-class data sets ($\mathcal{D}_{\mathrm{M}}$). A multi-class data set contains two or more classes $\mathcal{C}$ that are not necessarily labelled as normal or anomalous;

$$\mathcal{D}_{\mathrm{M}} = \bigcup_{i=1}^{m} \mathcal{C}_i \,, \tag{2.7}$$

where m is the number of classes. For example, the Iris flower data set (Fisher, 1936) consists of 50 data points from each of the three classes: (1) Setosa, (2) Versicolor, and (3) Virginica. Figure 2.7(top) shows a scatter plot of the Iris flower data set. Normally, multi-class data sets are used for binary classification (Asuncion and Frank, 2010).

We can construct a one-class data set from a multi-class data set by relabelling one class as the normal class ($\mathcal{C}_{\mathrm{N}}$) and the remaining $m - 1$ classes as anomalous ($\mathcal{C}_{\mathrm{A}}$). Let

$$\mathcal{D}_i = \mathcal{C}_{\mathrm{A}} \cup \mathcal{C}_{\mathrm{N}} \text{ such that } \mathcal{C}_{\mathrm{A}} = \bigcup_{\substack{j=1 \\ j \neq i}}^{m} \mathcal{C}_j \text{ and } \mathcal{C}_{\mathrm{N}} = \mathcal{C}_i. \tag{2.8}$$

For a multi-class data set containing m classes, we can repeat this m times, where each class is relabelled as the normal class once (Tax, 2001). We remark that a one-class data set contains the same data points as the multi-class data set, but with different labels. The bottom row of Figure 2.7 shows three one-class datasets: $\mathcal{D}_1$, $\mathcal{D}_2$, and $\mathcal{D}_3$ that are constructed from the Iris flower data set. These one-class data sets are suitable for evaluating an outlier-selection algorithm.

2.6.2 Simulating anomalies

In data sets that are obtained from multi-class data sets, as described in Subsection 2.6.1, usually both the normal and the anomalous class are well-represented, i.e., clustered. As such, an unsupervised outlier-selection algorithm would not classify any anomalous data points as outliers. Figure 2.8 shows a scatter plot of a 'Banana' data set where both

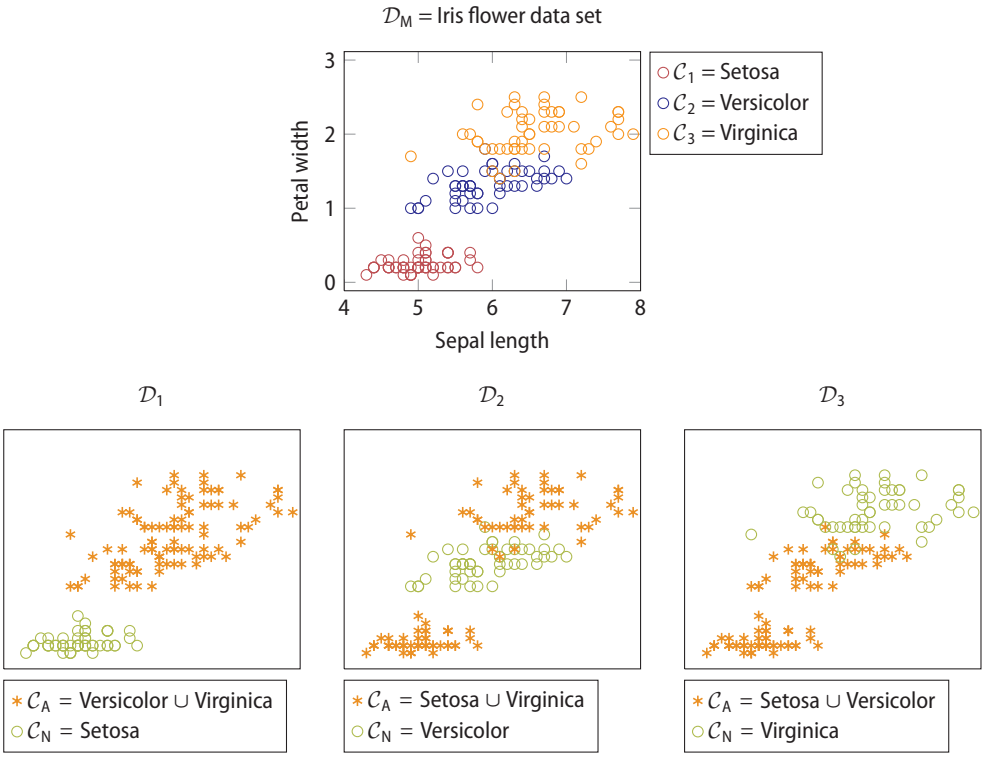


Figure 2.7 Illustration of relabelling a multi-class data set into multiple one-class data sets. **Top:** The Iris flower data set is a multi-class data set that consists of three classes: Setosa (○), Versicolor (○), and Virginica (○). **Bottom:** By relabelling the data points, three one-class data sets are obtained. Each of the three classes is made once the normal class (○).

classes are well-represented. (This data set will be used for the remainder of this chapter to explain the evaluation procedure.)

In order to use such a data set for the evaluation of outlier-selection algorithms, we employ a three-step procedure that simulates the anomalies to be rare. First, all data points of the anomalous class are removed from the data set. Second, the outlier-selection algorithm computes the outlier scores for all normal data points. Third, we add one of the anomalous data points and compute its outlier score, and remove it thereafter. The third step is repeated until all anomalous data points have been processed by the outlier-selection algorithm. The result is an outlier score for each normality and anomaly.

Figure 2.9 shows for the outlier scores of all data points in the Banana data set. The vertical dotted line separates the anomalies ($\mathcal{C}_A$) from the normalities ($\mathcal{C}_N$). (The other elements in the figure are explained in Subsection 2.6.3.) We note that overall the anomalies have a

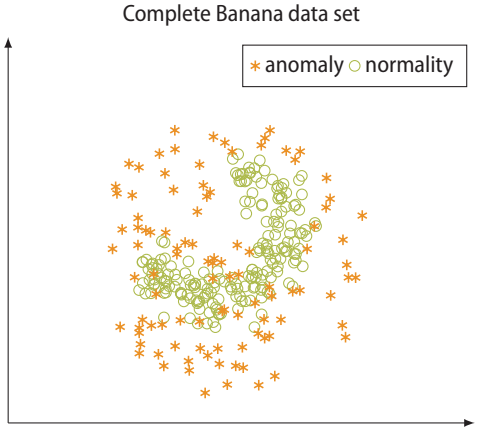


Figure 2.8 The complete Banana data set. In order to evaluate the outlier-selection algorithm we simulate anomalies to be rare.

higher outlier score than the normalities. The outlier scores are used in Subsection 2.6.3 to measure the AUC performance of the outlier-selection algorithm.

2.6.3 Area Under the Curve performance measure

In this section we explain how we measure the performance of an outlier-selection algorithm. In the thesis we employ the AUC performance measure (Bradley, 1997).

The computed outlier scores from Subsection 2.6.2 are converted to classifications $\{\text{outlier}, \text{inlier}\}$ using a threshold θ (see Equation 2.2 on page 24), so that these can be compared to the labels $\{\text{anomaly}, \text{normality}\}$ provided by the expert, resulting in a certain number of hits, false alarms, misses, and correct rejections. The four outcomes introduced in Section 2.5 have four associated rates.

$$\text{hit rate} = |\mathcal{H}i| / |\mathcal{C}_\text{A}| \tag{2.9}$$

$$\text{false alarm rate} = |\mathcal{FA}| / |\mathcal{C}_\text{N}| \tag{2.10}$$

$$\text{miss rate} = |\mathcal{M}i| / |\mathcal{C}_\text{A}| \tag{2.11}$$

$$\text{correct reject rate} = |\mathcal{CR}| / |\mathcal{C}_\text{N}| \tag{2.12}$$

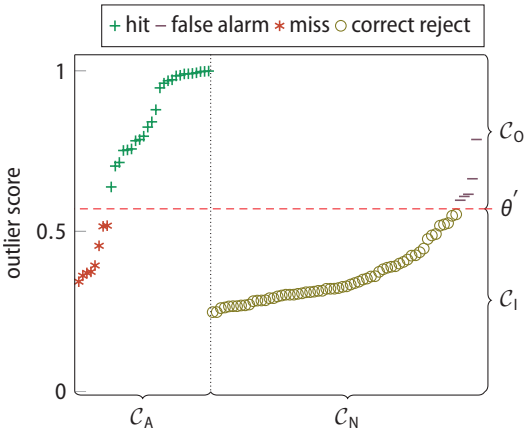


Figure 2.9 Outlier scores of the data points in the Banana data set. The dashed line indicates the threshold chosen by the domain expert.

Unless the outlier-selection algorithm classifies every data point correctly, there exists a trade-off between the hit rate and the false alarm rate. The rates are determined by the threshold θ, of which the optimal setting depends on the application and the associated cost of misclassification. By changing the threshold θ from the lowest possible value to the highest possible value, we obtain a range of *false alarm rates* and *hit rates*.

A plot of the range of *false alarm rates* against the range of *hit rates* is called an ROC curve. Figure 2.10 shows the ROC curve for the Banana data set that corresponds to the outlier scores shown in Figure 2.9. An ROC curve shows us how the threshold influences the *false alarm rate* and the *hit rate*. The threshold $\theta' = 0.57$ minimises the number of false alarms while maximising the number of hits. This threshold corresponds to the dashed horizontal line in Figure 2.9.

By integrating the ROC curve we obtain the Area Under the ROC Curve (AUC). The AUC is always between 0 to 1, where a higher value is better. Therefore, the AUC expresses the performance of an outlier-selection algorithm in a single value that is independent of the threshold (Vanderlooy, 2009).

A completely random outlier-selection algorithm, which outputs random classifications, has on average, an AUC of 0.5. Employing the AUC as performance measure avoids the use of one particular threshold θ and ensures a fair comparison of algorithms (Bradley, 1997). Of course, in a real-world setting the algorithm will be applied with a threshold θ that is optimal for the task at hand.

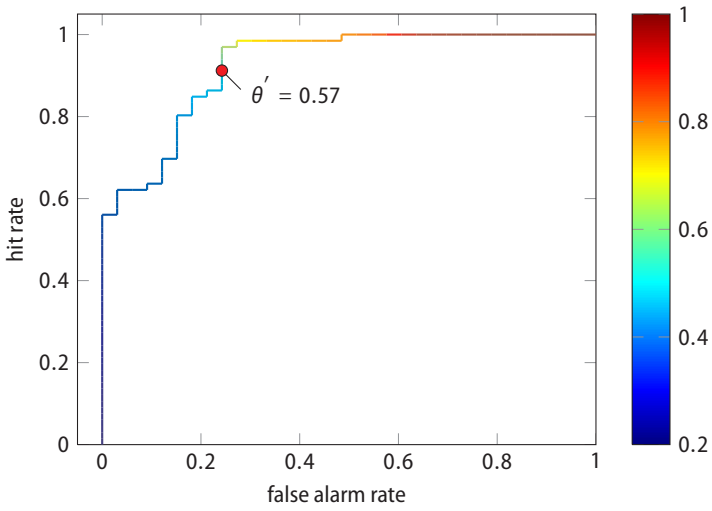


Figure 2.10 An ROC curve plots the false alarm rate agains the hit rate for all possible thresholds.

Definition 2.3 (AUC performance). *The AUC of an outlier-selection algorithm ϕ, given a data set $\mathcal{D}$, is defined as*

$$\mathrm{AUC}(\phi, \mathcal{D}) = \frac{1}{|\mathcal{C}_\mathrm{A}| \cdot |\mathcal{C}_\mathrm{N}|} \sum_{\mathbf{x}_\mathrm{A} \in \mathcal{C}_\mathrm{A}} \sum_{\mathbf{x}_\mathrm{N} \in \mathcal{C}_\mathrm{N}} \mathbb{1}\{\phi(\mathbf{x}_\mathrm{A}) > \phi(\mathbf{x}_\mathrm{N})\} \;, \tag{2.13}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function that returns a value of 1 if the anomaly $\mathbf{x}_\mathrm{A}$ has a higher outlier score than the normality $\mathbf{x}_\mathrm{N}$, and 0 otherwise.

Please note that the AUC performance measure treats both the misclassification of an anomaly and a normality as equal (Fawcett, 2004). This may not be optimal is a real-world situation, but since we cannot assume anything about the data sets used in our experiments, the AUC is a suitable performance measure.

2.6.4 Weighted AUC

When applying an outlier-selection algorithm to multiple one-class data sets that are constructed from the same multi-class data set, we report performances with the help of the weighted AUC.

To compute the weighted AUC, i.e., multi-class AUC (cf. Fawcett, 2004), the AUCs of the one-class data sets are averaged, where each one-class data set is weighted according to the prevalence of the normal class, $\mathcal{C}_{\mathrm{N}i}$.

$$\mathrm{AUC}_{\mathrm{M}}\left(\phi, \mathcal{D}_{\mathrm{M}}\right) = \sum_{i=1}^{m} \mathrm{AUC}\left(\phi, \mathcal{D}_i\right) \cdot \frac{\left|\mathcal{C}_{\mathrm{N}i}\right|}{\left|\mathcal{D}_{\mathrm{M}}\right|} \qquad 2.14$$

The use of a weighted average prevents one-class data sets containing few normalities from dominating the results (Hempstalk and Frank, 2008).

2.7 Method for comparing outlier-selection algorithms

When multiple outlier-selection algorithms have been evaluated on multiple data sets, we can compare their performances. A full list of (weighted) AUC performance measures allows for a detailed examination. We admit that raw numbers are inappropriate to arrive at general conclusions (cf. Demšar, 2006). In practice, we see that some machine learning papers report and draw conclusions from the average AUC performance[4] of outlier-selection algorithms across data sets from different applications domains. However, as Webb (2000) states, 'it is debatable whether error rates in different domains are commensurable, and hence whether averaging error rates across domains is very meaningful.' Therefore, as we explain in this section, we test for significant difference between the outlier-selection algorithms.

To compare multiple algorithms on multiple datasets, we follow Demšar (2006), who suggests to apply the following three steps. The first step is to apply the statistical Friedman test (Subsection 2.7.1). When the first step has a positive outcome we continue with the second step, which is to apply the post-hoc Nemenyi test (Subsection 2.7.2). The third step is to visualise the outcome of the second step (i.e., the significant differences between the algorithms) using a critical difference diagram (Subsection 2.7.3). We conclude the section in Subsection 2.7.4.

2.7.1 Friedman test

The Friedman test (Friedman, 1937) is used to investigate whether there is a significant difference between the performances of the outlier-selection algorithms. The Friedman test first ranks the algorithms for each data set, where the best performing algorithm is assigned the rank of 1, the second best the rank of 2, and so forth.

[4] Please note that the weighted AUC presented in Subsection 2.6.4 aggregates AUC performances for one-class data sets from a single multi-class data set.

Then it checks whether the measured average ranks R_j^2 are significantly different from the mean rank. For example, if there are four outlier-selection algorithms, the mean rank is 2.5. Iman and Davenport (1979) proposed the F_F statistic, which is less conservative than the Friedman statistic:

$$F_F = \frac{(N-1)\,\chi_F^2}{N(m-1)-\chi_F^2}, \tag{2.15}$$

where N is the number of data sets, m is the number of algorithms, and χ_F^2 is the Friedman statistic:

$$\chi_F^2 = \frac{12N}{m(m+1)}\left(\sum_j R_j^2 - \frac{m(m+1)^2}{4}\right). \tag{2.16}$$

The F_F statistic is distributed according to the F-distribution with $m - 1$ and $(m-1)(N-1)$ degrees of freedom. When there is a significant difference, we proceed with the post-hoc Nemenyi test.

2.7.2 Nemenyi test

The post-hoc Nemenyi test (Nemenyi, 1963) checks for each pair of outlier-selection algorithms whether there is a significant difference in performance. The performance of two outlier-selection algorithms is significantly different when the difference between their average ranks is greater than or equal to the critical difference:

$$\mathrm{CD} = q_\alpha \sqrt{\frac{m(m+1)}{6N}}, \tag{2.17}$$

where q_α is the Studentised range statistic divided by $\sqrt{2}$. The next step is to report the post-hoc analysis using a critical difference diagram.

2.7.3 Critical difference diagram

Critical difference diagrams are a visual representation of the post-hoc analysis of comparing multiple outlier-selection algorithms (Demšar, 2006). Figure 2.11 shows a critical difference diagram for four algorithms (C4.5, C4.5+m, C4.5+cf, C4.5m+cf). This example is taken from (Demšar, 2006)[5]. The arrows indicate the average ranks of the algorithms. Groups of algorithms that are not significantly different at a significance level

[5] The actual algorithms are not important, but the interested reader may want to know that the C4.5 is a top-down decision tree induction algorithm, and the other three algorithms are variations on C4.5.

of 0.10 (i.e., $p = .10$) are connected by a horizontal bar. The critical difference diagram reveals that two pairs of algorithms have a significantly different performance: (1) C4.5 and C4.5+m and (2) C4.5 and C4.5m+cf.

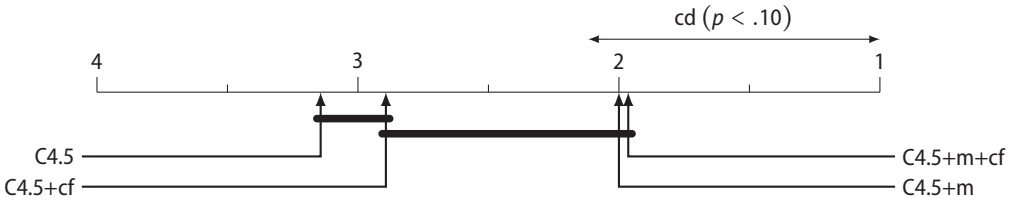


Figure 2.11 A critical difference diagram visualises the result of a comparison of algorithms with the Nemenyi test. Groups of algorithms that are not significantly different (at $p = .10$) are connected by a horizontal bar (after (Demšar, 2006)).

A critical difference diagram offers three advantages over presenting the results in textual or numerical form. First, it presents the order of the outlier-selection algorithms in terms of average ranks. Second, it shows the magnitude of the differences between the average ranks. Third, it indicates whether the differences are significant.

2.7.4 Section conclusion

In the section we described the three steps for comparing multiple algorithms on multiple data sets. The comparisons to be performed in the subsequent chapters involve all three steps. However, because in the thesis the Friedman test (step 1) always results in a positive outcome, we do not explicitly report it. Instead, we report the results of the Nemenyi test (step 2) using a critical difference diagram (step 3).

2.8 One-class classification setting

We start recalling that the data flow diagram in Figure 2.1 on page 17 contains a dashed line from the domain expert to the data set. The dashed line indicates that the labels $\{\text{anomaly}, \text{normality}\}$ are not necessarily transferred. If the labels are transferred, then the data set is labelled, otherwise it is unlabelled.

So far, our description of evaluating and comparing outlier-selection algorithms has assumed that algorithms process unlabelled data sets, only. This means that the outlier-selection algorithm does not know whether the domain expert considers a data point to

be a normality or an anomaly. We refer to this assumption as the unsupervised outlier-selection setting.

In certain real-world situations the domain expert has labelled some real-world observations. Then, we can employ a *semi*-supervised setting known as the one-class classification setting (Tax, 2001), which is the subject of this section.

The remainder of this section is structured as follows. In Subsection 2.8.1 we describe the similarities and differences between an unsupervised outlier-selection algorithm and a semi-supervised outlier-selection algorithm, i.e., a one-class classifier. In Subsection 2.8.2 we explain how a one-class classifier is trained and tested. In Subsection 2.8.3 we describe the cross validation, which is used to generalise the performance measure.

2.8.1 One-class classifier

Algorithms that make use of the labels provided by the domain expert are known as one-class classifiers. A one-class classifier is named as such since it makes use of one class only, namely the normal class. So, in this setting it is assumed that anomalies are rare.[6]

A one-class classifier takes as input a labelled data set $\mathcal{D}$ and outputs classifications $\{\text{outlier}, \text{inlier}\}$. Therefore, Definition 2.1 and consequently Definition 2.2 on page 23 also apply to one-class classifiers.

2.8.2 Training and testing

The main difference between a one-class classifier and an outlier-selection algorithm is that a one-class classifier needs to be *trained* before it can classify data points. We let the training data set contain only normalities, in order to mimic the low number of anomalies that occur in most real-world applications. Figure 2.12 illustrates a trained one-class classifier on the Banana data set. The red dashed line represents the selection boundary of the one-class classifier. If a new, unseen data point, i.e., a test data point, lies outside the selection boundary, it is classified as an outlier. Otherwise, the data point is classified as an inlier. Analogously to an outlier-selection algorithm, the classifications are compared to the labels of the domain expert, which results in the four outcomes

[6] If the data set contains sufficient data points from both the normal class and the anomalous class, then the domain expert may be better supported by a supervised binary classifier, which is beyond the scope of the thesis.

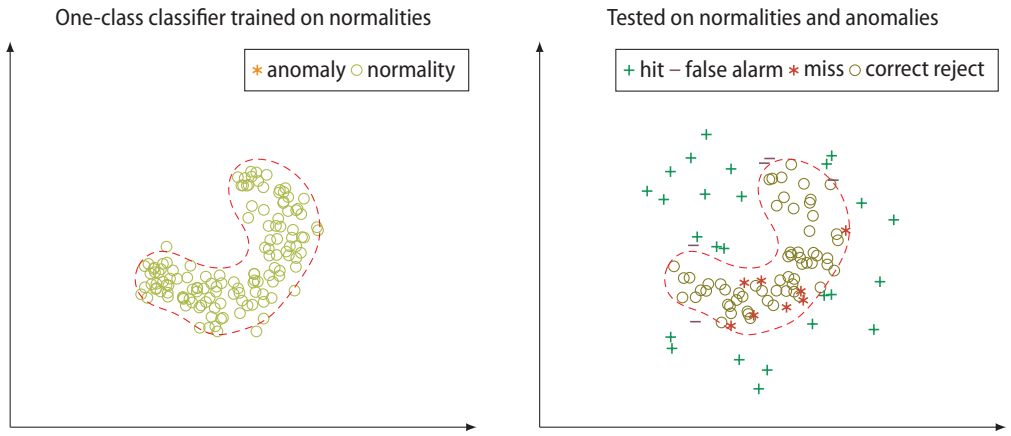


Figure 2.12 Training and testing a one-class classifier. **Left:** A one-class classifier is trained on the normal class only. The dashed line represents the decision boundary. **Right:** A trained one-class classifier is tested on both classes, which results in four possible outcomes.

introduced in Section 2.5. The scatter plot on the right-hand-side in Figure 2.12 shows the (dis)agreements after testing a trained one-class classifier (cf. Section 2.5).

Training a one-class classifier is also known as the one-class classifier building a model of the normal class. Once trained, the model remains fixed. As a result, a one-class classifier requires an evaluation procedure that differs from an outlier-selection algorithm.

A one-class classifier evaluates each testing data point by itself. In other words, the classification of a testing data point does not depend on other testing points, but only on the model. So, for a one-class classifier, the procedure to simulate anomalies (described in Subsection 2.6.2) is not needed. Because an outlier-selection algorithm does not build a model, but re-processes the entire data set, it is usually slower than a one-class classifier. As we will explain in the next subsection, for a proper evaluation of a one-class classifier, it is important that it is tested on unseen data points.

2.8.3 Cross validation

To estimate the AUC performance of a one-class classifier, we employ a technique called cross validation. In k-fold cross validation, e.g., $k = 10$, the data set $\mathcal{D}$ is split randomly into k mutually exclusive subsets, i.e., folds, $\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_k$ of equal size (Kohavi, 1995). We ensure that the cross-validation is stratified, which means each fold contains both normalities and anomalies with the same proportions as the original data set. The one-class classifier is trained and tested k times. Each time $t \in 1, 2, \ldots, k$, the one-class

classifier is trained on $\mathcal{D} \setminus \mathcal{D}_t$ and tested on $\mathcal{D}_t$. (We recall that a one-class classifier is trained on the normalities only.) This results in k AUC performances, which are averaged to one AUC performance. In our experiments, we employ ten-fold cross validation. To minimise the bias introduced by the random selection of folds, we repeat the ten-fold cross-validation ten times. Figure 2.13 shows a scatter plot of a training set and a test set, respectively. The training set consists of nine folds and the test consists of one fold.

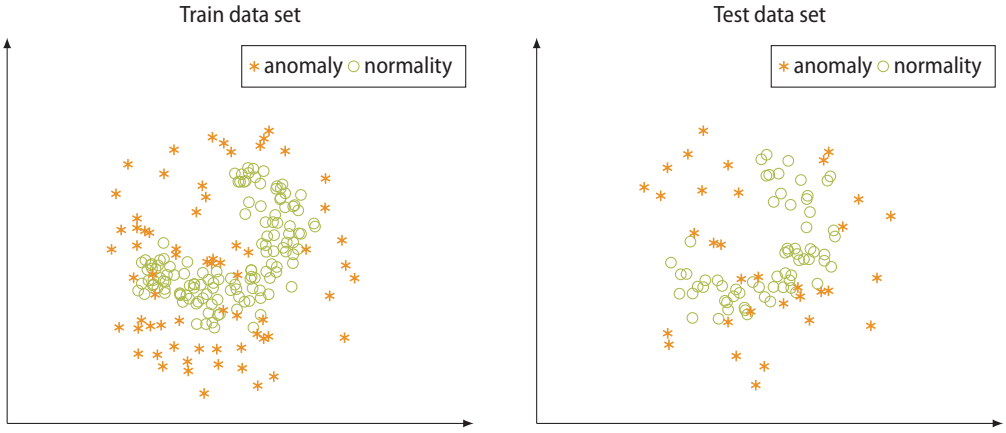


Figure 2.13 To evaluate an outlier-selection algorithm, the data set is split into a training data set and a test data set.

2.9 Chapter summary

In this chapter we introduced the main concepts of the thesis and explained the experimental set-up that is employed for answering RQ1, RQ2, and RQ3. We illustrated the relationship between (1) the domain expert, (2) the outlier-selection algorithm, (3) the real-world, (4) the data representation, and (5) the data set.

Outlier selection can be performed within two settings: (1) the unsupervised outlier-selection setting and (2) the one-class classification setting, which is semi-supervised. We started our explanation of the concepts by assuming the unsupervised setting. We described two types of data representations that outlier-selection algorithms can process: (1) feature-vector representation and (2) similarity-matrix representation. We outlined the two responsibilities of the domain expert, which are: (1) to configure the parameters of the outlier-selection algorithm and (2) to assess its classifications.

We described how to transform ordinary multi-class data sets into one-class data sets so that they can be used for our comparative experiments. We explained how the

performance of an outlier-selection algorithm is measured using the AUC performance measure. We discussed the three steps to compare the performances of multiple outlier-selection algorithms on multiple data sets, namely: (1) applying the Friedman statistical test, (2) applying the post-hoc Nemény statistical test, and (3) visualising the significant differences using a critical difference diagram. In the first seven sections we assumed the unsupervised outlier-selection setting, and in Section 2.8 we changed our assumption to the one-class classification setting. We described the technique of cross validation, which is used to evaluate one-class classifiers. With all the concepts and the experimental set-up explained, we are now ready to address RQ1.

Chapter 3

Evaluating and comparing outlier-selection algorithms

Contents:

In this chapter, we provide an answer to RQ1: *How should we evaluate and compare the performance of outlier-selection algorithms?* Two well-known unsupervised outlier-selection algorithms from the field of KDD are transformed into one-class classifiers so that these can be compared with three outlier-selection algorithms from the field of ML in a statistically valid way.

Based on:

- J.H.M. Janssens and E.O. Postma. One-class classification with LOF and LOCI: An empirical comparison. In M.G.J. van Erp, J.H. Stehouwer, and M.M. van Zaanen, editors, *Proceedings of the 18th Annual Belgian-Dutch Conference on Machine Learning*, pages 56–64, Tilburg, The Netherlands, June 2009.
- J.H.M. Janssens, I. Flesch, and E.O. Postma. Outlier detection with one-class classifiers from ML and KDD. In M.A. Wani, M. Kantardzic, V. Palade, L. Kurgan, and Y. Qi, editors, *Proceedings of the 8th International Conference on Machine Learning and Applications*, pages 147–153, Miami, FL, Dec. 2009.

Outline:

3.1 Outlier selection in ML and KDD. **3.2** ML outlier-selection algorithms. **3.3** KDD outlier-selection algorithms. **3.4** Experimental set-up. **3.5** Results. **3.6** Discussion. **3.7** Chapter conclusions.

The problem of outlier selection is well studied in the fields of Machine Learning (ML) and Knowledge Discovery in Databases (KDD). Both fields have produced their own outlier-selection algorithms and corresponding evaluation procedures. In ML, Nearest Neighbour (NN), Parzen Windows, and Support Vector Machines (SVM) are well-known algorithms that have variants which can be used for one-class classification (these are known as K-Nearest Neighbour Data Description (KNNDD), Parzen Window Data Description (PWDD), and Support Vector Data Description (SVDD)). Within the field of KDD, the heuristic local-density estimation algorithms LOF and LOCI are generally considered to be superior outlier-selection algorithms. To the best of our knowledge, so far, the performances of these ML and KDD algorithms have not been compared. As a direct consequence we are able to formulate the null hypothesis: all algorithms have an equal performance.

The first research question (RQ1) reads: *How should we evaluate and compare the performance of outlier-selection algorithms?* To answer this question we apply the following two steps. First, we formalise LOF and LOCI into algorithms with a one-class classification setting. Second, we evaluate the ML and KDD outlier-selection algorithms on 24 real-world data sets, and compare their results. For the evaluation and comparison we employ the techniques explained in Chapter 2.

Chapter 3 is organised as follows. Section 3.1 describes how outliers are selected in the fields of ML and KDD. In Sections 3.2 and 3.3 we describe the ML and KDD outlier-selection algorithms, respectively, and explain how they compute a measure of outlierness. We provide the set-up of our experiments in Section 3.4 and their results in Section 3.5. Section 3.6 discusses the results in terms of three observations: (1) Local density estimates outperform global density estimates, (2) LOF outperforms LOCI, and (3) Domain-based and density-based algorithms are competitive. Finally, Section 3.7 concludes by stating that the fields of ML and KDD have outlier-selection algorithms that are competitive in performance and deserve treatment on equal footing.

3.1 Outlier selection in ML and KDD

There is a growing interest in the automatic selection of abnormal or suspicious patterns in large data volumes to detect terrorist activity, illegal financial transactions, or potentially dangerous situations in industrial processes. The interest is reflected in the development and evaluation of outlier-selection algorithms (Tax and Duin, 1999;

Breunig et al., 2000; Tax, 2001; Papadimitriou et al., 2003). In recent years, outlier-selection algorithms have been proposed in two related fields: ML and KDD. Although both fields have considerable overlap in their objectives and subject of study, there appears to be some separation in the study of outlier-selection algorithms. In the ML field, outlier selection is generally based on data description algorithms inspired by NN, Parzen Windows, and SVM (Tax and Duin, 1999; Tax, 2001). These algorithms originate from statistics and pattern recognition, and have a solid theoretical foundation (Parzen, 1962; Schölkopf and Smola, 2002). In the KDD field, the LOF algorithm (Breunig et al., 2000) and the LOCI algorithm (Papadimitriou et al., 2003) are the two main algorithms for outlier selection. Like most algorithms from KDD, LOF and LOCI are targeted to process large volumes of data (Fayyad, Piatetsky-Shapiro, and Smyth, 1996a).

Interestingly, within both fields the evaluation of their outlier-selection algorithms occurs quite isolated from the other field. In the KDD field, LOF and LOCI are rarely compared to ML algorithms such as NN, PWDD, and SVDD (cf. Breunig et al., 2000; Papadimitriou et al., 2003) and in the ML field, LOF and LOCI are seldom mentioned. As a case in point, in the review of outlier-selection algorithms by Hodge and Austin (2004), LOF and LOCI are not mentioned at all.

Hido, Tsuboi, Kashima, Sugiyama, and Kanamori (2008) recently compared LOF, SVDD, and several other outlier-selection algorithms. However, their study is flawed because their approach has three serious drawbacks. First, the performances are obtained using a test set which contains the same normal data points as in the training set. Secondly, Hido et al. state that cross-validation is not available to LOF and SVDD. Thirdly, no statistical tests for significance of the obtained performances are carried out. In our opinion, a separate test set, cross-validation, and a statistical test are crucial for a sound empirical evaluation and comparison.

To the best of our knowledge, this is the first time that outlier-selection algorithms from the fields of ML and KDD are evaluated and compared in a statistically valid way, using a separate test set, cross-validation, and statistical tests.

For our purpose, we adopt the one-class classification setting (Tax, 2001). The setting allows outlier-selection algorithms from different settings to be (1) evaluated using the well-known performance measure AUC (Bradley, 1997), and (2) compared using statistically sound comparison test such as the Friedman test (Friedman, 1937) and the post-hoc Nemenyi test (Nemenyi, 1963).

The outlier-selection algorithms of which the performances are compared are: KNNDD, PWDD, and SVDD from the field of ML and LOF and LOCI from the field of KDD. de Ridder, Tax, and Duin (1998) and Tax (2001) already presented the ML algorithms as one-class classifiers. Therefore, we need only reformulate LOF and LOCI in terms of the one-class classification setting.

3.2 ML outlier-selection algorithms

In this section we briefly discuss the outlier-selection algorithms from the field of Machine Learning (ML). The three algorithms K-Nearest Neighbour Data Description (KNNDD), Parzen Window Data Description (PWDD), and Support Vector Data Description (SVDD) are explained in Subsections 3.2.1, 3.2.2, and 3.2.3, respectively.

3.2.1 K-Nearest Neighbour Data Description

The first ML algorithm is the K-Nearest Neighbour Data Description (KNNDD) by de Ridder, Tax, and Duin (1998). The outlier score computed by KNNDD is the ratio between two distances. The first distance is the distance between the test data point $\mathbf{x}_i$ and its k^{th} nearest neighbour in the training set $\text{NN}(\mathbf{x}_i, k)$. The second distance is the distance between the k^{th} nearest training data point and its k^{th} nearest neighbour. Formally:

$$\phi_{\text{KNNDD}}(\mathbf{x}_i, k, \mathcal{D}_{\text{train}}) = \frac{d\big(\mathbf{x}_i, \text{NN}(\mathbf{x}_i, k)\big)}{d\Big(\text{NN}(\mathbf{x}_i, k), \text{NN}\big(\text{NN}(\mathbf{x}_i, k), k\big)\Big)}. \tag{3.1}$$

The KNNDD algorithm is similar to LOF and LOCI in the sense that it locally samples the density. The main difference with LOF and LOCI is that KNNDD is less complex.

3.2.2 Parzen Window Data Description

The second ML algorithm is the Parzen Window Data Description (PWDD), which is based on Parzen Windows as proposed by Parzen (1962). PWDD estimates the probability density function of the normal class $\mathbf{x}_i$:

$$\phi_{\text{PWDD}}(\mathbf{x}_i, h, \mathcal{D}_{\text{train}}) = \frac{1}{Nh} \sum_{j=1}^{N} K\left(\frac{\mathbf{x}_i - \mathbf{x}_j}{h}\right), \tag{3.2}$$

where N is $|\mathcal{D}_{\text{train}}|$, h is a smoothing parameter, and K typically is a Gaussian kernel:

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2} . \tag{3.3}$$

The parameter h is optimised using a leave-one out maximum likelihood estimation (de Ridder et al., 1998). Since the outlier score ϕ_{PWDD} is a probability and not a distance, the threshold function (cf. Equation 2.2) for PWDD becomes:

$$f_{PWDD}(\mathbf{x}_i) = \begin{cases} \text{inlier} & \text{if } \phi_{\text{PWDD}}(\mathbf{x}_i, h, \mathcal{D}_{\text{train}}) \geq \theta, \\ \text{outlier} & \text{if } \phi_{\text{PWDD}}(\mathbf{x}_i, h, \mathcal{D}_{\text{train}}) < \theta, \end{cases} \tag{3.4}$$

such that a data point with a too low probability is classified as an outlier. PWDD, unlike LOF, LOCI, and KNNDD, estimates the density globally rather than locally.

3.2.3 Support Vector Data Description

The third ML algorithm is the Support Vector Data Description (SVDD) from Tax and Duin (1999). We confine ourselves to a brief description of this kernel-based data-description algorithm. The interested reader is referred to Tax and Duin (1999) and Tax (2001) for a full description of the SVDD algorithm.

SVDD is a domain-based outlier-selection algorithm inspired by Support Vector Machines (SVM) (Vapnik, 1995). It is domain based because, unlike LOF, LOCI, and PWDD, SVDD does not estimate the data density directly. Instead, it finds an optimal boundary around the normal class by fitting a non-linearly transformed hypersphere with minimal volume using the kernel trick, such that it encloses most of the normal data points. The optimal boundary is found using quadratic programming, where only distant normal data points are allowed to be outside the boundary (Vapnik, 1995; Schölkopf and Smola, 2002). The outlier score ϕ_{SVDD} is defined as the distance between data point $\mathbf{x}_i$ and the normal boundary. In our experiments we employ a Gaussian kernel of which the parameter s is found as described in Tax and Duin (1999).

3.3 KDD outlier-selection algorithms

In this section we describe two popular outlier-selection algorithms from the field of Knowledge Discovery in Databases (KDD), namely Local Outlier Factor (LOF) by

Breunig, Kriegel, Ng, and Sander (2000) and Local Correlation Integral (LOCI) by Papadimitriou, Kitagawa, Gibbons, and Faloutsos (2003). Both algorithms are based on *local* densities, meaning that they consider a data point to be an outlier when its surrounding space contains *relatively* few data points (i.e., when the data density in that part of the data space is relatively low).

We frame the KDD outlier-selection algorithms LOF and LOCI into the one-class classification setting (Janssens and Postma, 2009) by letting them compute an outlier score ϕ by: (1) constructing a neighbourhood around $\mathbf{x}_i$, (2) estimating the density of the neighbourhood, and (3) comparing this density to the neighbourhood densities of the neighbouring data points. The main difference between LOF and LOCI is that the latter considers densities at multiple scales (i.e., multiple levels of granularity). Subsections 3.3.1 and 3.3.2 explain how the three steps are implemented in LOF and LOCI, respectively.

3.3.1 Local Outlier Factor

The first KDD algorithm is the heuristic algorithm LOF (Breunig et al., 2000). The user needs to specify one parameter, k, which represents the number of neighbours constituting the neighbourhood used for assessing the local density. Below, we explain how LOF implements the three steps.

Step 1: Constructing the neighbourhood

In order to construct the neighbourhood of a data point $\mathbf{x}_i$, LOF defines the neighbourhood border distance d_{border} of $\mathbf{x}_i$ as the distance d from $\mathbf{x}_i$ to its k^{th} nearest neighbour $\text{NN}(\mathbf{x}_i, k)$:

$$d_{\text{border}}(\mathbf{x}_i, k) = d\big(\mathbf{x}_i, \text{NN}(\mathbf{x}_i, k)\big)\,. \tag{3.5}$$

Consequently, a neighbourhood $\mathcal{N}(\mathbf{x}_i, k)$ is constructed, containing all data points $\mathbf{x}_j$ of which the distance to $\mathbf{x}_i$ is not greater than the neighbourhood border distance d_{border}:

$$\mathcal{N}(\mathbf{x}_i, k) = \{\mathbf{x}_j \in \mathcal{D}_{\text{train}} \setminus \{\mathbf{x}_i\} \mid d\big(\mathbf{x}_i, \mathbf{x}_j\big) \leq d_{\text{border}}(\mathbf{x}_i, k)\}. \tag{3.6}$$

It should be noted that $\mathbf{x}_i$ is not included in the neighbourhood. Figure 3.1 illustrates the concept of the neighbourhood border distance for $k = 3$.

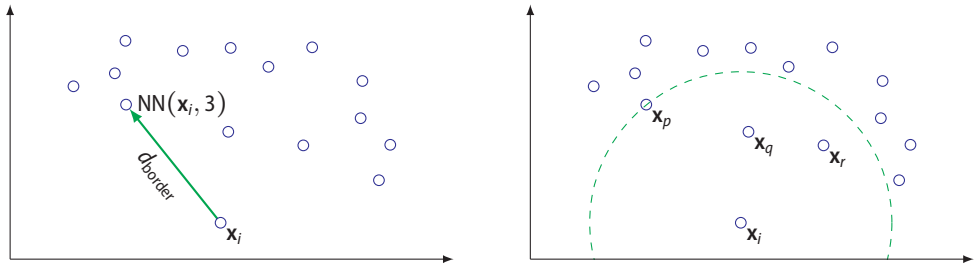


Figure 3.1 Illustration of the first step of LOF for $k = 3$ on a two-dimensional synthetic data set. **Left:** The border distance d_{border} is equal to the distance between data point $\mathbf{x}_i$ and its 3rd nearest neighbour. **Right:** The neighbourhood containing three data points.

Step 2: Estimating the neighbourhood density

To estimate the density of the constructed neighbourhood, the reachability distance is introduced. This distance ensures that a minimal distance between the two data points $\mathbf{x}_i$ and $\mathbf{x}_j$ is maintained, by 'keeping' data point $\mathbf{x}_i$ outside the neighbourhood of data point $\mathbf{x}_j$. The use of the reachability distance causes a smoothing effect of which the strength depends on the parameter k. The reachability distance d_{reach} is formally given by:

$$d_{\text{reach}}(\mathbf{x}_i, \mathbf{x}_j, k) = \max\left\{ d_{\text{border}}(\mathbf{x}_j, k), d(\mathbf{x}_j, \mathbf{x}_i) \right\}. \tag{3.7}$$

It should be noted that the reachability distance d_{reach} is an asymmetric measure. The neighbourhood density ρ of data point $\mathbf{x}_i$ depends on the number of data points in the neighbourhood, $\left|\mathcal{N}(\mathbf{x}_i, k)\right|$, and on their reachability distances. It is defined as:

$$\rho(\mathbf{x}_i, k) = \frac{\left|\mathcal{N}(\mathbf{x}_i, k)\right|}{\sum_{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i, k)} d_{\text{reach}}(\mathbf{x}_i, \mathbf{x}_j, k)}. \tag{3.8}$$

Data points $\mathbf{x}_j$ in the neighbourhood that are further away from data point $\mathbf{x}_i$, have a smaller contribution to the neighbourhood density $\rho(\mathbf{x}_i, k)$.

Step 3: Comparing the neighbourhood densities

In the third step, the neighbourhood density ρ of data point $\mathbf{x}_i$ is compared with those of its surrounding neighbourhoods. The comparison results in an outlier factor ω and requires the neighbourhood densities $\rho(\mathbf{x}_j, k)$ of the data points $\mathbf{x}_j$ that are inside the

neighbourhood of $\mathbf{x}_i$. The outlier factor ω is defined formally as:

$$\omega(\mathbf{x}_i, k) = \frac{\sum_{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i, k)} \frac{\rho(\mathbf{x}_j, k)}{\rho(\mathbf{x}_i, k)}}{\left|\mathcal{N}(\mathbf{x}_i, k)\right|}. \tag{3.9}$$

A data point which lies deep inside a cluster receives an outlier factor of around 1 because it has a neighbourhood density equal to its neighbours. Conversely, a data point which lies outside a cluster has a relatively low neighbourhood density and therefore receives a higher outlier factor.

LOF as one-class classifier

To fit LOF into the one-class classification setting it needs to compute an outlier score ϕ. We can interpret the local outlier factor ω from Equation 3.9 as an outlier score because a data point with a higher local outlier factor ω is considered to be a stronger outlier. Therefore, we reformulate LOF as a one-class classifier with an outlier score measure ϕ_{LOF} that is equivalent to the local outlier factor ω. The outlier score $\phi_{\text{LOF}}(\mathbf{x}_i, k, \mathcal{D}_{\text{train}})$ for a data point $\mathbf{x}_i$ with respect to a set of training data points $\mathcal{D}_{\text{train}}$ is defined as:

$$\phi_{\text{LOF}}(\mathbf{x}_i, k, \mathcal{D}_{\text{train}}) \equiv \omega(\mathbf{x}_i, k)\,.$$

3.3.2 Local Correlation Integral

The Local Correlation Integral (LOCI) was proposed by Papadimitriou, Kitagawa, Gibbons, and Faloutsos (2003) as an improvement over LOF. More specifically, the authors state that the choice of the neighbourhood size, k, in LOF is non-trivial and may lead to erroneous outlier selections. LOCI is claimed to be an improvement over LOF because it considers the local density at multiple scales or levels of granularity. Below we explain how LOCI implements the three steps (neighbourhood construction, density estimation, and density comparison) and we describe how we reformulate LOCI as a one-class classifier.

Step 1: Constructing the neighbourhood

LOCI differs from LOF because LOCI considers local densities at multiple scales. LOCI achieves this by iteratively performing the three steps, each time using a neighbourhood of increasing radius $r \in R^+$. Papadimitriou et al. describe how to determine the relevant radii r such that not every possible radius (or scale) has to be considered. We denote the set of relevant radii as $\mathcal{R}$.

A second difference with LOF is that LOCI defines two neighbourhoods for a data point $\mathbf{x}_i$: (1) the extended neighbourhood, $\mathcal{N}_{\text{ext}}$, and (2) the local neighbourhood, $\mathcal{N}_{\text{loc}}$. The extended neighbourhood of a data point $\mathbf{x}_i$ contains all data points $\mathbf{x}_j$ that are within radius r from $\mathbf{x}_i$:

$$\mathcal{N}_{\text{ext}}(\mathbf{x}_i, r) = \left\{\mathbf{x}_j \in \mathcal{D}_{\text{train}} \;\middle|\; d\left(\mathbf{x}_j, \mathbf{x}_i\right) \leq r\right\} \cup \mathbf{x}_i, \tag{3.10}$$

and the (smaller) local neighbourhood contains all data points that are within radius αr from data point $\mathbf{x}_i$:

$$\mathcal{N}_{\text{loc}}(\mathbf{x}_i, r, \alpha) = \left\{\mathbf{x}_j \in \mathcal{D}_{\text{train}} \;\middle|\; d\left(\mathbf{x}_j, \mathbf{x}_i\right) \leq \alpha r\right\} \cup \mathbf{x}_i, \tag{3.11}$$

where α defines the ratio between the two neighbourhoods ($\alpha \in (0, 1]$). Please note that $\mathbf{x}_i$ is included in the two neighbourhoods, contrarily to LOF. Figure 3.2 illustrates an extended neighbourhood containing six data points and the associated six local neighbourhoods, for one certain radius ($r = 25.0$, $\alpha = 0.5$).

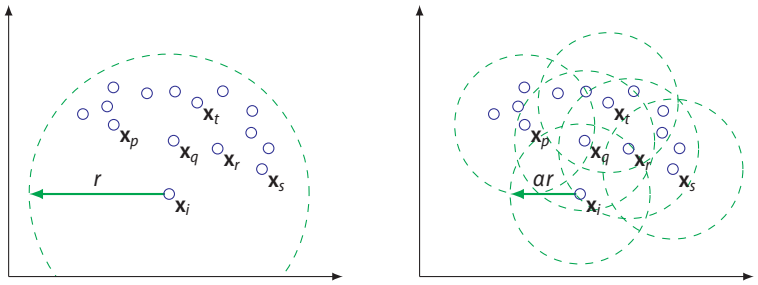


Figure 3.2 Illustration of the first step of LOCI applied to the same two-dimensional synthetic data set as shown in Figure 3.1 for $r = 25.0$, $\alpha = 0.5$. **Left:** The extended neighbourhood contains six data points. **Right:** The six local neighbourhoods with $\alpha r = 12.5$ for the data points within the extended neighbourhood.

Papadimitriou et al. recommend to define $\alpha < 1$ in order (1) to improve the density estimation of the extended neighbourhood and (2) to avoid singularities in the data point distribution. In case $\alpha = 1$, a singularity may occur, for example, when all data points in the neighbourhood of $\mathbf{x}_i$, except for $\mathbf{x}_i$ itself, lie on the neighbourhood border.

Step 2: Estimating the neighbourhood density

In LOCI, the density of the local neighbourhood of a data point $\mathbf{x}_i$ is denoted by $\rho(\mathbf{x}_i, \alpha r)$, and is equal to the number of data points in the local neighbourhood, i.e., $\left|\mathcal{N}_{\text{loc}}(\mathbf{x}_i, r, \alpha)\right|$. The extended neighbourhood of a data point $\mathbf{x}_i$ has a density $\hat{\rho}(\mathbf{x}_i, r, \alpha)$, which is defined as the average density of the local neighbourhoods of all data points in the extended neighbourhood of data point $\mathbf{x}_i$. In formal terms:

$$\hat{\rho}(\mathbf{x}_i, r, \alpha) = \frac{\sum_{\mathbf{x}_j \in \mathcal{N}_{\text{ext}}(\mathbf{x}_i, r)} \rho(\mathbf{x}_j, \alpha r)}{\left|\mathcal{N}_{\text{ext}}(\mathbf{x}_i, r)\right|}. \tag{3.12}$$

Step 3: Comparing the neighbourhood densities

The local neighbourhood density of data point $\mathbf{x}_i$ is compared to the extended neighbourhood density by means of the Multi-Granularity Deviation Factor (MDEF):

$$\text{MDEF}(\mathbf{x}_i, r, \alpha) = 1 - \frac{\rho(\mathbf{x}_i, \alpha r)}{\hat{\rho}(\mathbf{x}_i, r, \alpha)}. \tag{3.13}$$

MDEF quantifies the 'lagging behind' of the local neighbourhood density. A data point which lies deep inside a cluster has a local neighbourhood density equal to its neighbours and therefore receives an MDEF value around 0. The MDEF value approaches 1 as a data point lies more outside a cluster.

To determine whether a data point is an outlier, LOCI introduces the normalised MDEF, which is defined as:

$$\sigma_{\text{MDEF}}(\mathbf{x}_i, r, \alpha) = \frac{\sigma_{\hat{\rho}}(\mathbf{x}_i, r, \alpha)}{\hat{\rho}(\mathbf{x}_i, r, \alpha)}, \tag{3.14}$$

where $\sigma_{\hat{\rho}}(\mathbf{x}_i, r, \alpha)$ is the standard deviation of all $\rho(\mathbf{x}_j, \alpha r)$ in $\mathcal{N}_{\text{ext}}(\mathbf{x}_i, r)$. The normalised MDEF becomes smaller when the local neighbourhoods have the same density. This causes a cluster of uniformly distributed data points to have a tighter decision boundary than, for example, a Gaussian distributed cluster.

LOCI considers a data point $\mathbf{x}_i$ to be an outlier when, for any radius $r \in \mathcal{R}$,

$$\text{MDEF}(\mathbf{x}_i, r, \alpha) > k_\sigma \times \sigma_{\text{MDEF}}(\mathbf{x}_i, r, \alpha)\,, \tag{3.15}$$

where $k_\sigma = 3$ in Papadimitriou et al. (2003).

LOCI as one-class classifier

To use LOCI within the one-class classification setting, we reformulate LOCI as a one-class classifier by defining an outlier score. We define the outlier score ϕ_{LOCI} as the maximum ratio of MDEF to σ_{MDEF} of all radii $r \in \mathcal{R}$:

$$\phi_{\text{LOCI}}(\mathbf{x}_i, \alpha, \mathcal{D}_{\text{train}}) = \max_{r \in \mathcal{R}} \left\{ \frac{\text{MDEF}(\mathbf{x}_i, r, \alpha)}{\sigma_{\text{MDEF}}(\mathbf{x}_i, r, \alpha)} \right\}. \tag{3.16}$$

3.4 Experimental set-up

After the successful transformation of LOF and LOCI from the unsupervised outlier-selection setting into the one-class classification setting, we will now provide our experimental set-up. We aim to establish how the five outlier-selection algorithms from different settings can compared to each other. Reaching our aim means formulating an answer to RQ1.

This section describes the set-up of our experiments where we evaluate and compare the performances of KNNDD, PWDD, SVDD, LOF, and LOCI. For clarity, we have summarized the main features of the five outlier-selection algorithms in Table 3.1. For our experiments we employ the techniques described in Chapter 2. The techniques include: (1) transforming a multi-class data set into several one-class data sets, (2) cross validation, (3) the AUC performance measure, and (4) the Friedman and Neményi statistical tests.

In Subsection 3.4.1 we describe the data sets. The details of the cross-validation differ slightly from the ones presented in Chapter 2. Therefore, we describe the evaluation, i.e., the cross-validation procedure of the current experiment in Subsection 3.4.2.

Table 3.1 The main features of the ML and KDD outlier-selection algorithms used in our experiments.

	ML			KDD	
Feature	KNNDD	PWDD	SVDD	LOF	LOCI
Estimate local density	✓			✓	✓
Estimate global density		✓			
Domain based			✓		

3.4.1 Data sets

In order to evaluate the algorithms on a wide variety of data sets (i.e., varying in size, dimensionality, class volume overlap), we use 24 real-world multi-class data sets from the UCI Machine Learning Repository[1] (Asuncion and Frank, 2010) as redefined as one-class classification data sets by David Tax (`http://ict.ewi.tudelft.nl/~davidt`): *Arrhythmia*, *Balance-scale*, *Biomed*, *Breast*, *Cancer wpbc*, *Colon Gene*, *Delft Pump*, *Diabetes*, *Ecoli*, *Glass Building*, *Heart*, *Hepatitis*, *Housing*, *Imports*, *Ionosphere*, *Iris*, *Liver*, *Sonar*, *Spectf*, *Survival*, *Vehicle*, *Vowel*, *Waveform*[2], and *Wine*. Because these data sets contain multiple classes that are not necessarily defined as either normal or anomalous, they are relabelled into multiple one-class data sets using the procedure that is described in Subsection 2.6.1.

3.4.2 Evaluation

We apply the following procedure for the evaluation of an algorithm on a one-class data set. An independent test set containing 20% randomly selected data point of the entire data set is reserved. With the remaining 80% a 5-fold cross-validation procedure (see Subsection 2.8.3) is applied to optimise the parameters (i.e., $k = 1, 2, \ldots, 50$ for LOF and KNNDD; $\alpha = 0.1, 0.2, \ldots, 1.0$ for LOCI; h for PWDD; and s for SVDD). Each algorithm is trained with the optimal parameter value obtained by the cross-validation. Its AUC performance (see Subsection 2.6.3) is evaluated using the independent test set that was reserved at the beginning of this procedure. The entire procedure is repeated 5 times. We report on the performances of the algorithms applied on an entire multi-class data

[1] Except the Delft Pump data set which is taken from Ypma (2001).

[2] The author has been informed that *Waveform* is, in fact, generated by an algorithm and regrets that the analysis cannot be redone in time to account for this.

set, i.e., the weighted AUC (see Subsection 2.6.4). Then we perform the Friedman and post-hoc Nemenyi test (see Subsection 2.7).

3.5 Results

Table 3.2 presents the weighted AUC performances of each algorithm on the 24 real-world data sets. The best performance for each data set is in bold. The average ranks of the algorithms are shown at the bottom of the table. On these 24 real-world data sets, SVDD and LOF perform best, both with an average rank of 2.083. With an average rank

Table 3.2 The weighted AUC performance in percentages obtained by the Machine Learning and Knowledge Discovery outlier-selection algorithms on 24 real-world data sets. The best performance per data set is in bold. The corresponding average rank for each algorithm is reported in the bottom row.

Dataset	KNNDD	PWDD	SVDD	LOF	LOCI
Arrhytmia	56.50	50.00	61.76	**62.87**	56.70
Balance-scale	93.50	94.18	**95.80**	94.66	91.26
Biomed	**88.29**	72.99	88.20	78.61	85.04
Breast	96.68	72.24	97.75	96.81	**98.31**
Cancer wpbc	61.37	56.56	60.59	**62.57**	58.55
Colon Gene	**75.53**	50.00	70.18	73.47	42.08
Delft Pump	93.13	92.97	94.43	**94.93**	87.27
Diabetes	65.03	63.14	67.98	**68.24**	64.86
Ecoli	96.53	93.09	93.39	**96.74**	96.10
Glass Building	78.93	77.39	77.32	**81.47**	75.79
Heart	60.53	56.86	**62.05**	61.82	60.88
Hepatitis	64.72	58.73	60.89	**66.53**	62.05
Housing	**64.56**	61.66	64.24	62.86	63.75
Imports	80.24	80.09	**82.11**	80.49	74.24
Ionosphere	75.47	71.14	**81.38**	74.28	68.51
Iris	98.49	98.26	**99.20**	98.28	98.23
Liver	55.53	53.46	59.15	**59.57**	59.10
Sonar	74.22	74.45	75.77	**76.89**	66.71
Spectf	52.81	51.57	**78.92**	60.20	56.11
Survival	66.64	62.30	**68.12**	67.02	64.18
Vehicle	79.15	78.81	**81.37**	75.68	74.99
Vowel	99.49	**99.56**	99.28	97.89	95.22
Waveform	89.85	86.89	**90.36**	90.26	89.17
Wine	**89.43**	84.20	86.94	88.42	87.68
Average rank	2.625	4.292	2.083	2.083	3.917

of 2.625, KNNDD performs surprisingly well. LOCI and PWDD perform the worst, with average ranks of 3.917 and 4.292, respectively. Interestingly, SVDD seems to perform well on those data sets where LOF performs worse and vice versa. Apparently, both algorithms are complementary with respect to the nature of the data set at hand.

To see whether there is a significant difference between these average ranks, we calculate the Friedman statistic, $\chi_F^2 = 48.53$, which results in an F_F statistic of $F_F = 23.52$. With five algorithms and 24 data sets, F_F is distributed according to the F distribution with $5-1 = 4$ and $(5-1) \times (24-1) = 92$ degrees of freedom. The critical value of $F(4, 92)$ for $\alpha = .05$ is 2.471, so we must reject the null-hypothesis, which states that all algorithms have an equal performance (see the beginning of the chapter).

Subsequently, we continue with the Nemenyi test, for which the critical distance CD, for $\alpha = .05$, is 1.245. We identify two groups of algorithms. The performances of LOCI and PWDD are significantly worse than that of KNNDD, LOF, and SVDD.

Figure 3.3 graphically displays the result of the Nemenyi test in a critical difference diagram. Groups of algorithms that are not significantly different (at $p = .05$) are connected. The diagram reveals that, in terms of performances, the algorithms examined fall into two groups. The group of best-performing algorithms consists of SVDD, LOF, and KNNDD. The other cluster contains PWDD and LOCI.

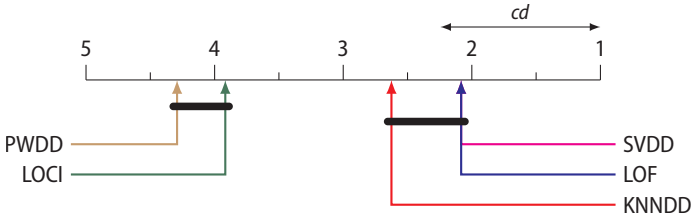


Figure 3.3 Comparison of all algorithms against each other with the Nemenyi test. Groups of algorithms that are not significantly different (at $p = .05$) are connected.

3.6 Discussion

We have evaluated five outlier-selection algorithms; three from the field of ML and two from the field of KDD on a variety of real-world data sets. The performances of the algorithms have been statistically compared using the Friedman and Nemenyi tests. From

the obtained experimental results we report three main observations. We describe each observation separately and provide possible reasons for each of them below.

3.6.1 Observation 1: Local density estimates outperform global density estimates

The first observation is that PWDD performs significantly worse than LOF. It may be explained by the fact that PWDD performs a global density estimate. Such an estimation becomes an obstacle when there exist large differences in the density, because data points in sparse clusters will be erroneously classified as outliers.

LOF and LOCI overcome this problem by performing an additional step. Instead of using the density estimate as an outlier score, they locally compare the density with the neighbourhood. This produces an estimate which is both relative and local, and enables LOF and LOCI to cope with different densities across different subspaces. For LOF, the local density estimate results in a better performance. For LOCI, however, this is not the case. Possible reasons are discussed in the second observation below.

3.6.2 Observation 2: LOF outperforms LOCI

The second observation is that LOCI is outperformed by LOF. This is unexpected because LOCI (1) considers local densities just like LOF and (2) performs a multi-scale analysis of the data set, which is often stated as an important improvement over LOF. We provide two possible reasons for the relative weak performance of LOCI.

The first possible reason is that LOF considers three consecutive neighbourhoods to compute the outlier score. LOCI, instead, considers two neighbourhoods, only. The three-fold density analysis of LOF is more profound than the two-fold analysis of LOCI and therefore LOF yields a better estimation of the data density.

The second possible reason for the observed results is that LOCI constructs a neighbourhood with a given radius, and not with a given number of data points. For small radii, the extended neighbourhood may contain one data point only, implying that there may be no deviation in the density and that outliers might be missed at a small scale. In contrast, LOF does not suffer from the limited number of data points because it constructs a neighbourhood with a given number of data points.

3.6.3 Observation 3: Domain-based and density-based algorithms are competitive

The third observation we make is that domain-based (SVDD) and density-based (LOF) algorithms are competitive in performance. To obtain adequate estimates, density-based algorithms require data sets to contain many data points, especially when they have a high dimensionality. This implies that in case of sparsely sampled data sets, density-based algorithms may fail to detect outliers (cf. Aggarwal and Yu, 2001). SVDD describes only the domain in the data point space (i.e., it defines a closed boundary around the normal class), and does not estimate the complete data density. It is therefore less sensitive to an inaccurate sampling and better able to deal with small sample sizes (Tax, 2001).

3.7 Chapter conclusions

In this chapter, we focussed on RQ1: *How should we evaluate and compare the performance of outlier-selection algorithms?* Previously, the KDD algorithms LOF and LOCI had not been compared with each other—and not with any other outlier-selection algorithm—in a statistically valid way. In the chapter, we have been able to evaluate and compare LOF and LOCI with three outlier-selection ML algorithms KNNDD, PWDD, and SVDD in a statistically valid way, by framing LOF and LOCI into the one-class classifier setting. By doing so, each outlier-selection algorithm is treated under the same circumstances.

From the experimental results we may conclude that the techniques presented in Chapter 2 allow for a statistical evaluation and comparison of outlier-selection techniques. When the outlier-selection algorithms come from different settings, i.e., the unsupervised outlier-selection setting (e.g., the field of KDD) and the one-class classification setting (e.g., the field of ML), they should be framed in a common setting. We achieved this by transforming LOF and LOCI into the one-class classification setting.

In addition to our main conclusion concerning RQ1, it is worth mentioning that we have determined that, for the data sets presented in Subsection 3.4.1, the algorithms with the best average performance are KNNDD, LOF, and SVDD. Our findings indicate that both fields have produced outlier-selection algorithms that are competitive and deserve treatment on equal footing. Such a finding may not have been possible without framing LOF and LOCI into the one-class classification setting.

Framing KDD algorithms into the one-class classification setting thus facilitates the comparison of algorithms across fields and may lead to novel algorithms that combine ideas of both fields. For instance, our results suggest that it may be worthwhile to develop outlier-selection algorithms that combine elements of domain-based and local density-based algorithms.

We identify two directions for future research. The first direction is to combine the best of both fields. For example, to develop an outlier-selection algorithm that (1) employs local densities (cf. LOF from the field of KDD) and (2) has a theoretical, and possibly probabilistic, foundation (cf. SVDD from the field of ML). As a direct result of these ideas, we introduce in Chapter 4 Stochastic Outlier Selection (SOS).

The second direction is to investigate the complementarity of LOF and SVDD with respect to the nature of the data set. The relative strengths of both algorithms appear to depend on the characteristics of the data set. Therefore, in Chapter 5 we investigate which data set characteristics determine the relative performance of an outlier-selection algorithm and a one-class classifier.

Chapter 4
Stochastic Outlier Selection

Contents:

In this chapter we attempt to answer RQ2: *Can an effective outlier-selection algorithm be devised that employs the concept of affinity?* To this end, we design and evaluate a novel, unsupervised algorithm for classifying data points as outliers, called Stochastic Outlier Selection (SOS). SOS uses affinity to compute for each data point an outlier probability. A data point is considered to be an outlier when the other data points have insufficient affinity with it. We evaluate SOS on real-world and synthetic data sets. The results obtained on these data sets show that SOS (1) has a significantly higher performance and (2) is more robust to data perturbations and varying densities than four related algorithms. From these results we may conclude that SOS is an effective algorithm for classifying data points as outliers.

Based on:

- J.H.M. Janssens, E.O. Postma, and H.J. van den Herik. Unsupervised outlier selection with pairwise affinities. In *SNN Symposium: Intelligent Machines*, Nijmegen, The Netherlands, Nov. 2010b.
- J.H.M. Janssens, F. Huszar, E.O. Postma, and H.J. van den Herik. Stochastic Outlier Selection. Technical Report TiCC TR 2012-001, Tilburg University, Tilburg, The Netherlands, 2012.

Outline:

4.1 An affinity-based approach to outlier selection. **4.2** The Stochastic Outlier Selection algorithm. **4.3** Qualitative evaluation of SOS and four related algorithms. **4.4** Experiments and results. **4.5** Discussion of the results. **4.6** Chapter conclusions.

In this chapter we propose a novel, unsupervised algorithm for classifying data points as outliers. The algorithm is called Stochastic Outlier Selection (SOS). It applies the concept of affinity to the problem of outlier selection. We explain and motivate the use of affinity in Section 4.1. How the SOS algorithm selects outliers is described in Section 4.2. Section 4.3 presents four related unsupervised outlier-selection algorithms: K-Nearest Neighbour Data Description (KNNDD), Local Outlier Factor (LOF), Local Correlation Integral (LOCI), and Least Squares Outlier Detection (LSOD). Using outlier-score plots, we illustrate and discuss the qualitative performances of SOS and the four algorithms. In Section 4.4, we evaluate all five algorithms on eighteen real-world data sets, and by the Neményi statistical test (cf. Subsection 2.7.2) we show that SOS performs significantly better. Moreover, seven synthetic data sets are used to gain insight into the behaviour of the algorithms. From our experiments we may conclude that SOS is more robust against data perturbations and varying densities than the other four algorithms. The results are discussed in Section 4.5. Finally, we give our conclusions in Section 4.6.

4.1 An affinity-based approach to outlier selection

We recall from Chapter 1 that the objective is to classify automatically those data points as outliers that are labelled as anomalous by the expert. Each algorithm approaches this objective in a different way. The SOS algorithm, which is the topic of the current chapter, selects outliers using an affinity-based approach.

The general idea of SOS is as follows. SOS employs the concept of affinity to quantify the relationship from one data point to another data point. Affinity is proportional to the similarity between two data points. So, a data point has little affinity with a dissimilar data point. A data point is selected as an outlier when all the other data points have insufficient affinity with it. The concept of affinity and the SOS algorithm are explained more precisely in the next section. First, in Subsection 4.1.1, we mention two problems to which affinity has been successfully applied. As a result of employing affinity, SOS computes outlier probabilities instead of outlier scores. The advantages of computing probabilities with respect to scores are discussed in Subsection 4.1.2.

4.1.1 Two successful applications of affinity

So far, affinity has been applied successfully to at least two other problems: (1) clustering and (2) dimensionality reduction. They will be discussed briefly below. To the best of our

knowledge, SOS is the first algorithm that applies the concept of affinity to the problem of outlier selection.

First, affinity has been successfully applied to the problem of clustering. The goal of clustering is to select a (given) number of data points that serve as the representatives of the clusters (i.e., cluster exemplars) in a data set. The clustering algorithm 'Affinity Propagation' by Frey and Dueck (2007) updates iteratively the affinity that a data point has to a potential cluster exemplar by passing messages. In other words, the clustering algorithm employs affinity to quantify the relationships among data points.

Second, affinity has been successfully applied to the problem of dimension reduction. The goal of dimension reduction is to map a high-dimensional data set onto a low-dimensional space (Roweis and Saul, 2000; Tenenbaum, de Silva, and Langford, 2000). The challenge is to preserve the structure of the data set as much as possible. Several algorithms address this challenge successfully by concentrating on preserving the local relationships using affinity (Hinton and Roweis, 2003; van der Maaten and Hinton, 2008; van der Maaten, 2009b). Again, as with clustering, affinity is used to quantify the relationships among data points.

Because of these two successful applications of affinity, we expect that affinity is also beneficial to outlier selection and in particular to the SOS algorithm. Here we note that affinity is calculated differently in the two applications. Our definition of affinity (which is presented in Subsection 4.2.2) is based on the definition found in the works concerning dimension reduction because that allows us to compute outlier probabilities.

4.1.2 Outlier probabilities instead of scores

Current outlier-selection algorithms typically compute unbounded outlier scores (see Gao and Tan, 2006). The scores computed by such algorithms differ widely in their scale, range, and meaning. Moreover, the scores may also differ from data set to data set for the same algorithm (Kriegel, Kröger, Schubert, and Zimek, 2009).

SOS computes outlier *probabilities*, i.e., the probability that a data point is an outlier. Because an outlier probability is a number between 0 and 1, both the minimum and the maximum value of outlier probabilities are consistent from data set to data set. We state three reasons why outlier probabilities are favourable to outlier scores. The first reason is that outlier probabilities are easier to interpret by an expert than outlier scores (Kriegel et al., 2009). The second reason is that outlier probabilities allow to select

an appropriate threshold for outlier selection (i.e., classification) using a Bayesian risk model (Gao and Tan, 2006). A Bayesian risk model takes into account the relative cost of misclassifications. Employing a Bayesian risk model is not possible with unbounded outlier scores. The third reason is that outlier probabilities provide a more robust approach for developing an ensemble outlier selection framework out of individual outlier-selection algorithms than unbounded outlier scores (Kriegel, Kröger, Schubert, and Zimek, 2011).

Whereas Gao and Tan (2006) and Kriegel et al. (2011) suggest converting the unbounded outlier scores from existing algorithms into calibrated probabilities, SOS computes the outlier probabilities directly from the data.

4.2 The Stochastic Outlier Selection algorithm

In this section we describe how SOS selects outliers. Stated more formally, cf. Definition 2.2 on page 23, we describe how the outlier-selection algorithm f_{SOS} maps data points to the classifications 'outlier' and 'inlier'.

The data that is used by SOS, i.e., the input data, the intermediate data, and the output data, can be represented as five matrices. Figure 4.1 shows the five matrices and summarises SOS as a series of matrix transformations. The numbers above the arrow denote the subsections that discuss the matrix transformations. Next to each matrix we find a colour bar that maps a range of values to a range of colours. In the case of matrix $\mathbf{X}$, for example, 0 is mapped to white and 8 is mapped to dark blue. As such, Figure 4.1 may serve as an overview reference throughout our description of SOS.

Subsections 4.2.1—4.2.6 are concerned with the outlier scoring part ϕ_{SOS} that maps data points to outlier probabilities. We briefly mention the purpose of each subsection. In Subsection 4.2.1, we discuss the input data and the dissimilarity between data points (i.e., input matrix $\mathbf{X}$ and dissimilarity matrix $\mathbf{D}$). In Subsection 4.2.2, we explain how dissimilarities are transformed into affinities (i.e., affinity matrix $\mathbf{A}$). In Subsection 4.2.3 we continue our description by using graph theory, and generate stochastic neighbour graphs that are based on binding probabilities (i.e., binding matrix $\mathbf{B}$). We present three ways of computing the outlier probabilities (i.e., output matrix $\mathbf{\Phi}$). In Subsection 4.2.4 we show that by sampling stochastic neighbour graphs, we can estimate outlier probabilities. Through marginalisation we can compute exactly the outlier probabilities as is described

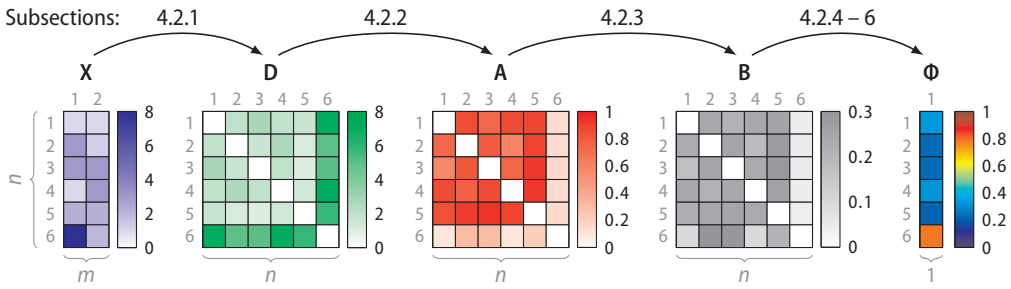


Figure 4.1 From input to output in five matrices: (1) the input matrix **X** containing the feature values of the data points, (2) the dissimilarity matrix **D**, (3) the affinity matrix **A**, (4) the binding probability matrix **B**, and (5) the output matrix **Φ** containing the outlier probabilities. The transformations from one matrix to the next matrix are explained in the subsections stated above the arrows.

in Subsection 4.2.5. With the help of probability theory, we observe in Subsection 4.2.6 that the outlier probabilities can also be computed in closed form.

In Subsection 4.2.7, the outlier scoring part ϕ_{SOS} is transformed into the outlier-selection algorithm f_{SOS}, so that SOS can classify data points as outliers. Finally, in Subsection 4.2.8, we explain in detail the concept of perplexity that allows SOS to create soft neighbourhoods.

4.2.1 Input to the algorithm

SOS is an unsupervised outlier-selection algorithm. Therefore, SOS requires as input an unlabelled data set $\mathcal{D}$, only. We recall from Chapter 1 that an unlabelled data set means that the labels 'anomaly' and 'normality' are unavailable. Therefore, SOS does not know whether a domain expert considers the real-world observations (that correspond to the data points) to be anomalous or normal. Nevertheless, SOS is able to classify the data points in the data set either as 'outlier' or 'inlier'.

The number of data points in the unlabelled data set $\mathcal{D}$ is denoted by n. In our description of SOS, each data point is represented by an m-dimensional, real-valued, feature vector $\mathbf{x} = [x_1, \ldots, x_m] \in \mathbb{R}^m$. As such, a data point can be regarded as a point in an m-dimensional Euclidean space. The data set $\mathcal{D}$ is represented by a matrix $\mathbf{X}$ of size $n \times m$, i.e., number of data points × number of features. The vector $\mathbf{x}_i$ denotes the i^{th} row of the matrix $\mathbf{X}$. In our description of SOS we do not distinguish between the data set $\mathcal{D}$ and the corresponding matrix $\mathbf{X}$, and often refer to the data set by $\mathbf{X}$ for readability.

Figure 4.2 shows the example data set that we use throughout our description of SOS. The data set contains six data points (○), and each data point has two features. The corresponding two-dimensional points are plotted on the left side of Figure 4.2. On the right side of the figure, the same data set is represented by matrix **X**. We recall that the colour of the cells in the matrix **X** correspond to the feature values of the data points.

The features of the data points are used to measure the *dissimilarity* between pairs of data points. (Dissimilarity forms the basis for affinity, see the next subsection.) The dissimilarity between data point $\mathbf{x}_i$ and data point $\mathbf{x}_j$ is a non-negative scalar that is computed by a dissimilarity measure d. In our description and in our experiments we employ the Euclidean distance as the dissimilarity measure between pairs of data points. Let

$$d_{ij} = \sqrt{\sum_{k=1}^{m} \left(x_{jk} - x_{ik}\right)^2}, \tag{4.1}$$

where x_{ik} denotes the k^{th} feature value of i^{th} data point, i.e., cell i, k of matrix **X**. From Equation 4.1 it follows (1) that our employed dissimilarity measure is symmetric, i.e., $d_{ij} \equiv d_{ji}$, and (2) that the dissimilarity between a data point and itself is zero, i.e., $d_{ii} = 0$. The data points on the left side of Figure 4.2 are connected by green lines with varying brightnesses. Both the length and the brightnesses of these green lines illustrate the dissimilarity between data point $\mathbf{x}_2$ and the other five data points. The right side of Figure 4.2 shows the dissimilarity matrix **D** that is obtained by applying Equation 4.1 to each pair of data points in the matrix **X**. (The bold, upright letter '**D**' should not be confused with the calligraphic letter '$\mathcal{D}$' that denotes a data set.) The

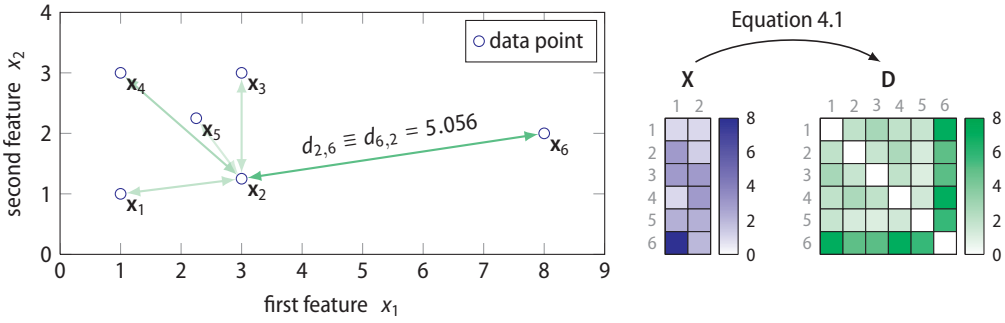


Figure 4.2 The example data set used for our SOS description. Each data point has two features and is a point in two-dimensional Euclidean space. The dissimilarity $d_{2,6}$ (and $d_{6,2}$) is the Euclidean distance between data points $\mathbf{x}_2$ and $\mathbf{x}_6$ (see Equation 4.1).

brightnesses of the green lines are equal to the brightnesses of the cells in the second row of **D**. In fact, because the Euclidean distance d is symmetric, the resulting dissimilarity matrix **D** is symmetric, meaning that the rows are equal to the columns. Therefore, the brightnesses of the green lines are also equal to the brightnesses of the cells of the second column.

In the next subsection, dissimilarities are used to compute *affinities* between data points. In other words, the dissimilarity matrix **D** is transformed into the affinity matrix **A**.

4.2.2 Transforming dissimilarity into affinity

As mentioned in Section 4.1, we employ affinity in order to quantify the relationship from one data point to another data point. Our definition of affinity is based on the definitions used for the problem of dimension reduction (Hinton and Roweis, 2003; Goldberger, Roweis, Hinton, and Salakhutdinov, 2005; van der Maaten and Hinton, 2008).

Definition 4.1 (Affinity). *Let d_{ij} denote the dissimilarity that data point $\mathbf{x}_j$ has to data point $\mathbf{x}_i$. Then the affinity that data point $\mathbf{x}_i$ has with data point $\mathbf{x}_j$ is given by*

$$a_{ij} = \begin{cases} \exp\left(-d_{ij}^2 \,/\, 2\sigma_i^2\right) & \text{if } i \neq j \\ 0 & \text{if } i = j\,, \end{cases} \tag{4.2}$$

where σ_i^2, known as the variance, is a scalar associated with data point $\mathbf{x}_i$.

We note that (1) the affinity that data point $\mathbf{x}_i$ has with data point $\mathbf{x}_j$ decays Gaussian-like with respect to the dissimilarity d_{ij}, and (2) a data point has no affinity with itself, i.e., $a_{ii} = 0$ (the justification is discussed below).

In words, Equation 4.2 states that the affinity that data point $\mathbf{x}_i$ has with data point $\mathbf{x}_j$ is proportional to the probability density at $\mathbf{x}_j$ under a Gaussian distribution that has mean $\mathbf{x}_i$ and variance σ_i^2. A graphical comparison of three variance values for Equation 4.2 is provided in Figure 4.3. A lower variance causes the affinity to decay faster. The higher a variance is, the less affinity is influenced by the dissimilarity. As an extreme example, an infinitely high variance yields an affinity of 1, no matter how high the dissimilarity is (because $e^0 = 1$). Stated differently, the Gaussian distribution becomes a uniform distribution.

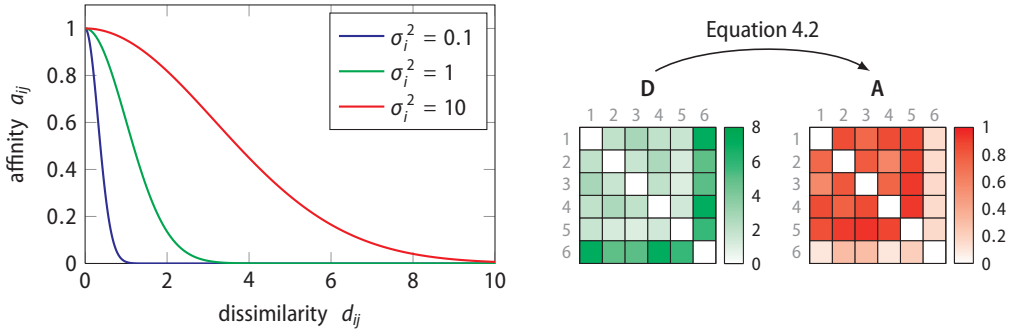


Figure 4.3 From dissimilarity to affinity. **Left:** Graphs of the affinity a_{ij} with respect to the dissimilarity d_{ij} as defined by Equation 4.2, for three values of the variance σ_i^2. **Right:** The affinity matrix **A** is obtained by applying Equation 4.2 to each cell in the dissimilarity matrix **D**.

SOS has one parameter only, which is called the perplexity parameter and is denoted by h. The perplexity parameter h can be compared with the parameter k as in k-nearest neighbours, with two important differences. First, because affinity decays smoothly, 'being a neighbour' is not a binary property, but a smooth property. In fact, in the next subsection we formalise 'being a neighbour' into a probabilistic property using Stochastic Neighbour Graphs. Second, unlike the parameter k, perplexity is not restricted to be an integer, but can be any real number between 1 and $n - 1$. Stated differently, a data point can have a minimum of 1 and a maximum of $n - 1$ effective neighbours. Perplexity may therefore be interpreted as a smooth measure for the effective number of neighbours. Because a data point has no affinity with itself, i.e., $a_{ii} = 0$, it is never its own neighbour, which implies (as we will see later) that only *other* data points have influence on its outlier probability.

The value of each variance σ_i^2 is determined using an adaptive approach such that each data point has the same number of effective neighbours, i.e., h. The adaptive approach yields a different variance for each data point, causing the affinity to be asymmetric (Hinton and Roweis, 2003). To be precise, the affinities a_{ij} and a_{ji} are equal only when (1) the dissimilarities d_{ij} and d_{ji} are equal, which is always the case with the Euclidean distance but need not be the case with other dissimilarity measures, and (2) the variances σ_i^2 and σ_j^2 are equal, which is rarely the case unless the two data points are equally dissimilar to their own neighbours. As a counterexample to asymmetric affinity we mention the dimensionality reduction algorithm by van der Maaten and Hinton (2008), which symmetrises purposefully the affinities between data points (i.e., $a_{ij} = \frac{1}{2}a_{ij} + \frac{1}{2}a_{ji}$), such that dissimilar data points are not isolated but joined with one of the clusters

in the low-dimensional mapping. The purpose of SOS, however, is to classify these dissimilar data points as outliers, and, therefore, does not symmetrise the affinities. We elaborate upon the details of assigning the variances with the adaptive approach in Subsection 4.2.8.

Figure 4.4 shows the variances for the data points in the example data set. The radii of the six circles correspond to the variances for the six data points (and please note that the circles do not represent a boundary). In all figures and calculations concerning the example data set, the perplexity h is set to 4.5, with Figure 4.4 being the only exception to this. In Figure 4.4, the perplexity h is set to 3.5 because otherwise, the radii of the six circles would be to be too large to create an illustrative figure (see Figure 4.13 for the variances that correspond to all possible settings of the perplexity parameter). Figure 4.4 shows, for example, that for data point $\mathbf{x}_6$ to have 3.5 effective neighbours, the variance σ_6^2 must be set higher than the other variances.

Using the dissimilarity matrix $\mathbf{D}$ and Equation 4.2, we compute the affinity that each data point has to every other data point, resulting in the affinity matrix $\mathbf{A}$ (illustrated in the right side of Figure 4.3). The affinity matrix $\mathbf{A}$ is in two aspects similar to the dissimilarity

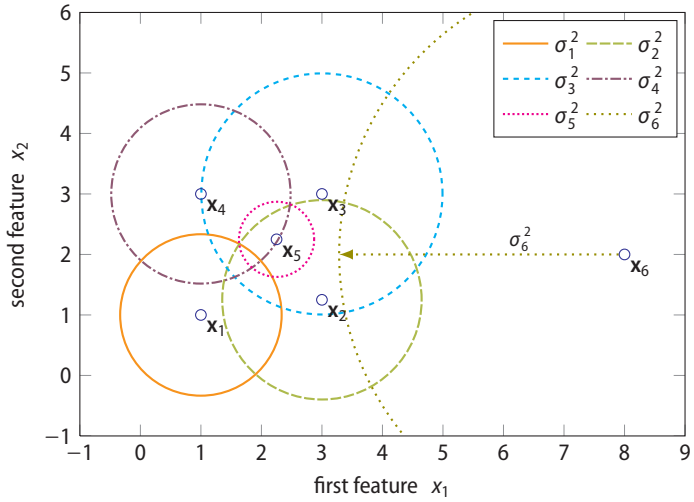


Figure 4.4 The radii of the circles correspond to the variance for the data points. The variance σ_i^2 adapts to the density of the data and is determined for each data point $\mathbf{x}_i$ separately, such that each data point has the same number of effective neighbours, i.e., perplexity. In this figure the perplexity is set to 3.5. The variance influences the amount of affinity that a data point has to the other data points (see Figure 4.3).

matrix $\mathbf{D}$, (1) the size is $n \times n$ and (2) the diagonal is 0, because data points have no affinity with themselves. However, unlike $\mathbf{D}$, $\mathbf{A}$ is not symmetric, because the affinity between two data points is not symmetric. The i^{th} row of the affinity matrix, denoted by $\mathbf{a}_i$, is called the *affinity distribution* of data point $\mathbf{x}_i$. Let

$$\mathbf{a}_i = \left[a_{i,1}, \ldots, a_{i,n}\right] , \tag{4.3}$$

where a_{ii} is 0. We note that the affinity distribution is not a probability distribution because it does not sum to 1. In the next subsection we continue with the affinity matrix $\mathbf{A}$ to generate a Stochastic Neighbour Graph.

4.2.3 Stochastic neighbour graphs based on affinities

In the remainder of our description of SOS, we employ graph theory, because (1) it provides a solid mathematical framework to perform calculations with affinities, and (2) it allows us to derive outlier probabilities. To model explicitly the data points and their relationships (i.e., affinities) using vertices and directed edges, we generate a Stochastic Neighbour Graph (SNG). The set of vertices $\mathcal{V}$ is associated with the data set $\mathbf{X}$. So, each vertex v_i corresponds to data point $\mathbf{x}_i$. Generating the directed edges between the vertices depends on *binding probabilities*. Therefore, we first introduce (A) the concept of binding probabilities and subsequently define (B) the generative process for an SNG. Finally, we introduce (C) the binary property of being an outlier given one SNG.

A Binding probabilities

The binding probability b_{ij} is the probability that vertex v_i binds to vertex v_j, i.e., the probability of generating a directed edge from v_i to v_j. We denote a directed edge from v_i to v_j by '$i \rightarrow j$'. The binding probability b_{ij} is proportional (denoted by $\propto$) to the affinity that data point $\mathbf{x}_i$ has with data point $\mathbf{x}_j$.

$$b_{ij} \equiv p\left(i \rightarrow j \in \mathcal{E}_G\right) \propto a_{ij} , \tag{4.4}$$

which is equal to the affinity a_{ij} normalised, such that $\sum_{k=1}^{n} b_{ik}$ sums to 1.

$$b_{ij} = \frac{a_{ij}}{\sum_{k=1}^{n} a_{ik}} . \tag{4.5}$$

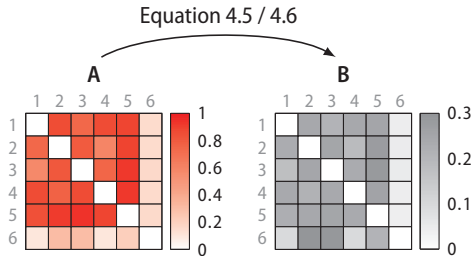


Figure 4.5 The binding matrix **B** is obtained by normalising each row in the affinity matrix **A**.

We note that b_{ii} is always 0, because a_{ii} is always 0 (see Equation 4.2). By applying Equation 4.5 to every cell in the affinity matrix **A**, we obtain the binding matrix **B** (see Figure 4.5). The binding probabilities for one vertex v_i form together a *binding distribution*, which is a discrete probability distribution

$$\mathbf{b}_i = \left[b_{i,1}, \ldots, b_{i,n}\right]. \tag{4.6}$$

Similar to the affinity distribution $\mathbf{a}_i$ from Equation 4.3, the binding distribution $\mathbf{b}_i$ is the i^{th} row of the binding matrix **B**.

We can illustrate the binding probabilities by representing the data set as a graph, where each data point $\mathbf{x}_i$ is associated with a vertex v_i. Because each vertex has *some* probability of binding to any other vertex, the graph is fully connected. Figure 4.6 shows this graph four times (these are not yet Stochastic Neighbour Graphs, those are shown later, in Figure 4.7). The brightness of each directed edge $i \rightarrow j$ is determined by the binding probability b_{ij}, as can be seen on the right side of Figure 4.6, where the binding matrix **B** is depicted four times. The same graph is shown four times, i.e., graph (a) to graph (d), where each graph illustrates a different aspect. In graph (a), five directed edges are coloured orange. The brightness of an orange edge is associated with the probability that vertex v_1 binds to another vertex. The corresponding binding distribution $\mathbf{b}_1$ sums to 1. The associated row in the binding matrix **B**, i.e., the first row, is also coloured orange. From the graph we can see, for example, that the probability that v_1 binds to v_6 is relatively low ($\approx .04$). If we reconsider Figure 4.1 on page 63, then we can see that the low binding probability $b_{1,6}$ is due to the low affinity $a_{1,6}$, which is due to the high dissimilarity $d_{1,6}$, which is due mostly to the large difference in the values of the first feature of data points $\mathbf{x}_1$ and $\mathbf{x}_6$. In graph (b), the five edges from vertex v_6 are coloured orange. We can see that the binding distribution $\mathbf{b}_6$ is distributed more evenly than $\mathbf{b}_1$, because from the perspective

of data point $\mathbf{x}_6$, the other five data points are roughly equally distant. In both graph (c) and graph (d), five edges are coloured cyan, to illustrate the probability that *other* vertices bind to vertex v_1 and vertex v_6, respectively. These binding probabilities do not necessarily sum to 1. From graph (d), (and similarly from the sixth column in the binding matrix **B**), we can see that all vertices have a low probability to bind to vertex v_6. Please note that these four graphs only serve to illustrate the concept of binding probabilities, and will not be used to compute outlier probabilities.

B Generating a stochastic neighbour graph

After the introduction of the binding probabilities, we define formally the Stochastic Neighbour Graph (SNG) and discuss the generative process for an SNG. We do so by two assumptions and four subsequent consequences. After that we describe the generative process of the set of directed edges $\mathcal{E}_G$.

Definition 4.2 (Stochastic neighbour graph). *A Stochastic Neighbour Graph (SNG) is a graph $G = (\mathcal{V}, \mathcal{E}_G)$ with a set of vertices $\mathcal{V}$ and a set of directed edges $\mathcal{E}_G$. Let matrix* **X** *be a data set containing n data points. For each data point $\mathbf{x}_i \in \mathbf{X}$, a vertex v_i is added to $\mathcal{V}$.*

Let d be the dissimilarity measure (e.g., the Euclidean distance as stated by Equation 4.1). Let h be the perplexity parameter, where $h \in [1, n-1]$. Then matrix **D** *is the dissimilarity matrix of size $n \times n$ that is obtained by applying dissimilarity measure d on all pairs of data points in data set* **X**. *Then $\{\sigma_1^2, \ldots, \sigma_n^2\}$ are n variances, whose values are determined in Subsection 4.2.8 using perplexity h. Then matrix* **A** *is the affinity matrix of size $n \times n$ as obtained by applying Equation 4.2 with the n variances to each cell of the dissimilarity matrix* **D**. *Then matrix* **B** *is the binding matrix of size $n \times n$ as obtained by applying Equation 4.5 to each cell of the affinity matrix* **A**.

The set of directed edges $\mathcal{E}_G$ is generated by the following stochastic binding procedure. Let $\mathbf{b}_i$ be the binding distribution of vertex v_i, which is the i^{th} row of the binding matrix **B**. *Let each vertex $v_i \in \mathcal{V}$ bind independently to one other vertex v_j, where the index j is an integer sampled from the binding distribution $\mathbf{b}_i$. Let $i \to j$ denote a directed edge from vertex v_i to vertex v_j. The directed edge $i \to j$ is added to $\mathcal{E}_G$ if vertex v_i binds to vertex v_j.*

Below we discuss seven properties of an SNG that are implied by Definition 4.2 and the two generative procedures. We illustrate our description by Figure 4.7. Figure 4.7 shows three possible SNGs for the example data set with the binding matrix **B** that is illustrated in Figure 4.5. (The right side of the figure is introduced below.)

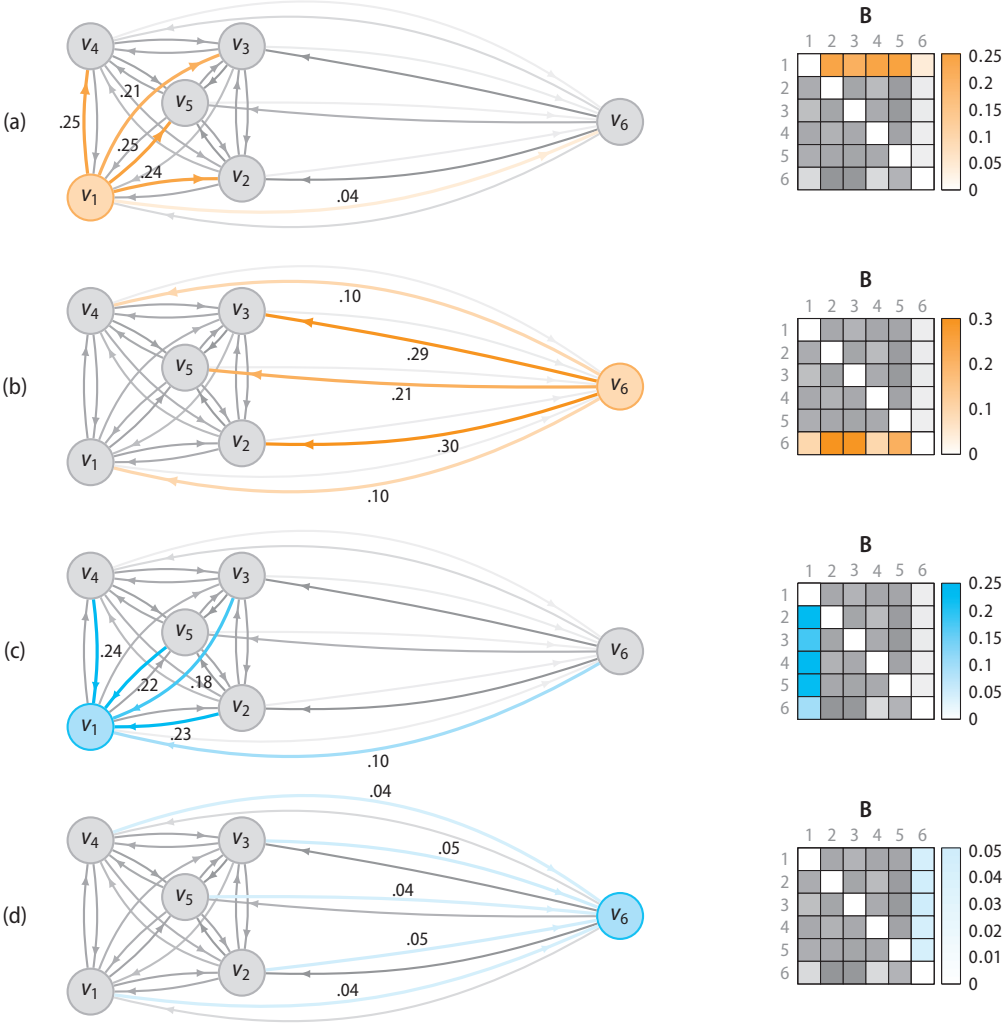


Figure 4.6 Illustration of binding probabilities. **Left:** Each vertex v_i in the graph is associated with a binding probability distribution $\mathbf{b}_i$. **Right:** The binding matrix **B**. A darker cell in the matrix corresponds to a higher binding probability.

First, an SNG always has n directed edges because there are n vertices that each connect to one vertex. Second, although the vertices of an SNG are fixed given a data set $\mathbf{X}$, each generated SNG may be different because the directed edges are generated using a stochastic binding procedure. Third, each binding distribution $\mathbf{b}_i$ is based on the affinity distribution $\mathbf{a}_i$, so an SNG reflects the relationship between data points. Fourth, an SNG has no self-loops, i.e., no vertex binds to itself because b_{ii} is 0, and as a consequence the index j that is sampled from the binding distribution will never be equal to i. Fifth, the vertices bind independently, which means that vertices do not influence each other's binding process. Sixth, each vertex v_i has an out-degree of 1, denoted by $\deg_G^+(v_i) = 1$, because each vertex binds to one other vertex. Seventh, because the binding process is stochastic, more than one vertex may bind to the same vertex. For example, both vertices v_1 and v_2 may bind to vertex v_3. Vertex v_3 now has an in-degree of 2, denoted by $\deg_G^-(v_3) = 2$. This implies that there is (at least) one vertex in the graph that no vertex binds to and thus has an in-degree of 0. In Figure 4.7 we can see, for instance, that in both graphs G_a and G_b, vertex v_6 has an in-degree of 0. In graph G_c, vertex v_6 has an in-degree of 3. It is possible (albeit improbable) to generate this graph, because each vertex (except vertex v_6 itself) has some probability of binding to vertex v_6. In the next subsection we elaborate on the probability of constructing a particular SNG.

C Being an outlier given one SNG

If there is a directed edge from v_i to v_j, i.e., $i \to j \in \mathcal{E}_G$, then we say that $\mathbf{x}_j$ is a neighbour of $\mathbf{x}_i$. In other words, data point $\mathbf{x}_i$ chooses data point $\mathbf{x}_j$ as a neighbour. If there is no directed edge from v_j *back* to v_i, i.e., $j \to i \notin \mathcal{E}_G$, then $\mathbf{x}_j$ is not a neighbour of $\mathbf{x}_i$.

We define that a data point is an outlier in graph G, when the data point has no neighbours in graph G. In graphical terms, a data point $\mathbf{x}_i$ belongs to the outlier class $\mathcal{C}_\text{O}$ if its corresponding vertex v_i has no inbound edges in graph G, i.e., its in-degree $\deg_G^-$ is zero:

$$\mathcal{C}_\text{O} \,|\, G = \left\{ \mathbf{x}_i \in \mathbf{X} \;\middle|\; \deg_G^-(v_i) = 0 \right\} . \tag{4.7}$$

In words, there is no vertex v_j that binds to vertex v_i,

$$\mathcal{C}_\text{O} \,|\, G = \left\{ \mathbf{x}_i \in \mathbf{X} \;\middle|\; \nexists v_j \in \mathcal{V} : j \to i \in \mathcal{E}_G \right\} , \tag{4.8}$$

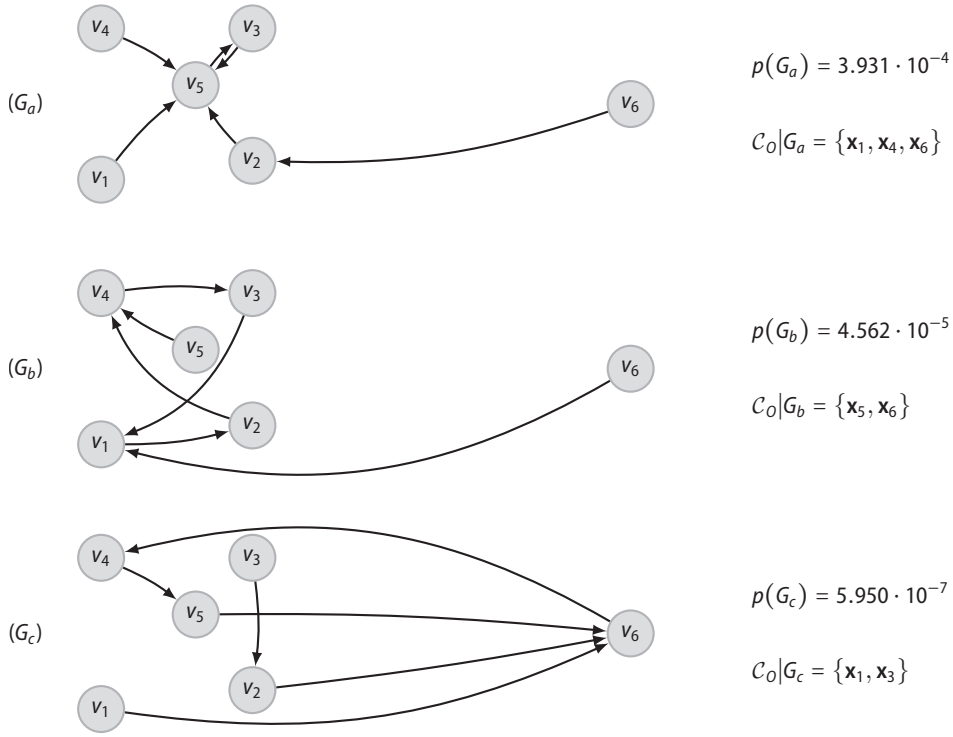


Figure 4.7 Three stochastic neighbour graphs (SNGs) generated with perplexity h set to 4.5. **Left:** The three SNGs G_a, G_b, and G_c are sampled from the discrete probability distribution $P(\mathcal{G})$. In Figure 4.8 it is indicated where the three SNGs are in the set of all graphs $\mathcal{G}$. **Right:** The probability $p(G_i)$ is the probability that graph G_i is generated (see Equation 4.10). The outlier class $\mathcal{C}_O|G_i$ contains the data points that are outliers given graph G_i (see Equation 4.7).

or, similarly, all vertices v_j do not bind to vertex v_i,

$$\mathcal{C}_{\mathrm{O}} \,|\, G = \left\{ \mathbf{x}_i \in \mathbf{X} \;\middle|\; \forall v_j \in \mathcal{V} : j \to i \notin \mathcal{E}_G \right\} . \qquad \boxed{4.9}$$

The right side of Figure 4.7, lists, for the three SNGs, the data points that belong to the outlier class $\mathcal{C}_{\mathrm{O}} \,|\, G$.

At this moment, being an outlier is a *binary* property; given one particular SNG G, a data point is either a member of the outlier class or not. The next subsection explains that by generating many SNGs, we can estimate the outlier *probability* .

4.2.4 Estimating outlier probabilities through sampling

In the previous subsection the outlier class $\mathcal{C}_O$ is established deterministically given one particular SNG G. But because G itself is generated stochastically (using the binding probabilities), the outlier class $\mathcal{C}_O$ becomes a stochastic subset of the data set. Therefore, data points are currently randomly selected as outlier, which is an obstacle.

We alleviate this obstacle by taking not one, but all possible SNGs into account. For a data set containing n data points, there are $(n-1)^n$ binding combinations possible, because each of the n vertices binds independently to one of $n-1$ vertices. We denote the set of all $(n-1)^n$ possible graphs by $\mathcal{G}$. For our example data set, the set $\mathcal{G}$ contains $5^6 = 15{,}625$ SNGs.

Because the binding probabilities are not distributed uniformly, certain edges are more probable to be generated than other edges. For instance, in our example data set, the edge $2 \to 1$ is more probable than the edge $2 \to 6$. As a consequence, certain SNGs are more probable to be generated than others. Since the vertices $\mathcal{V}$ are constant, the probability of generating a certain SNG G depends only on the binding probabilities.

$$p(G) = \prod_{i \to j \in \mathcal{E}_G} b_{ij} \,. \tag{4.10}$$

The sampling probabilities of the three SNGs are listed on the right side of Figure 4.7.

The set of all graphs $\mathcal{G}$ is thus associated with a discrete probability distribution $P(\mathcal{G})$. To sample an SNG from the probability distribution, denoted by $G \sim P(\mathcal{G})$, means to generate an SNG. Figure 4.8 shows the probability mass and the cumulative probability mass for $\mathcal{G}$, for three values of the perplexity h. A lower perplexity (e.g., $h = 4.0$, blue line) yields less uniform binding distributions, and consequently leads to more variation in the probabilities by which SNGs are sampled. Figure 4.8 is annotated by three arrows pointing to the red line. These three arrows indicate the 'positions' of the three SNGs of Figure 4.7 in $P(\mathcal{G})$. We can see that G_a is the most probable SNG to be generated, because each data point chooses as its neighbour the data point to which it has the most affinity. We can also see that G_c is one of the least probable SNGs.

We are now ready to use a sampling procedure to estimate the probability that a data point belongs to the outlier class. Given a number of sampled SNGs G, we compute the

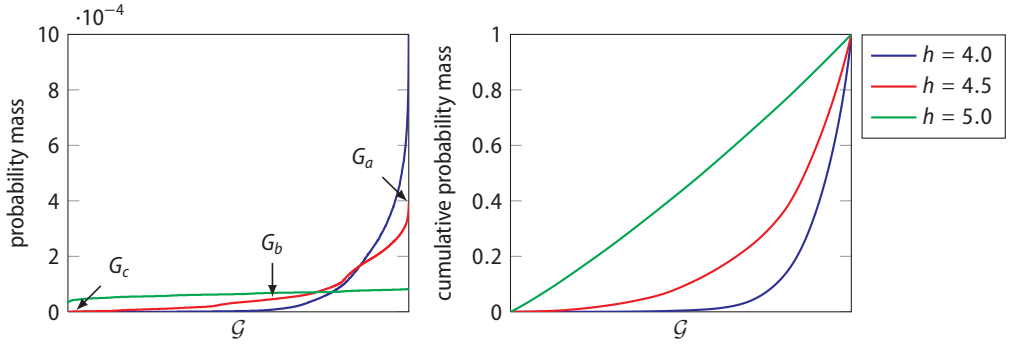


Figure 4.8 Discrete probability distribution for the set of all SNGs. **Left:** The probability mass for the discrete probability distribution for the set of all Stochastic Neighbour Graphs (SNGs) $\mathcal{G}$ for the example data set, for three values of the perplexity h. For this plot, the graphs in the set are ordered ascendingly by their probability of being generated. The annotations G_a, G_b, and G_c correspond to the probability of the three SNGs shown in Figure 4.7. **Right:** The cumulative probability mass for $\mathcal{G}$. The high sensitivity to the perplexity is due to the small number of data points in the example data set.

relative frequency that the data point belongs to the outlier class. As the number of samples S approaches infinity, the relative frequency converges to the outlier probability.

$$p(\mathbf{x}_i \in \mathcal{C}_O) = \lim_{S\to\infty} \frac{1}{S} \sum_{s=1}^{S} \mathbb{1}\left\{\mathbf{x}_i \in \mathcal{C}_O \,|\, G^{(s)}\right\} \quad , \quad G^{(s)} \sim P(\mathcal{G}) \ , \tag{4.11}$$

where $\mathbb{1}\{\cdot\}$ is an indicator random variable that has a value of 1 if data point $\mathbf{x}_i$ belongs to the outlier class $\mathcal{C}_O$ given graph G, and 0 otherwise.

The sampling procedure implies that if a particular data point belongs to the outlier class for all possible graphs, then its outlier probability is 1 (since the sum of the probabilities of all possible graphs is 1, i.e., $\sum_{G\in\mathcal{G}} p(G) = 1$). Similarly, if a data point is a member of the outlier class for 30% of the sampled graphs, then its outlier probability is 0.3. We note that the sampling procedure leads to an outlier probability that is a probability from a Frequentist point of view (Bayarri and Berger, 2004), and that it is not obtained by normalising an unbounded outlier score as in, for example, Gao and Tan (2006) and Kriegel et al. (2011).

The outlier probabilities during the first 1,000,000 iterations of one run of the sampling procedure are plotted in Figure 4.9. The figure reveals that when the perplexity is set to 4.5, the estimated outlier probabilities of data points $\mathbf{x}_1$ to $\mathbf{x}_6$ of the example data set, after 1,000,000 samples, are: 0.336, 0.234, 0.237, 0.323, 0.224, and 0.788, respectively. Because sampling SNGs is a stochastic process, each run produces in the beginning (say, the first

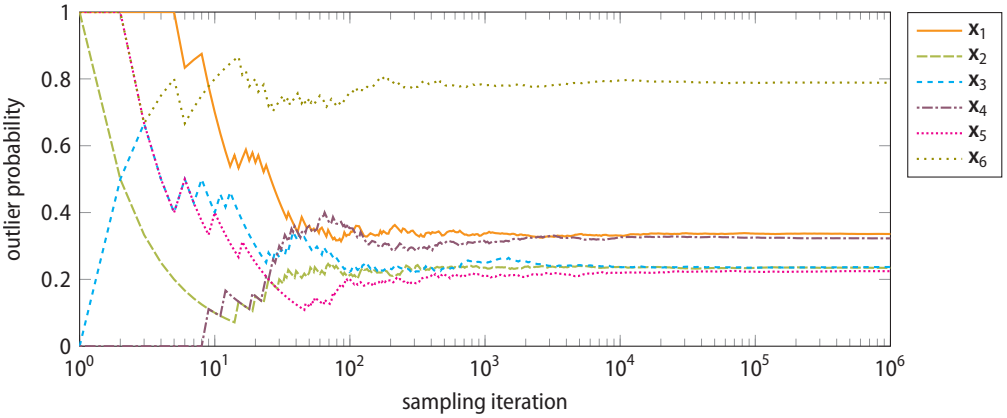


Figure 4.9 Convergence of the outlier probabilities by repeatedly sampling SNGs. The plot shows the first 1,000,000 sampling iterations of one run.

10 iterations) different outlier probabilities. Eventually (say, after 10,000 iterations), all runs produce outlier probabilities that converge to the same values.

4.2.5 Computing outlier probabilities through marginalisation

The relative frequency computed by Equation 4.11 only converges to the outlier probability when the number of samples approaches infinity. It turns out that we can compute the outlier probability exactly, by enumerating once over all possible SNGs. If we state the enumeration procedure in more technical terms, then we say that we compute the marginal probability of any particular data point being an outlier, by marginalising out the stochastic graph G. Because one SNG is more probable than the other (due to the binding probabilities), it is important to take this into account as well.

$$p(\mathbf{x}_i \in \mathcal{C}_O) = \sum_{G \in \mathcal{G}} \mathbb{1}\{\mathbf{x}_i \in \mathcal{C}_O \mid G\} \cdot p(G) \tag{4.12}$$

$$= \sum_{G \in \mathcal{G}} \mathbb{1}\{\mathbf{x}_i \in \mathcal{C}_O \mid G\} \cdot \prod_{q \to r \in \mathcal{E}_G} b_{qr} \,. \tag{4.13}$$

The exact outlier probabilities for the data points in the example data set, as computed by marginalisation, are: 0.335, 0.235, 0.237, 0.323, 0.224, and 0.788, respectively.

So, instead of *estimating* the outlier probabilities by sampling, say, 1,000,000 SNGs, we can *compute exactly* the outlier probabilities by marginalising over 15,625 SNGs. However,

such a gain holds only for small data sets, such as our example data set. To illustrate how quickly the size of $\mathcal{G}$, i.e., $|\mathcal{G}|$, grows with respect to n, consider $n = 5$, $n = 10$, and $n = 100$. These data set sizes correspond to $|\mathcal{G}| = 1024$, $|\mathcal{G}| = 3.5 \cdot 10^9$, and $|\mathcal{G}| = 3.7 \cdot 10^{199}$, respectively. So, even small data sets lead to a combinatorial explosion, making Equation 4.12 intractable to compute. In the next subsection we present a way to avoid this problem.

4.2.6 Computing outlier probabilities in closed form

Because each vertex binds to exactly one other vertex, the outlier probability can be computed in closed form, without actually enumerating all the SNGs in $\mathcal{G}$. Here we note that if a vertex would have been allowed to bind to *multiple* vertices, the outlier probability could not be computed in closed form.

We observe that the probability that data point $\mathbf{x}_i$ belongs to the outlier class, is equal to the probability that its in-degree is zero, i.e., the probability that none of the other vertices bind to vertex v_i. Without using graph-theoretical terms, the outlier probability of data point $\mathbf{x}_i$ can be reformulated as the joint probability that data point $\mathbf{x}_i$ is never chosen as a neighbour by the other data points. As a consequence, we can compute the outlier probabilities directly, without generating any SNG.

Theorem 4.1 (Outlier probability). *Let* $\mathbf{X}$ *be a data set containing n data points. Let* $\mathcal{C}_O$ *denote the outlier class. If* a_{ij} *is the affinity that data point* $\mathbf{x}_i$ *has with data point* $\mathbf{x}_j$*, then by Equation 4.5 we have that* b_{ij} *is the normalised affinity, i.e., the probability that* $\mathbf{x}_i$ *chooses* $\mathbf{x}_j$ *as its neighbour. Then the probability that data point* $\mathbf{x}_i$ *belongs to the outlier class is given by*

$$p(\mathbf{x}_i \in \mathcal{C}_O) = \prod_{j \neq i} \left(1 - b_{ji}\right) . \tag{4.14}$$

Proof. We recall from Equation 4.7 that a data point $\mathbf{x}_i$ belongs to the set of outliers $\mathcal{C}_O$, given one SNG graph G, when the corresponding vertex v_i has an in-degree of zero:

$$\mathbf{x}_i \in \mathcal{C}_O \,|\, G \iff \deg_G^-(v_i) = 0 . \tag{4.15}$$

We aim to compute the marginal probability that a data point is an outlier, given all SNGs. By associating the right-hand side of Equation 4.15 with an indicator random value $\mathbb{1}\{\cdot\}$, which has a value of 1 if v_i has an in-degree of zero and has a value of 0 otherwise, we may rewrite the probability as the expected value of the indicator random variable (cf. Cormen et al., 2009, p. 118),

$$p(\mathbf{x}_i \in \mathcal{C}_O) = \mathbb{E}_{\mathcal{G}}\left[\mathbb{1}\left\{\deg_G^-(v_i) = 0\right\}\right], \tag{4.16}$$

where the subscripted $\mathcal{G}$ indicates the sample space. By rewriting Equation 4.9, which states that the in-degree of v_i is zero if none of the vertices bind to v_i, as a product, we obtain,

$$p(\mathbf{x}_i \in \mathcal{C}_O) = \mathbb{E}_{\mathcal{G}}\left[\prod_{j \neq i} \mathbb{1}\left\{j \to i \notin \mathcal{E}_G\right\}\right]. \tag{4.17}$$

Substituting the indicator random variable by its complement yields,

$$p(\mathbf{x}_i \in \mathcal{C}_O) = \mathbb{E}_{\mathcal{G}}\left[\prod_{j \neq i} \left(1 - \mathbb{1}\left\{j \to i \in \mathcal{E}_G\right\}\right)\right]. \tag{4.18}$$

Because the vertices bind independently, the expected value operator is multiplicative (Ross, 2007, p. 52), which allows us to move the expected value operator inside the product,

$$p(\mathbf{x}_i \in \mathcal{C}_O) = \prod_{j \neq i} \left(1 - \mathbb{E}_{\mathcal{G}}\left[\mathbb{1}\left\{j \to i \in \mathcal{E}_G\right\}\right]\right). \tag{4.19}$$

We employ the same argument that we used for Equation 4.16, and we rewrite the expected value of the indicator random variable as the binding probability,

$$p(\mathbf{x}_i \in \mathcal{C}_O) = \prod_{j \neq i} \left(1 - p\left(j \to i \in \mathcal{E}_G\right)\right), \tag{4.20}$$

which is equal to the probability that data point $\mathbf{x}_j$ chooses data point $\mathbf{x}_i$ as its neighbour (see Equation 4.4). Hence, the probability that data point $\mathbf{x}_i$ belongs to the outlier class is

$$p(\mathbf{x}_i \in \mathcal{C}_O) = \prod_{j \neq i} \left(1 - b_{ji}\right) , \tag{4.21}$$

which, in words, states that the outlier probability of data point $\mathbf{x}_i$ is the probability that data point $\mathbf{x}_i$ is never chosen as a neighbour by the other data points. □

Figure 4.10 illustrates, for completeness, the final matrix transformation. The output matrix $\mathbf{\Phi}$, which holds the outlier probabilities, is obtained by applying Equation 4.14 to the binding matrix $\mathbf{B}$. The right side of Figure 4.10 contains a plot of our example data set with the data points filled with the colour corresponding to their outlier probability (see the colour bar next to matrix $\mathbf{B}$). By Equation 4.22 we formally conclude our description of how SOS computes outlier probabilities, in terms of Definition 2.2: the scores produced by the outlier scoring algorithm ϕ_{SOS}, are equivalent to the outlier probabilities.

$$\phi_{\text{SOS}}(\mathbf{x}_i) \equiv p(\mathbf{x}_i \in \mathcal{C}_O) \ . \tag{4.22}$$

In the next subsection we transform the outlier scoring algorithm ϕ_{SOS} into an outlier-selection algorithm, f_{SOS}, i.e., we transform the outlier probabilities into the classifications 'outlier' and 'inlier'.

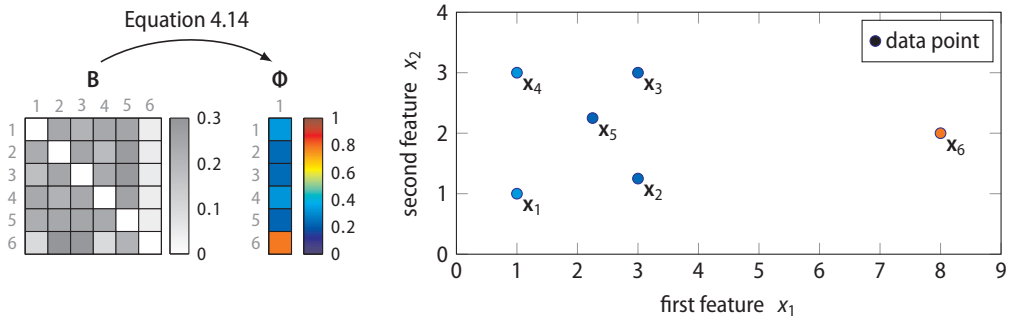


Figure 4.10 Outlier probabilities of the example data set. **Left:** The outlier probabilities in the output matrix $\mathbf{\Phi}$ are obtained by applying Equation 4.14 to the binding matrix $\mathbf{B}$. **Right:** A plot of our example data set with the data points filled by the colour corresponding to their outlier probability as computed by SOS.

4.2.7 Classifying outliers

In the previous six subsections, we have presented SOS as an outlier *scoring* algorithm ϕ_{SOS}, because it maps data points onto outlier scores (see Definition 2.2). It is time for SOS to fulfil its name and transform it into an outlier *selection* algorithm f_{SOS}. We recall from Chapter 2, and in particular Equation 2.2 on page 24, that any outlier scoring algorithm can be transformed into an outlier-selection algorithm. Stated differently, by thresholding the computed outlier scores, classifications of the form 'outlier' and 'inlier' are obtained. Although SOS computes outlier probabilities instead of outlier scores, classifications are obtained in the same way.

$$f_{\text{SOS}}(\mathbf{x}) = \begin{cases} \text{outlier} & \text{if } \phi_{\text{SOS}}(\mathbf{x}) > \theta, \\ \text{inlier} & \text{if } \phi_{\text{SOS}}(\mathbf{x}) \leq \theta. \end{cases} \tag{4.23}$$

If the expert sets the threshold θ to 0.5, then applying Equation 4.23 to the outlier probabilities of the example data set results in the classifications as shown in Figure 4.11. The figure reveals that the first five data points, i.e., $\{\mathbf{x}_1, \ldots, \mathbf{x}_5\}$, are classified as inlier and that data point $\mathbf{x}_6$ is classified as outlier. We can indeed verify that only the outlier probability of $\mathbf{x}_6$, i.e., $\phi_{\text{SOS}}(\mathbf{x}_6) = 0.788$, exceeds the threshold of 0.5. The selection boundary is obtained using the first five data points. So, the selection boundary indicates the region where a sixth data point would be classified as inlier. Because in the example data set, data point $\mathbf{x}_6$ lies outside the selection boundary, it is classified as outlier.

Choosing a proper threshold for a certain real-world application can be challenging. If we are equipped with a loss associated with misclassifications, the Bayesian risk framework may be used to set the threshold so that the average expected loss of our decisions is minimised (Zellner, 1986).

4.2.8 Adaptive variances via the perplexity parameter

A possible challenge with the outlier probability in Equation 4.14 is that two data points that are similar to each other, but dissimilar from the remaining data points may have a low outlier probability, because the two data points have sufficient affinity with each other. We avoid this challenge by setting adaptively the variances σ_i^2 that are used for computing affinities in Equation 4.2.

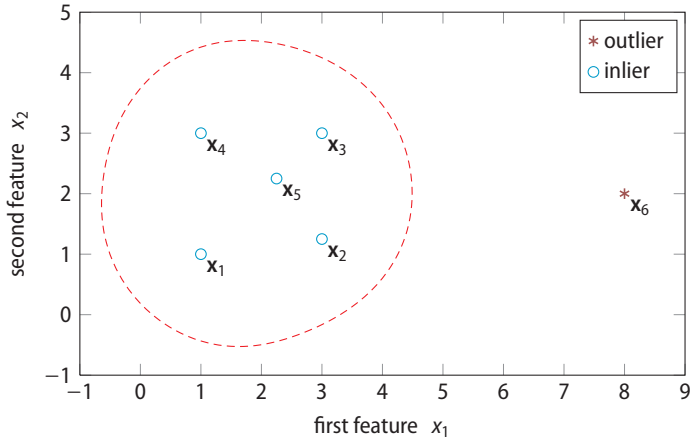


Figure 4.11 Classifications made by SOS on the example data set. Data point $\mathbf{x}_6$ is selected as outlier. The red, dashed line illustrates the selection boundary that corresponds to a threshold θ of 0.5.

We recall from Subsection 4.2.2 that SOS has one parameter h, called the perplexity. So far, we have treated the perplexity parameter h as a smooth measure for the effective number of neighbours of a data point, set by the expert. In fact, perplexity is a measurement that stems from the field of information theory. Perplexity may be computed for a probability distribution, for example, to compare it with another probability distribution (Jelinek, Mercer, Bahl, and Baker, 1977). In SOS, perplexity is employed to set adaptively the variances in such a way that each data point has h effective neighbours (Hinton and Roweis, 2003). To be precise, we require that the binding distribution $\mathbf{b}_i$ of each data point $\mathbf{x}_i$ has a perplexity that is equal to the perplexity parameter set by the expert,

$$h(\mathbf{b}_i) = 2^{H(\mathbf{b}_i)} \,, \tag{4.24}$$

where $H(\mathbf{b}_i)$ is the Shannon entropy of $\mathbf{b}_i$ (Shannon, 1948; MacKay, 2003),

$$H(\mathbf{b}_i) = -\sum_{\substack{j=1 \\ j \neq i}}^{n} b_{ij} \log_2 \left(b_{ij} \right) \,. \tag{4.25}$$

We remark that b_{ii} is not taken into account, because a data point is never its own neighbour. As a consequence of this requirement, the variances adapt to the local density of data points, in such a way that a higher density leads to a lower variance, causing the affinity a_{ij} to decay faster. The effect of such an adaptive variance is that $\mathbf{x}_i$ distributes, in general, around 90% of its affinity to its h nearest neighbours. So, indeed, the value

of perplexity h may be interpreted as a smooth measure for the effective number of neighbours of a data point (van der Maaten and Hinton, 2008).

Figure 4.12 shows the influence that the perplexity parameter h has on the outlier probabilities. Having a fixed perplexity h, rather than a fixed variance σ^2 (cf. bandwidth in kernel density estimation (Parzen, 1962)), allows the SOS algorithm (1) to classify accurately data points in data sets with varying densities, and (2) to avoid the challenge with small clusters of outliers.

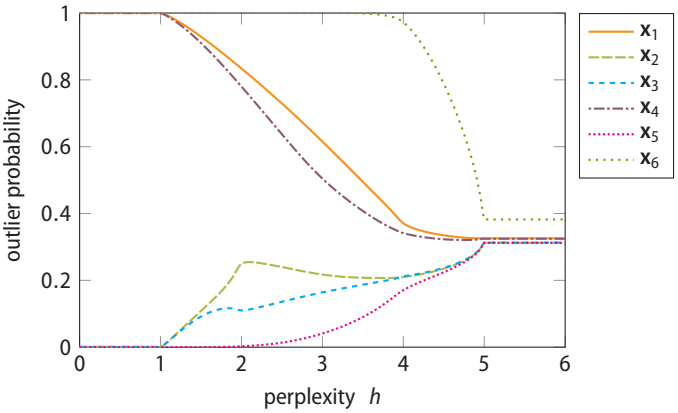


Figure 4.12 Influence of the perplexity h on the outlier probabilities of the six data points in the example data set.

Figure 4.13 shows how the variance influences the perplexity of the binding distribution. The values of the variances that correspond to the desired perplexity (which is set by the expert) are found using a binary search. The binary search starts with a sufficiently large interval (e.g., from 0.1 to 30) in which the desired variances lie. (This initial interval is derived from the distances between the data points.) In each iteration, the binary search bisects the interval and then selects a subinterval until the desired variances for each data point are found. Figure 4.14 shows the first 10 iterations of a binary search for the example data set. The figure shows that the perplexity of each binding distribution converges to the desired perplexity ($h = 4.5$).

4.3 Qualitative evaluation of SOS and four related algorithms

In this section we introduce and discuss four related outlier-selection algorithms and evaluate them, together with SOS, in a qualitative manner. The aim is to make this qualitative evaluation complementary to the quantitative evaluation (of the same

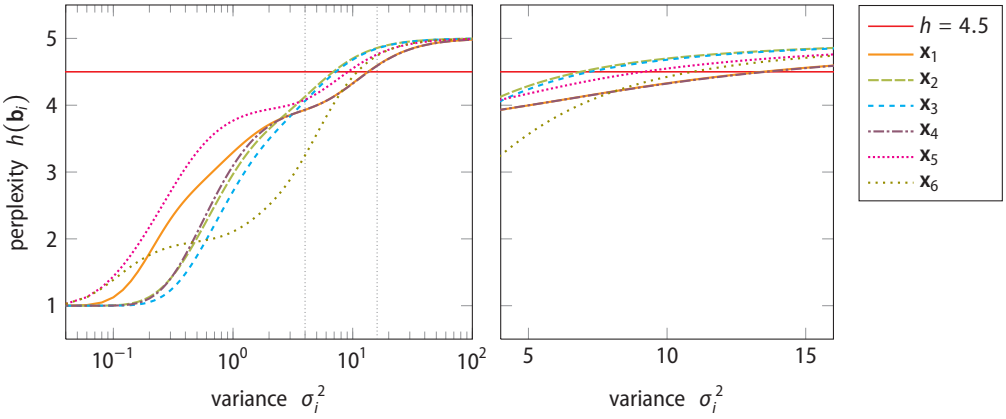


Figure 4.13 Six graphs of the perplexity $h(\mathbf{b}_i)$ with respect to the variance σ_i^2 for the six data points in the example data set. For each data point, a different variance is required such that the binding probability distribution $\mathbf{b}_i$ has the desired perplexity h of 4.5 (denoted by the horizontal, red line). **Left:** Semi-log plot with a logarithmic scale on the x-axis. **Right:** Linear plot with the same graphs, zoomed in on the range of values where each variance corresponds to the desired perplexity.

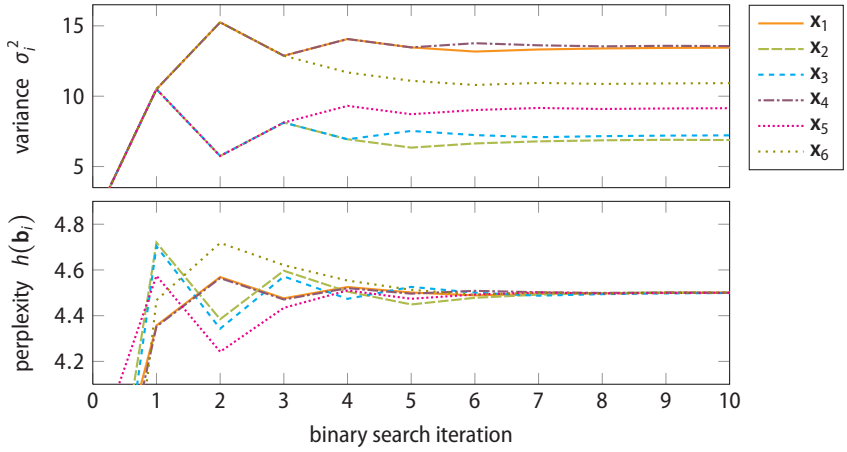


Figure 4.14 The first 10 iterations of the binary search that sets adaptively the variances. The desired perplexity h is set to 4.5. **Top:** The current variance for each of the six data points in the example data set. **Bottom:** The current perplexities given the current values of the variances.

algorithms) presented in Section 4.4. The four related algorithms are: K-Nearest Neighbour Data Description (KNNDD) by Tax (2001), Local Outlier Factor (LOF) by Breunig et al. (2000), Local Correlation Integral (LOCI) by Papadimitriou et al. (2003), and Least Squares Outlier Detection (LSOD) by Hido et al. (2008) and Kanamori, Hido, and Sugiyama (2009). Please note that the algorithms KNNDD, LOF, and LOCI have also been employed in Chapter 3. There, the algorithms were semi-supervised. In the current chapter we employ their unsupervised versions. (SVDD is not included in this comparison because there is no unsupervised version available.)

To achieve our aim, we first explain outlier-score plots in Subsection 4.3.1. Subsequently, for each algorithm (SOS included) we (1) provide a brief description and (2) discuss several of its strong and weak points using the outlier-score plots shown in Figure 4.15 (Subsections 4.3.2–4.3.6).

4.3.1 Outlier-score plots

We qualitatively evaluate SOS and the four related algorithms using outlier-score plots on three small example data sets. The fifteen corresponding outlier-score plots are shown in Figure 4.15. An outlier-score plot is a two-dimensional plot that shows how well the algorithm captures the structure of a particular data set. (They are related to the plots employed in Aha, Kibler, and Albert (1991), which illustrate the decision boundaries of various instance-based learning algorithms.) So, an outlier-score plot may increase our understanding of outlier-selection algorithms.

In practice, an outlier-score plot is obtained as follows. First, we assume that we have a two-dimensional data set $\mathbf{X}$. Second, we generate a set of data points $\mathbf{Z}$ whose feature vectors correspond to the pixels of the resulting outlier-score plot. For instance, generating a plot of size 100 by 100 pixels requires $|\mathbf{Z}| = 10{,}000$ data points. Third, we remove one data point ($\mathbf{x}_{\text{new}}$) of $\mathbf{Z}$ and add it to $\mathbf{X}$. Fourth, we apply the outlier-selection algorithm to the data set $\mathbf{X}$ and record the outlier score (or probability in the case of SOS) for data point $\mathbf{x}_{\text{new}}$. Fifth, we remove $\mathbf{x}_{\text{new}}$ from $\mathbf{X}$. Steps 3, 4, and 5 are repeated until $\mathbf{Z}$ is empty. Sixth, the pixels of the plot are coloured by mapping the recorded outlier scores onto a colour map (cf. the colour map next to the output matrix in Figure 4.1 on page 63). The mapping of colours to outlier probabilities / scores varies by plot, since the possible

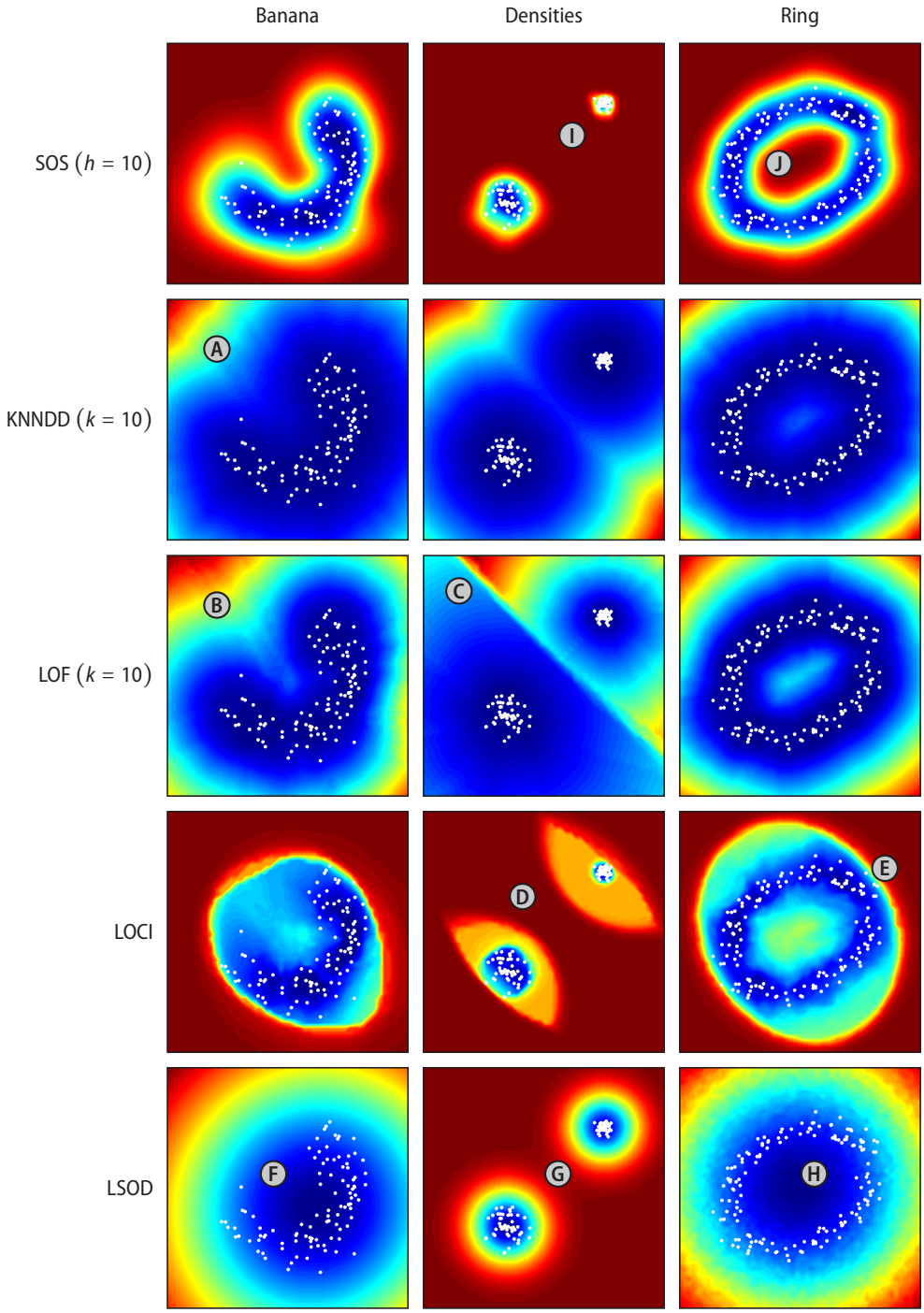


Figure 4.15 Outlier-score plots for SOS and four related outlier-selection algorithms as applied to three small example data sets. The colour at a certain location corresponds to the outlier probability / score that would be assigned to a new data point, should it appear at that location.

minimum and maximum scores may differ by algorithm and by data set (except for SOS). Finally, in order to see the data set $\mathbf{X}$, its data points are plotted as white dots.[1]

The first data set in Figure 4.15, *Banana*, contains a banana-shaped cluster with 100 data points. The asymmetric shape allows us to investigate how algorithms cope with irregular distributions. The second data set, *Densities*, contains two Gaussian-distributed clusters with 50 data points each. The second cluster is denser than the first, which allows us to investigate how the algorithms cope with varying densities. The third data set, *Ring*, contains a rotated ellipse-shaped cluster with 150 data points. The ring allows us to investigate how algorithms cope with low density regions that are enclosed by data points. We note that each of the five algorithms is applied to exactly the same three data sets and that its parameter settings are kept constant. The number of data points has no influence on the outlier-score plots. We are now ready for a description and qualitative evaluation of SOS and the four related algorithms.

4.3.2 SOS

The results of SOS are shown in the top row of Figure 4.15. We see that SOS has a smooth boundary around the data points. For the Banana data set, the shape is captured well. For the Densities data set, we see that the boundary around the denser cluster is tighter than around the big, sparse cluster (see I). This indicates that SOS takes the relative density of the data well into account. For the Ring data set, the outlier probability assigned to a data point in the middle of the ring would be roughly equal to those appearing outside the ring (see J). The transition from a low to high outlier probability seems often smoother than with the other algorithms, which can be explained by the use of affinities that causes the property of 'being a neighbour' to be a smooth property.

[1] We restrict ourselves to two dimensions because it is not practical to visualise higher-dimensional outlier-score plots. For instance, a three-dimensional data set would require us to show the colour (outlier score) of all the 10,000,000 data points that lie in the cube of 100 by 100 by 100 voxels. The goal of outlier-score plots to gain an intuitive understanding of the outlier-selection algorithms. However, because the outlier scores computed by each of the five algorithms are determined by the Euclidean distance between the data points, we may expect that the performance of each algorithm to be affected similarly by higher-dimensional data sets. In Section 4.4.1 we report on the performances obtained on real-world data sets, which have a variety of dimensionalities.

4.3.3 KNNDD

The K-Nearest Neighbour Data Description (KNNDD) by Tax (2001) is an algorithm with one free parameter k. KNNDD defines the outlier score for a data point $\mathbf{x}$ as the ratio of the distance between $\mathbf{x}$ and its k-nearest neighbour $\mathbf{x}'$, and the distance between $\mathbf{x}'$ and *its* k-nearest neighbour. KNNDD differs from SOS in that it employs discrete rather than soft neighbourhood boundaries (this holds for LOF and LOCI too).

For the Banana data set, KNNDD generalises the shape of the data too much. The outlier scores for the Densities data set increase similarly for both clusters, which means that KNNDD is not so much influenced by the density of the data. Data points appearing in the middle of the Ring data set get a moderately higher outlier score. KNNDD has the drawback that outlier scores are unbounded (see A).

4.3.4 LOF

The Local Outlier Factor (Breunig et al., 2000) computes an outlier score by estimating the relative density in the neighbourhood of the data point. A data point, whose nearest neighbours have a smaller neighbourhood with an equal number of data points, is assigned a higher outlier score. The neighbourhood size is determined by a free parameter k.

LOF's outlier scores are also unbounded (see B). It captures the shape of the data in the Banana data set better than KNNDD. At the boundary between the two clusters in the Density data set, the outlier scores exhibit a discontinuity (see C), which is due to the use of discrete neighbourhood boundaries, and the fact that LOF takes densities explicitly into account, as opposed to KNNDD. The outlier scores in the middle of the Ring data set are more increased than with KNNDD.

4.3.5 LOCI

The Local Correlation Integral (Papadimitriou et al., 2003) also estimates the relative density in the neighbourhood of the data point, but then for a whole range of neighbourhood sizes. The outlier score is based on the maximum ratio between the local and global densities that is found in the range. Although some values in the actual algorithm can be adjusted, Papadimitriou et al. claim that LOCI has no free parameters, and will therefore be treated as such. The same holds for LSOD.

The outlier scores computed by LOCI do have a maximum, but this maximum may be different per data set. For the Banana data set, LOCI forms a strange shape around the data points. The strange shapes (see D) in the Densities data set, mostly include higher outlier scores, and are probably a result of the fact that the neighbourhoods of the constituent data points include data points from both clusters. The shape in the Ring data set seems to be orthogonal with the cluster (see E) and have very tight boundaries at certain places.

4.3.6 LSOD

Least Squares Outlier Detection (Kanamori et al., 2009; Hido et al., 2008) is an 'inlier-based' outlier-selection algorithm. Unlike the other algorithms, it is supervised, in that it uses a given set of data points, labelled as normality. The outlier scores of the remaining data points are given by the ratio of probability densities between the normalities and the remaining data points. For our experiments, we slightly alter LSOD by computing the outlier score of one data point at a time, and treating all other data points as normal. This way, LSOD can be considered as an unsupervised outlier-selection algorithm and thus ensures a fair comparison with the other algorithms. According to Kanamori et al. LSOD, like LOCI, has no free parameters.

For the Banana data set, LSOD seems to model the data as a spherical structure (see F), such that the outlier scores increase linearly with distance from the cluster centre. For the Densities data set, the outlier scores of LSOD seem to be less influenced by the density of clusters (see G). Data points appearing in the middle of the Ring data set would not be selected as outliers (see H).

4.4 Experiments and results

In this section we present our experiments and the corresponding results. We evaluate SOS and the four algorithms discussed in Section 4.3 (i.e., KNNDD, LOCI, LOF, and LSOD) on eighteen real-world data sets (Section 4.4.1) and on seven synthetic data sets (Section 4.4.2).

4.4.1 Real-world data sets

We evaluated SOS and the four related outlier-selection algorithms on eighteen real-world data sets (see Figure 4.16 for the list of data sets).[2] Except for the Delft Pump data set (Ypma, 2001) and the Colon Gene data set (Alon, Barkai, Notterman, Gish, Ybarra, Mack, and Levine, 1999), all data sets come from the UCI Machine Learning Repository (Asuncion and Frank, 2010).[3] Because these data sets contain multiple classes that are not necessarily defined as either normal or anomalous, they are relabelled into multiple one-class data sets using the procedure that is described in Subsection 2.6.1. The performance measure is the weighted AUC (see Subsection 2.6.4).

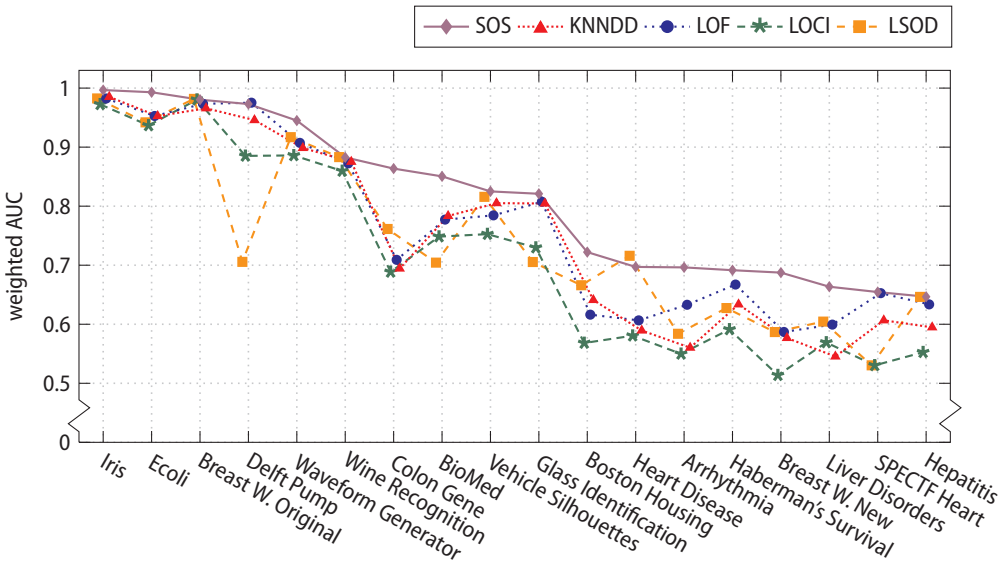


Figure 4.16 The weighted AUC performances of the five outlier-selection algorithms on eighteen real-world data sets. The data sets are ordered by the performances of SOS.

Figure 4.16 shows the weighted AUC performances of the five outlier-selection algorithms on the real-world data sets. The performance of SOS is illustrated by a solid (purple) line, while the other outlier-selection algorithms are illustrated by dashed and dotted lines. For clarity, the data sets are ordered according to the performance of SOS.

[2] The interested reader is referred to Table 5.1 on page 106 for the dimensionality, number of normalities, and number anomalies of each data set.

[3] These data sets have also been employed in Chapter 3, except for six sets which have been replaced by synthetic data sets (see Subsection 4.4.2).

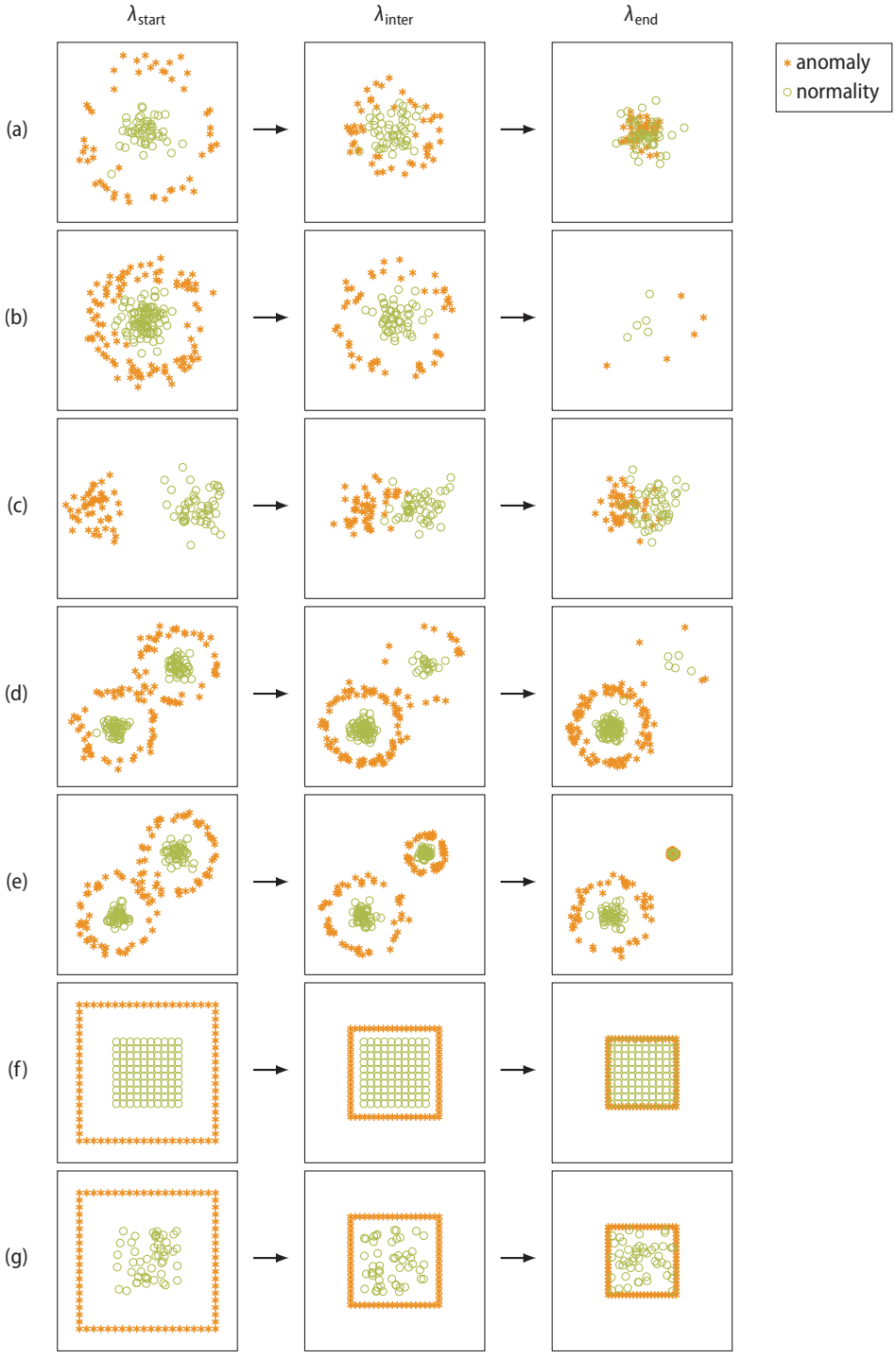


Figure 4.17 From left to right, the three columns illustrate, for each synthetic data set, three instantiations where λ is set to (1) the start value λ_{start}, (2) an intermediate value λ_{inter}, and (3) the end value λ_{end}, respectively.

The figure reveals that SOS has a superior performance on twelve data sets. On the other six data sets its performance was at least 98% of the best performing algorithm.

For completeness, the AUC performances of SOS, KNNDD, LOF, LOCI, and LSOD on the 47 real-world one-class data sets are stated in Table 4.2 for various parameter settings. We recall that these 47 one-class data sets are the result of relabelling the 18 real-world data using the procedure that is described in Subsection 2.6.1. In Table 4.2, the real-world data set is stated as the data set name, as used in Figure 4.16, followed by the name of the normal class in italics and in brackets, e.g., 'Iris *(Setosa)*'. In addition to the maximum achieved AUC performance as is also shown in Figure 4.16, Table 4.2 also shows the performances for other parameter values. In the case of SOS, h_5, for example, is shorthand notation for $h = 5$. The maximum AUC performance is given by h_b. The parameter value corresponding to the best AUC performance is given by h and is indicated in gray. For KNNDD and LOF, the columns are k_b and k, respectively. Note that LOCI and LSOD have no free parameters.

4.4.2 Synthetic data sets

Although real-world data sets give a serious indication of how well the five algorithms will perform in a real-world setting, we may gain additional insight into their behaviour using synthetic data sets. Therefore, we designed seven synthetic data sets that contain data points from both the normal and the anomalous class. The synthetic data sets are two-dimensional. For each synthetic data set we introduced a single parameter, λ, that determines one property of the data set. In one data set, for example, λ corresponds to the distance between two Gaussian clusters. If we gradually adjust λ, i.e., adjust the distance between the two clusters while keeping the other properties constant, then we observe

Table 4.1 The seven synthetic data sets controlled by parameter λ.

	Parameter λ			
Data set	Determines	λ_{start}	step size	λ_{end}
(a)	Radius of ring	5	0.1	2
(b)	Cardinality of cluster and ring	100	5	5
(c)	Distance between clusters	4	0.1	0
(d)	Cardinality of one cluster and ring	0	5	45
(e)	Density of one cluster and ring	1	0.05	0
(f)	Radius of square ring	2	0.05	0.8
(g)	Radius of square ring	2	0.05	0.8

how cluster-overlap influences the performances of the algorithms under consideration. In general, the purpose of the synthetic data sets is to measure the resilience of each algorithm for the different data-set properties.

Table 4.1 lists all seven synthetic data sets and the function of parameter λ, viz. what property it determines. Besides cluster overlap (data sets (a), (c), (f), and (g)), we also evaluate the influence of cluster densities (e) and cluster cardinality ((b) and (d)).

Figure 4.17 illustrates for each synthetic data set three instantiations with three different values of λ, namely the start value, an intermediate value, and the end value. The reader should note that, similar to the real-world data sets, the true class-labels indicate whether a data point is anomalous or normal, and not whether it is an outlier or inlier. Consequently, due to the unsupervised nature of the outlier-selection algorithms, anomalous data points might not be selected as outliers (corresponding to a *miss*), especially as λ reaches its end value, λ_{end}.

Because the synthetic data sets contain random samples from various distributions and are generated anew for each evaluation, we applied each algorithm to 100 instantiations of each data set per value of λ, and computed the average AUC performances. Figure 4.18 displays the performances of the five algorithms on the seven synthetic data sets. Again, the performance of SOS is illustrated by a solid (purple) line and the other algorithms by dashed lines.

4.5 Discussion of the results

To compare the performances of multiple algorithms on multiple data sets, Demšar (2006) suggests the Neményi test (Neményi, 1963). The Neményi test checks for significant difference by ranking each algorithm for each data set, where the best performing algorithm is assigned the rank of 1, the second best the rank of 2, and so forth. Two algorithms are significantly different when their average ranks differ more than the critical distance, which is in our case 1.438 for $p = .05$.

We apply the Neményi test on the performances obtained by the five outlier-selection algorithms on the eighteen real-world data sets. The outcome is shown in the top part of Figure 4.19 by a critical difference diagram. Groups of methods that are not significantly different (at $p = .05$) are connected by a horizontal bar. For the real-world data sets, three (partially overlapping) groups of algorithms, ranked from low to high performance,

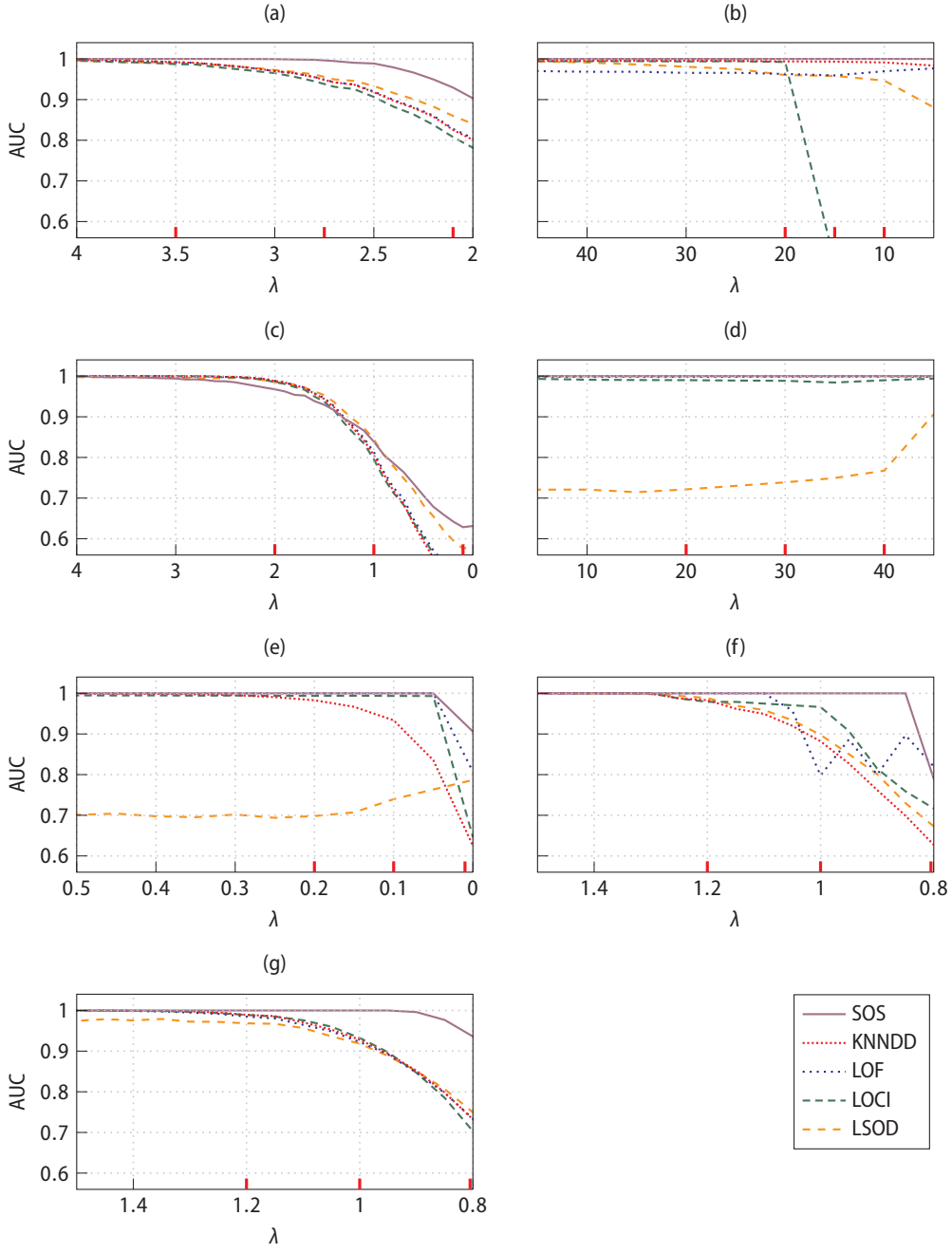


Figure 4.18 The AUC performances of the five outlier-selection algorithms on the seven synthetic data sets.

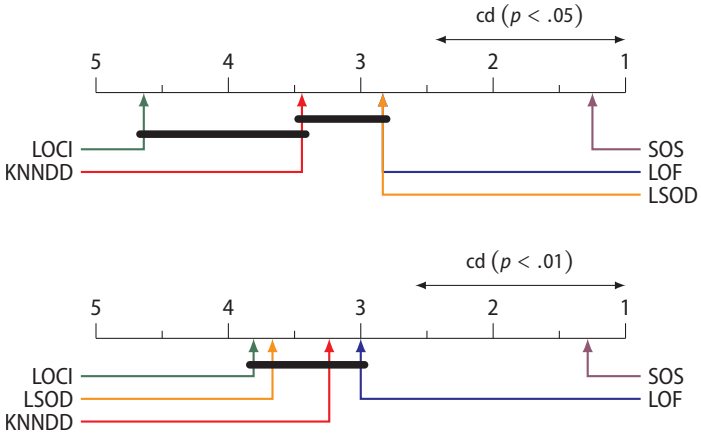


Figure 4.19 Critical difference diagrams. **Top:** From applying the outlier-selection algorithms on the eighteen real-world data sets. The critical distance is 1.438. **Bottom:** From applying the algorithms on the seven synthetic data sets. The critical distance is 1.588. Groups of algorithms that are connected are not significantly different.

are identified: (1) LOCI and KNNDD, (2) KNNDD, LSOD, and LOF, and (3) SOS. The performance of SOS is significantly higher than the other algorithms.

From the results on synthetic data sets (b) and (d), we may conclude that SOS has a superior performance with data sets that contain clusters with low cardinality. The performance on data set (e) indicates that SOS copes best with data sets containing clusters with varying densities. Although on data set (c), SOS is outperformed by the other algorithms for intermediate values of λ , the performance increases for $\lambda \leq 1$. This observation, combined with the superior results on data sets (a), (f), and (g), implies that SOS is, in general, less sensitive to cluster overlap than the other algorithms.

Apart from a few exceptions, the algorithms KNNDD, LOF, and LOCI show similar trends among the seven synthetic data sets. This was expected since they are all based on the *k*-nearest neighbour algorithm. For data set (b), LOCI shows a poor performance when the cardinality of the cluster is below 20. This is due to the requirement of LOCI to have a sample size of at least 20 (Papadimitriou et al., 2003). We expect that without this constraint, LOCI will perform comparably to KNNDD and LOF for $\lambda < 20$. Regarding LSOD, the results on the real-world data sets already showed that it has a relatively poor performance. Its performance on the synthetic data sets (d) and (e) confirm that LSOD is unable to handle data sets containing clusters of different cardinality or densities.

The bottom part of Figure 4.19 shows the critical difference diagram for the synthetic data sets. There are two groups of algorithms, ranked from low to high performance: (1) LOCI, LSOD, KNNDD, and LOF; and (2) SOS. The performance of SOS is significantly higher (at $p = 0.01$) than the other algorithms.

4.6 Chapter conclusions

In this chapter we set out to answer RQ2: *Can an effective outlier-selection algorithm be devised that employs the concept of affinity?* To this end, we developed and evaluated Stochastic Outlier Selection (SOS), a novel unsupervised algorithm for classifying data points as outliers in a data set. SOS computes for each data point an outlier probability, using affinities. The outlier probabilities provide three advantages with respect to unbounded outlier scores as computed by existing outlier-selection algorithms (cf. Subsection 4.1.2). First, a Bayesian risk model can be used to find an appropriate threshold for classifying data points as outliers (see Gao and Tan, 2006). Second, an ensemble outlier selection framework can be built by aggregating probabilities from individual outlier-selection algorithms. Third, we expect that outlier probabilities are easier to interpret by domain experts than outlier scores.

We described an evaluation procedure that enables us to evaluate unsupervised outlier-selection algorithms with standard benchmark data sets. We introduced the concept of an outlier-score plot, which allowed us to inspect visually how well an algorithm captures the structure of a data set (cf. Subsection 4.3.1). Using both real-world data sets and synthetic data sets, we have shown that SOS has an outstanding performance when compared to the current outlier-selection algorithms, KNNDD, LOF, LOCI, and LSOD. The Neményi statistical test revealed that SOS's performance is significantly higher. The seven synthetic data sets were parametrised by λ, such that the outlier-selection algorithms could be evaluated on individual data-set properties.

Our answer to RQ2 reads as follows. From our empirical results we observe that (1) SOS is an effective algorithm for classifying data points as outliers in a data set and that (2) SOS compares favourably to state-of-the-art outlier-selection algorithms. We may therefore conclude that the concept of affinity, which forms the basis SOS, is successfully applied to the problem of outlier selection.

Table 4.2 AUC performances on real-world one-class data sets.

	SOS							KNNDD							LOF							LOCI	LSOD
Dataset *(normal class)*	h_5	h_{10}	h_{20}	h_{50}	h_{100}	h_b	h	k_5	k_{10}	k_{20}	k_{50}	k_{100}	k_b	k	k_5	k_{10}	k_{20}	k_{50}	k_{100}	k_b	k	–	–
Iris *(Setosa)*	1	1	1	1	1	1	10	1	1	1	–	–	1	1	1	1	1	–	–	1	10	1	1
Iris *(Versicolor)*	.97	.98	.98	.98	.98	1	49	.98	.98	.97	–	–	.99	3	.97	.97	.96	–	–	.98	7	.97	.96
Iris *(Virginica)*	.94	.96	.97	.97	.97	.99	49	.95	.95	.94	–	–	.97	1	.95	.96	.93	–	–	.97	3	.95	.99
Breast W. Original *(malignant)*	.81	.85	.88	.91	.92	.98	250	.68	.8	.88	.94	.96	.97	150	.7	.74	.85	.96	.97	.97	149	.98	.98
Breast W. Original *(benign)*	.81	.85	.88	.91	.92	.98	250	.68	.8	.88	.94	.96	.97	150	.7	.74	.85	.96	.97	.97	149	.98	.98
Heart Disease *(diseased)*	.62	.58	.56	.55	.54	.67	2	.51	.5	.5	.49	.5	.51	3	.53	.55	.54	.48	.5	.56	11	.53	.68
Heart Disease *(healthy)*	.67	.64	.61	.6	.6	.72	2	.65	.65	.65	.64	.62	.66	16	.59	.61	.64	.63	.6	.65	18	.62	.75
BioMed *(healthy)*	.87	.89	.89	.88	.88	.91	122	.9	.9	.9	.89	.88	.91	4	.88	.87	.88	.89	.87	.89	4	.88	.9
BioMed *(diseased)*	.71	.67	.65	.64	.02	.73	2	.33	.33	.36	.51	–	.55	61	.52	.47	.44	.37	–	.56	42	.49	.33
Arrhythmia *(normal)*	.75	.76	.77	.79	.81	.85	200	.77	.77	.77	.77	.77	.78	16	.77	.78	.77	.77	.76	.78	9	.76	.76
Arrhythmia *(abnormal)*	.36	.33	.31	.3	.3	.5	1	.26	.25	.24	.25	.25	.28	1	.3	.27	.25	.26	.29	.45	1	.27	.35
Hepatitis *(normal)*	.6	.57	.53	.51	.55	.65	2	.59	.59	.59	.59	.57	.59	79	.54	.58	.57	.6	.53	.63	109	.55	.65
Ecoli *(periplasm)*	.9	.92	.91	.99	.99	.99	51	.94	.95	.95	.95	–	.95	47	.93	.94	.95	.72	–	.95	39	.94	.94
Delft Pump *(AR app.)*	1	1	1	1	1	1	100	1	1	.99	.99	.96	1	1	1	1	.99	.98	.84	1	4	.98	.95
Delft Pump *(5x3)*	.93	.96	.98	.99	.99	.99	70	.94	.91	.82	.71	.62	.95	1	.98	.97	.8	.83	.47	.99	3	.93	.66
Delft Pump *(5x1)*	.89	.93	.94	.94	.93	.97	132	.88	.76	.68	.56	.54	.94	1	.92	.79	.75	.46	.47	.95	1	.87	.68
Delft Pump *(3x2)*	.88	.96	.98	.96	.95	.99	136	.9	.83	.72	.59	.58	.94	1	.98	.77	.7	.5	.58	.99	4	.87	.69
Delft Pump *(2x2)*	.79	.88	.91	.88	.44	.92	99	.89	.81	.69	.52	–	.9	1	.94	.73	.66	.5	–	.97	2	.84	.66
Delft Pump *(1x3)*	.86	.92	.97	.92	.74	1	70	.95	.91	.7	.65	–	.97	1	.93	.89	.47	.64	–	.96	3	.9	.74
Delft Pump *(5x3 noisy)*	.9	.91	.88	.84	.82	.91	7	.75	.6	.54	.51	.49	.9	1	.61	.35	.46	.53	.48	.93	3	.74	.55
Delft Pump *(5x1 noisy)*	.94	.96	.95	.91	.45	.96	10	.83	.73	.73	.65	–	.94	1	.74	.78	.79	.56	–	.98	2	.85	.75
Delft Pump *(3x2 noisy)*	.86	.97	.99	.95	.41	.99	82	.92	.82	.73	.65	–	.97	1	.94	.67	.66	.53	–	1	2	.92	.72
Delft Pump *(2x2 noisy)*	.86	.95	.97	.92	.49	.98	63	.89	.81	.7	.58	–	.94	1	.97	.8	.53	.54	–	1	2	.92	.74
Delft Pump *(1x3 noisy)*	.93	.99	.99	.71	.71	1	40	.9	.87	.8	–	–	.96	1	1	.97	.66	–	–	1	7	.94	.79

Table 4.2 (Continued)

	SOS							KNNDD							LOF							LOCI	LSOD
Dataset *(normal class)*	h_5	h_{10}	h_{20}	h_{50}	h_{100}	h_b	h	k_5	k_{10}	k_{20}	k_{50}	k_{100}	k_b	k	k_5	k_{10}	k_{20}	k_{50}	k_{100}	k_b	k	–	–
Breast W. New *(non-ret)*	.61	.55	.53	.48	.49	.7	2	.58	.57	.58	.58	.58	.59	1	.51	.55	.54	.57	.56	.59	64	.51	.61
Breast W. New *(ret)*	.63	.55	.51	.36	.36	.65	3	.43	.4	.44	–	–	.52	1	.53	.41	.48	–	–	.57	2	.52	.53
SPECTF Heart *(0)*	.94	.94	.93	.92	.85	.95	93	.9	.88	.86	.85	–	.97	1	.83	.84	.84	.85	–	.92	1	.91	.98
SPECTF Heart *(1)*	.53	.48	.46	.43	.44	.54	4	.26	.22	.2	.2	.2	.47	1	.31	.25	.21	.2	.22	.55	1	.39	.36
Colon Gene *(2)*	.61	.63	.68	.82	.82	.86	39	.68	.69	.68	–	–	.69	12	.67	.7	.69	–	–	.71	9	.69	.76
Glass Identification *(float)*	.86	.86	.86	.87	.64	.9	68	.82	.79	.76	.73	–	.86	1	.85	.87	.85	.63	–	.87	12	.78	.86
Glass Identification *(nonfloat)*	.72	.72	.71	.72	.31	.75	75	.73	.71	.69	.65	–	.75	1	.73	.74	.69	.67	–	.75	9	.68	.56
Liver *(1)*	.61	.59	.57	.55	.58	.68	2	.59	.59	.59	.59	.58	.6	2	.56	.57	.59	.59	.56	.6	144	.58	.67
Liver *(2)*	.64	.62	.61	.59	.59	.65	2	.51	.5	.5	.49	.48	.51	4	.54	.55	.52	.49	.44	.6	2	.56	.56
Wine Recognition *(1)*	.95	.95	.96	.95	.98	.98	100	.93	.94	.96	.95	–	.96	20	.87	.82	.96	.93	–	.96	22	.95	.92
Wine Recognition *(2)*	.81	.77	.75	.75	.71	.81	5	.83	.82	.82	.82	–	.83	4	.73	.79	.79	.76	–	.82	34	.81	.85
Wine Recognition *(3)*	.87	.85	.83	.82	.82	.87	5	.8	.8	.81	–	–	.84	1	.79	.81	.81	–	–	.83	38	.83	.89
Waveform Generator *(0)*	.8	.82	.83	.85	.87	.92	300	.84	.84	.85	.85	.86	.86	127	.82	.84	.84	.86	.87	.87	94	.84	.87
Waveform Generator *(1)*	.85	.87	.88	.9	.91	.97	300	.9	.9	.91	.91	.92	.92	150	.88	.89	.9	.91	.93	.93	144	.92	.95
Waveform Generator *(2)*	.86	.87	.89	.9	.92	.95	300	.9	.9	.91	.91	.91	.91	147	.87	.89	.89	.9	.91	.91	112	.9	.93
Vehicle Silhouettes *(van)*	.95	.96	.97	.97	.97	.97	190	.96	.95	.93	.87	.81	.96	4	.95	.93	.91	.89	.73	.95	3	.92	.96
Vehicle Silhouettes *(Saab)*	.7	.72	.72	.73	.74	.74	99	.71	.7	.68	.64	.61	.71	5	.65	.65	.57	.45	.62	.67	3	.68	.74
Vehicle Silhouettes *(bus)*	.84	.85	.85	.87	.84	.9	200	.85	.81	.77	.71	.68	.91	1	.72	.68	.78	.75	.67	.88	1	.8	.87
Vehicle Silhouettes *(Opel)*	.67	.66	.65	.65	.65	.69	2	.65	.64	.63	.6	.59	.65	4	.59	.58	.55	.42	.62	.65	123	.62	.7
Haberman's Survival *(>5yr)*	.71	.69	.65	.62	.61	.72	4	.67	.68	.68	.66	.62	.68	10	.62	.67	.7	.65	.59	.71	18	.64	.64
Haberman's Survival *(<5yr)*	.62	.56	.53	.5	.36	.63	4	.41	.4	.43	.47	–	.5	74	.47	.45	.45	.5	–	.56	71	.45	.59
Boston Housing *(MEDV<35)*	.68	.66	.63	.59	.58	.72	2	.61	.61	.56	.51	.46	.62	1	.52	.59	.56	.52	.57	.59	11	.56	.65
Boston Housing *(MEDV>35)*	.69	.64	.6	.33	.33	.72	3	.66	.66	.63	–	–	.83	1	.42	.64	.63	–	–	.82	3	.62	.85

We observe that SOS is not the best-performing algorithm on all data sets. This observations leads to an important questions: for which type of problems (i.e., real-world data sets) is SOS the most suitable outlier-selection algorithm? We investigate this question in the next chapter, where we evaluate 19 algorithms on 255 data sets.

Chapter 5

Meta-features for one-class data sets

Contents:

In this chapter we set out to answer RQ3: *To what extent can we solve the one-class classifier selection problem using a meta-learning approach?* The 'No Free Lunch' theorem implies that there is no single best one-class classifier. For each one-class data set, there may be a different one-class classifier that performs best. The performance of a one-class classifier is greatly determined by the characteristics (or meta-features) of the one-class data set at hand. Our goal is to understand the relationship between meta-features and one-class classifier performance. In the chapter, we define 36 meta-features and apply them to 255 data sets; this procedure generates a set of meta-feature values. These values are used as input for the next chapter, where we use meta-learning to relate the meta-features to performance of one-class classifiers.

Outline:

5.1 No free lunch for one-class classification. **5.2** The one-class classifier selection problem. **5.3** Meta-learning. **5.4** Overview of one-class data sets. **5.5** Preprocessing one-class data sets. **5.6** Meta-features. **5.7** Results and discussion. **5.8** Chapter summary.

This chapter deals with the problem of selecting the one-class classifier that will perform best on a given one-class data set. It is a challenging problem. Given the large number of available one-class classifiers, it is infeasible to apply them all and choose the one with the best performance. Moreover, in real-world applications it is an even bigger problem because domain experts may lack (1) the knowledge to pre-select a limited number of one-class classifiers, and (2) the time to apply even a limited number of one-class classifiers (Brazdil, Giraud-Carrier, Soares, and Vilalta, 2009). Rice (1976) formalised the above problem for *binary classifiers* as the algorithm-selection problem. Since we are considering one-class classifiers only, we call it the one-class classifier selection problem (a more detailed explanation and a formal definition will be given in Section 5.2).

Meta-learning is a research area within the field of machine learning. It aims to solve the algorithm-selection problem automatically (cf. Aha, 1992; Mitchie, Spiegelhalter, and Taylor, 1994; King, Feng, and Sutherland, 1995; Sohn, 1999; Peng, Flach, Soares, and Brazdil, 2002; Kalousis, Gama, and Hilario, 2004; Ali and Smith, 2006; Brazdil et al., 2009; Jankowski, Duch, and Grąbczewski, 2011; Song, Wang, and Wang, 2012). In this chapter and the next chapter, we apply meta-learning to the one-class classification setting. In other words, we aim to solve the one-class classifier selection problem. To the best of our knowledge, we are the first to do so. Our third research question (RQ3) therefore reads as follows: *To what extent can we solve the one-class classifier selection problem using a meta-learning approach?*

Smith-Miles (2008) lists four prerequisites for solving the *algorithm* selection problem using a meta-learning approach. We adapt the four prerequisites for our *one-class classifier* selection problem as follows. There should be available:

1. a large number of one-class data sets of various complexities;
2. suitable meta-features to characterise the one-class data sets;
3. a large number of diverse one-class classifiers to apply to the one-class data sets;
4. a performance measure to evaluate any one-class classifier performance.

By relating the meta-features (2) to the performances (4) of a large number of one-class classifiers (3) on a large number of one-class data sets (1) we may obtain a comprehensive set of meta-knowledge about the one-class classifiers (cf. Smith-Miles, 2008). In the current chapter (Chapter 5), we focus on prerequisites (1) and (2) by characterising 255 one-class data sets using 36 meta-features. The main result of the current chapter are the 36 meta-features. In Chapter 6, we continue with prerequisite (3) by applying 19

one-class classifiers on the 255 one-class data sets. Prerequisite (4) is already satisfied, since we employ the AUC performance measure. The AUC has been the performance measure in the previous chapters as well. We refer to Chapter 2 for an explanation of the AUC.

The remainder of the current chapter is organised as follows. In Section 5.1 we provide relevant background on the one-class classifier selection problem and discuss the 'No Free Lunch' theorem. In Section 5.2 we describe and define the one-class classifier selection problem. In Section 5.3 we state the meta-learning approach. In Section 5.4 we present the one-class data sets. In Section 5.5 we discuss the use of preprocessing data sets. In Section 5.6 we give an overview of the meta-features that we use to characterise the one-class data sets. In Section 5.7 we present and discuss the results. They will be used as input for the next chapter. Finally, in Section 5.8, we complete the chapter with a summary.

5.1 No free lunch for one-class classification

In Chapter 4, five outlier-selection algorithms were applied to a variety of data sets. When we consider the average AUC performance of these algorithms, then SOS was the best performing algorithm. However, when we observe the performance on each particular data set, then SOS was sometimes outperformed by another algorithm (e.g., by LSOD on the Heart Disease data set). This observation is consistent with Song et al. (2012), who summarise the mixed performance of various binary classification algorithms (e.g., Decision Trees, Support Vector Machines, and Neural Networks) on various data sets (e.g., Iris, Thyrioid, and Sonar) as reported over the past two decades. From these mixed performances we may conclude that (1) no single algorithm can perform uniformly well over all data sets, and (2) there is no universally best algorithm.

Wolpert and Macready (1995) have formalised the above conclusions into the 'No Free Lunch' theorem, which states: '…if algorithm A outperforms algorithm B on some cost functions, then loosely speaking there must exist exactly as many other functions where B outperforms A.' Please note that, in practice, the value of the cost function depends on the data set at hand. No Free Lunch theorems have been proved and studied extensively for the settings of supervised learning, optimization, and search (Wolpert, 1996a,b, 2001). Although the theorem has not been explicitly studied for the setting of one-class

classification, we assume that (1) no single *one-class classifier* can perform uniformly well over all *one-class data sets*, and (2) there is no universally best *one-class classifier*.

5.2 The one-class classifier selection problem

In this section we introduce the one-class classifier selection problem. It is derived from the more general algorithm-selection problem (cf. Rice, 1976).

Rice (1976) presented a formal abstract model that we can use to explore the question: *With so many available one-class classifiers, which one is likely to perform best on my one-class data set?* The model is illustrated in Figure 5.1. The model consists of four components, presented counter-clockwise:

- the one-class data set space $\mathcal{P}(\mathcal{X})$, i.e., the power set of all real-world observations (cf. Chapter 2);
- the meta-feature space $\mathcal{F}$, which contains measurable characteristics of the one-class data sets;
- the one-class classifier space Φ; and
- the performance space $\mathcal{AUC}$, which represents the mapping of each one-class classifier to the AUC performance measure.

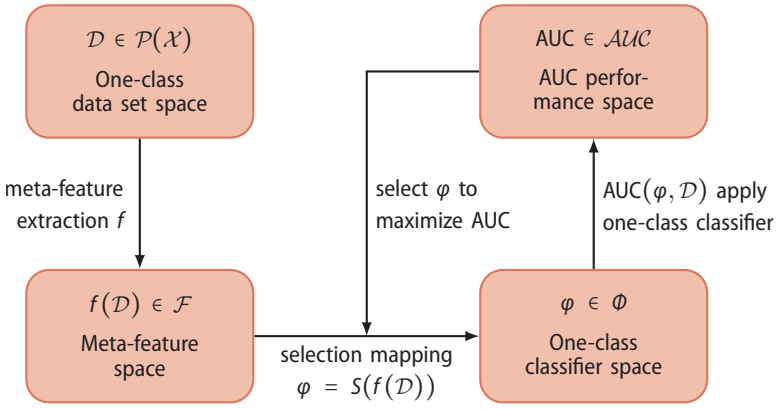


Figure 5.1 One-class classifier selection problem (adapted from Rice (1976)).

The one-class classifier selection problem can now be formally defined as follows.

Definition 5.1 (one-class classifier selection problem). *For a given one-class data set $\mathcal{D} \in \mathcal{P}(\mathcal{X})$, with meta-features $f(\mathcal{D}) \in \mathcal{F}$, find the selection mapping $S(f(\mathcal{D}))$ into one-class*

classifier space Φ*, such that the selected one-class classifier* $\phi \in \Phi$ *maximises the performance mapping* $\mathrm{AUC}(\phi, \mathcal{D}) \in \mathcal{AUC}$.

5.3 Meta-learning

In this section we describe the research field known as 'meta-learning'. First, we explain meta-learning by comparing it to base-learning (Subsection 5.3.1). Second, we describe the most-common use of meta-learning, namely binary classification (Subsection 5.3.2). Third, we discuss how meta-learning may be employed for the one-class classifier selection problem (Subsection 5.3.3).

5.3.1 Meta-learning compared to base learning

We explain meta-learning by comparing it to what is known as *base learning*. On the one hand, in base learning, each data point represents a real-world observation. So, the outlier-selection algorithms and one-class classifiers presented in the thesis are, in fact, base learners. Base learning is the normal or default type of learning in machine learning. (The term base learning is, however, rarely used in the machine learning literature, since it is only relevant with respect to meta-learning.)

On the other hand, in meta-learning, each data point represents a data set. So, each data point refers to an entire data set. The features of such data points are the characteristics of the data sets. We refer to them as meta-features so that they can be distinguished from the features of data points that represent real-world observations.

5.3.2 Meta-learning for binary classification

Meta-learning may be applied to a number of problems, such as classification, regression, constraint satisfaction, and sorting. In each application, exploiting meta-knowledge about the problem or the algorithm in order to improve the performance or selection of the algorithms plays a central role (see Vilalta and Drissi, 2002; Smith-Miles, 2008). Meta-learning is perhaps most often applied to the problem of binary classification, i.e., the algorithm-selection problem (Aha, 1992; Mitchie et al., 1994; King et al., 1995; Sohn, 1999; Peng et al., 2002; Kalousis et al., 2004; Ali and Smith, 2006; Brazdil et al., 2009; Jankowski et al., 2011; Song et al., 2012).

Rendell and Cho (1990) performed controlled experiments with artificial data sets, which showed that some data set characteristics (e.g., data set size and amount of noise) may drastically affect the performance of a binary classifier. That study used two algorithms only. Subsequently, Aha (1992) developed a general method that characterised the situations when distinct algorithms have a significant difference in performance. Artificial data sets were generated that closely resembled a real-world data set. So, although the number of data sets and algorithms were limited, both studies can be regarded as applying a meta-learning approach to select binary classifiers on artificial data sets.

The European research project StatLog (see Mitchie et al., 1994) studied meta-learning by relating the performance of classifiers to meta-features, in terms of rules. For example, they discovered a rule which stated that 'data sets with extreme distributions (*skew* > 1 and *kurtosis* > 7) and with many binary or categorical distributions (> 38%) tend to favour symbolic learning algorithms.' In their study, sixteen binary classifiers and twelve classification data sets were employed. The main conclusion of the project was consistent with the No Free Lunch Theorem, namely, that no algorithm uniformly outperforms all other algorithms on all data sets.

Developments in the field of meta-learning mainly focus on four aspects: (1) meta-features (Lee and Giraud-Carrier, 2008; Peng et al., 2002), (2) performance measures (Giraud-Carrier, 1998; Ali and Smith, 2006; Lee and Giraud-Carrier, 2011), and (3) algorithm recommendation frameworks (Song et al., 2012). For a more detailed overview of the development of meta-learning we refer to Vilalta and Drissi (2002), Smith-Miles (2008), and Jankowski et al. (2011).

5.3.3 Meta-learning for one-class classification

As far as we know, meta-learning for the problem of one-class classification is less studied than for the problem of binary classification. In Section 5.2 we introduced the one-class classifier selection problem, which resembled the algorithm-selection problem. Because of the resemblance, we adapt the meta-learning approach employed for the binary classification problem (cf. Subsection 5.3.2). To be precise, the one-class classifier selection problem (see Section 5.2) relates to the fourth definition of meta-learning in the survey paper by Vilalta and Drissi (2002). We adapt their definition as follows.

Definition 5.2 (Meta-learning for one-class classification). *Meta-learning for one-class classification is constructing meta-models that relate the meta-features of a one-class data set to the performance of a one-class classifier.*

One-class classification differs from binary classification in the assumption that data points from the anomalous class are rare. This assumption has consequences for the meta-features that we can employ to characterise the one-class data sets, as we will see in the next section.

5.4 Overview of one-class data sets

In this section we provide an overview of the one-class data sets. It is the first prerequisite for solving the one-class classifier selection problem. The one-class data sets are constructed from a variety of multi-class data sets, using the relabelling procedure from Subsection 2.6.1. All data sets, except the Delft pump data set (Ypma, 2001), come from the UCI Machine Learning Repository (Asuncion and Frank, 2010).

Table 5.1 lists 85 one-class data sets. (We explain in the next section how we get from 85 to 255 data one-class data sets.) The seven columns of the table represent the following. The first column, *Code*, represents the code of the one-class data set. The codes allow for easy reference when we present our results. The second column, *Multi-class data set*, represents the multi-class data set from which the one-class was constructed. The third column, *Normal class*, represents the name of class that was relabelled as the normal class. The fourth column, m, represents the dimensionality, or number of features. The fifth column, m_{PCA}, represents the dimensionality when the one-class data set has been preprocessed with the dimension reduction technique PCA. We explain this preprocessing procedure in Section 5.5. The sixth column, $\mathcal{C}_{\text{N}}$, and seventh column, $\mathcal{C}_{\text{A}}$, represent the number of normalities and anomalies in the one-class data set.

Before we compute the meta-features of these one-class data sets (see Section 5.6), we first apply three preprocessing procedures to the one-class data sets.

Table 5.1 Overview of the data sets.

Code	Multi-class data set	Normal class	m	m_{PCA}	$\mathcal{C}_N$	$\mathcal{C}_A$
501	Iris	setosa	4	2	50	100
502	Iris	versicolor	4	2	50	100
503	Iris	virginica	4	2	50	100
504	Breast	malignant	9	7	241	458
505	Breast	benign	9	7	458	241
506	Heart	diseased	13	3	137	160
507	Heart	healthy	13	3	160	137
508	Sonar	mines	60	17	111	97
509	Sonar	rocks	60	17	97	111
511	Biomed	healthy	5	2	127	67
512	Biomed	diseased	5	2	67	127
513	Imports	1	25	13	71	88
514	Arrhythmia	normal	278	38	237	183
515	Arrhythmia	all abnormal	278	36	183	237
516	Hepatitis	normal	19	2	123	32
517	Diabetes	present	8	3	500	268
518	Diabetes	absent	8	3	268	500
519	Ecoli	periplasm	7	5	52	284
541	Delft pump	AR app.	160	51	189	531
542	Delft pump	5x3	64	29	376	1,124
543	Delft pump	5x1	64	29	133	367
544	Delft pump	3x2	64	27	137	463
545	Delft pump	2x2	64	24	100	300
546	Delft pump	1x3	64	27	71	229
547	Delft pump	5x3 noisy	64	23	216	684
548	Delft pump	5x1 noisy	64	34	78	222
549	Delft pump	3x2 noisy	64	32	83	277
550	Delft pump	2x2 noisy	64	28	64	176
551	Delft pump	1x3 noisy	64	33	41	139
554	Cancer wpbc	non-ret	33	1	151	47
555	Cancer wpbc	ret	33	1	47	151
556	Spectf	0	44	23	95	254
557	Spectf	1	44	23	254	95
570	Colon	1	1,908	41	22	40
571	Colon	2	1,908	41	40	22
572	Leukemia	1	3,571	53	25	47
573	Leukemia	1	3,571	53	47	25
574	Metas	1	4,919	98	46	99
575	Metas	2	4,919	98	99	46
579	Balance-scale	left	4	4	288	337
580	Balance-scale	middle	4	4	49	576
581	Balance-scale	right	4	4	288	337

Table 5.1 (Continued)

Code	Multi-class data set	Normal class	m	m_{PCA}	$\mathcal{C}_N$	$\mathcal{C}_A$
582	Glass	building float	9	5	70	144
583	Glass	building nonfloat	9	5	76	138
584	Glass	vehicle float	9	5	17	197
585	Glass	containers	9	5	13	201
587	Glass	headlamps	9	5	29	185
588	Ionosphere	good	34	23	225	126
589	Ionosphere	bad	34	23	126	225
590	Liver	1	6	3	145	200
591	Liver	2	6	3	200	145
592	Thyriod	normal	21	12	93	3,679
593	Thyriod	hyperfunction	21	12	191	3,581
594	Thyriod	subnormal	21	12	3,488	284
595	Wine	1	13	1	59	119
596	Wine	2	13	1	71	107
597	Wine	3	13	1	48	130
598	Waveform	0	21	18	300	600
599	Waveform	1	21	18	300	600
600	Waveform	2	21	18	300	600
601	Vowel	0	10	8	48	480
602	Vowel	1	10	8	48	480
603	Vowel	2	10	8	48	480
604	Vowel	3	10	8	48	480
605	Vowel	4	10	8	48	480
606	Vowel	5	10	8	48	480
607	Vowel	6	10	8	48	480
608	Vowel	7	10	8	48	480
609	Vowel	8	10	8	48	480
610	Vowel	9	10	8	48	480
611	Vowel	10	10	8	48	480
612	Vehicle	van	18	1	199	647
613	Vehicle	saab	18	1	217	629
614	Vehicle	bus	18	1	218	628
615	Vehicle	opel	18	1	212	634
616	Survival	>5yr	3	3	225	81
617	Survival	<5yr	3	3	81	225
618	Housing	MEDV<35	13	2	458	48
619	Housing	MEDV>35	13	2	48	458
621	Satellite	Red soil	36	6	1,072	3,363
622	Satellite	Cotton crop	36	6	479	3,956
623	Satellite	Grey soil	36	6	961	3,474
624	Satellite	Damp grey soil	36	6	415	4,020
625	Satellite	Soil with vegetation stubble	36	6	470	3,965
626	Satellite	Very damp grey soil	36	6	1,038	3,397

5.5 Preprocessing one-class data sets

In this section we discuss the use of preprocessing. A preprocessing technique transforms the data set into a data set which is (more) appropriate for further processing. Obviously, this happens before the data set is processed by the one-class classifier. We describe three preprocessing techniques: (1) none (Subsection 5.5.1), (2) variance (Subsection 5.5.2), and (3) PCA (Subsection 5.5.3). Figure 5.2 illustrates the application of the three processing techniques on a toy one-class data set (first scatter plot) that has two features.

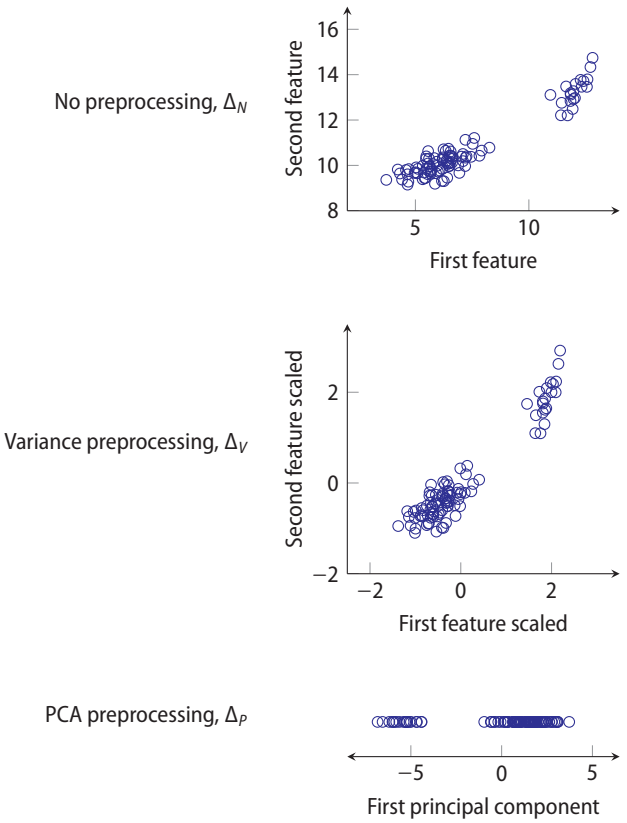


Figure 5.2 The one-class data sets are preprocessed by three techniques. **Top:** No preprocessing, Δ_N, leaves the data set intact. **Middle:** Variance preprocessing, Δ_V, rescales the dimensions. **Bottom:** PCA preprocessing, Δ_P, reduces the dimensionality of the one-class data set.

5.5.1 No preprocessing

The first preprocessing technique is actually no preprocessing at all. That is, all values in the one-class data set are left intact. The reason that we still consider this a preprocessing technique is for presentation purposes only. We call this preprocessing technique 'none' and denote it by Δ_N. We include this preprocessing technique in our experiments because it will serve as a baseline for the other two preprocessing techniques.

In the top scatter plot of Figure 5.2, no preprocessing is applied; these are the original values of the toy one-class data set. The other middle and bottom scatter plots correspond to the other two preprocessing techniques.

5.5.2 Variance preprocessing

Because the features of the data points in a one-class data set may have different domains, and because different features may be expressed in different units, each feature may have a different scale.

Many of the one-class classifiers employ the Euclidean distance measure for determining the distance between the data points. The scale of the feature directly influences the distance. On the one hand, features with a rather large scale are considered to be quite important, because they largely determine the distance between data points. On the other hand, features with a rather small scale do not matter any more; they have no influence on the distance. This may result in a 'badly' scaled one-class data set that hampers the performance of the one-class classifier.

To overcome this scaling problem, we apply a preprocessing step that is called the variance preprocessing, which is denoted by Δ_V. Each feature is rescaled separately in such a way that the variance of the values equals 1. Moreover, the values are shifted such that the mean of each feature becomes 0. (This is similar to computing a standard score, with the exception that with standard scores, the standard deviation equals 1). The middle scatter plot in Figure 5.2 shows the result of applying this preprocessing technique to the toy data set of the first scatter plot. Here, each feature is equally important.

5.5.3 PCA preprocessing

When the data points in a one-class data set have many features, we say that the one-class data set has a high dimensionality. For example, a data point representing a digital

image has one feature per pixel. A collection of images that are 100 by 100 pixels results in a 10,000-dimensional data set. The performance of a one-class classifier may be hampered by such a high dimensionality, which constitutes a problem. This problem is known as the 'curse of dimensionality' (see Marimont and Shapiro, 1979). The curse of dimensionality occurs because, as the number of features increase, the volume of the data set increases exponentially. Stated differently, the Euclidean distances between the data points increase, causing the relative differences between distances to decrease.

The curse of dimensionality can be mitigated by applying a dimensionality reduction technique to the one-class data set. The third preprocessing technique that we employ is Principal Component Analysis (PCA) (Pearson, 1901). The bottom scatter plot in Figure 5.2 shows the result of applying this preprocessing technique to the toy data set of the first scatter plot. We note that the preprocessed data set has only one feature.

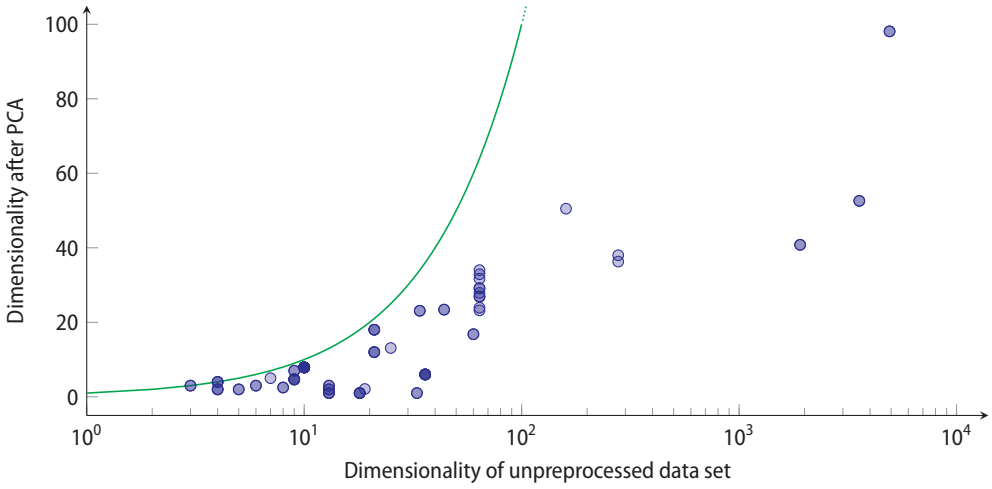


Figure 5.3 PCA.

PCA finds a linear transformation of the original data set such that the new features are linearly separated. The new features are also called the principal components. The first principal component usually explains most of the variation in the feature values. PCA is a *parametric* dimensionality reduction technique, which is required because we need to re-apply the same linear transformation to the test data set.

After PCA has transformed the original one-class data set into a new data set, we keep only the first few numbers of the principal components, viz. those that account for at least 95% of the total variance. In our toy data set, this results in the second principal component to be discarded, since at least 95% of the total variance is explained by

the first principal component. The discarding of principal components ensures that the dimensionality of the data set is reduced, while maintaining the overall structure of the original data set.

5.5.4 From 85 to 255 one-class data sets

We consider a preprocessed data set to be a different data set. From a theoretical point of view, one may argue that a data point, even though it has been preprocessed, still refers to the same real-world object, and that the one-class data set remains the same problem. However, we take a more practical point of view, and argue as follows. Because the values presented to the one-class classifier are different, the data set is a different problem to be solved. In any case, it is convenient to call them different data sets. This brings us to $3 \times 85 = 255$ data sets in total.

Once a one-class data set has been preprocessed, its code is appended with a postfix. For no preprocessing the postfix is 'N', for variance preprocessing the postfix is 'V', and for PCA preprocessing the postfix is 'P'. For example, the code '501P' refers to data set 501 that has been preprocessed by PCA.

5.6 Meta-features

In this section we describe the meta-features that we employ to characterise a one-class data set $\mathcal{D}$, which is the second prerequisite for solving the one-class classifier selection problem. Formally, a meta-feature is defined as follows.

Definition 5.3 (Meta-feature). *A meta-feature is a mapping $\mu : \mathcal{D} \to \mathbb{R}$ and $\mu(\mathcal{D})$ is said to be a characteristic of one-class data set $\mathcal{D} \in \mathcal{P}(\mathcal{X})$, where $\mathcal{P}(\mathcal{X})$ is the space of one-class data sets (cf. Section 5.2).*

So, a meta-feature can be seen as a function that takes a one-class data set as input and produces a scalar value as output. A collection of meta-features therefore produces a vector of values. In other words, each one-class data set is represented by a feature vector. (See Subsection 2.2.1 for a more detailed explanation of feature vectors.) Each feature vector is to be accompanied by a label. For example, the label can denote the one-class classifier that performed best on that one-class data set. (Other types of labels are discussed in Chapter 6.) We note that a combination of feature vectors and labels is, in fact, a labelled meta-data set.

If the values of the meta-features correlate with the labels, then meta-rules can be formulated (cf. the example rule found by the StatLog project mentioned in Subsection 5.3.2). Such meta-rules can then be used to recommend a domain expert which one-class classifier to should be applied on a new one-class data set. Here we note that constructing meta-rules is the goal of Chapter 6. We recall that the goal in the current chapter is to find appropriate meta-features.

Ideally, we would only employ meta-features that can be computed from the normalities, because anomalies are assumed to be rare (cf. Chapter 1). In Subsection 5.6.1 we discuss to what extent this assumption affects the quality of the meta-feature values. We employ both (1) novel meta-features and (2) existing meta-features from the scientific literature. The meta-features are grouped into six categories.

1. Elementary meta-features (Subsection 5.6.2)
2. Statistical meta-features (Subsection 5.6.3)
3. Decision tree-based meta-features (Subsection 5.6.4)
4. Information-theory-based meta-features (Subsection 5.6.5)
5. Euclidean-distance-based meta-features (Subsection 5.6.6)
6. Miscellaneous meta-features (Subsection 5.6.7)

The meta-features were chosen because of their (1) usage within the research area of meta-learning and (2) effectiveness to characterise data sets.

5.6.1 Computing meta-features

In the thesis, as mentioned in Chapter 1, anomalies are assumed to be rare. The assumption implied that in our comparative experiment, the one-class classifiers were trained on the normalities only (cf. Tax, 2001). Here we remark that in case the data set contains both sufficient normalities and anomalies, the operator may be better supported by employing a binary classifier instead of a one-class classifier.

In order to maintain the assumption, the meta-learning approach should employ meta-features that are based on the normalities only. This differs from meta-learning for binary classification, where meta-features are based on both classes. As a result, we cannot straightforwardly employ all the meta-features proposed in the literature.

Our implementation of the assumption that anomalies are rare may be too strict in order to solve the one-class classifier selection problem. Between employing (1) all anomalies

(as is the case in meta-learning for binary classification) and (2) no anomalies at all (as is the case in meta-learning one-class classification), there exist a range of fractions of anomalies. Indeed, the term 'rare' is vaguely defined. So, in order to solve the one-class classifier selection problem, we start relaxing the implementation of employing no anomalies at all. Now the following question arises: *how many anomalies should we employ to compute the meta-features of a one-class data set?* On the one hand, we aim to employ as few anomalies as possible, in order to maintain a realistic situation of rare anomalies. On the other hand, we expect that employing more anomalies increases the reliability of the meta-features and thus increases the extent by which the one-class classifier selection problem may be solved using a meta-learning approach. In Chapter 6 we investigate the balance between (1) a realistic situation and (2) reliable meta-features.

Below we describe the five-step procedure for computing a meta-feature of a one-class data set $\mathcal{D}$. The Iris flower data set, introduced in Subsection 2.6.1, has 50 data points from each of the three classes 'setosa', 'versicolor', and 'virginica'. Applying the relabelling procedure to the Iris flower data set, three one-class data sets are constructed (cf. Subsection 2.6.1). The three data sets are listed in Table 5.1, and have the codes 501, 502, and 503, respectively. For brevity, we refer to specific one-class data sets by their code instead of by their name (cf. Tax, 2012). We use data set 501 to explain the five-step procedure.

The first step is to initialise the data set, and count the number of normalities and anomalies. Data set 501 contains 150 data points in total, of which 50 normalities and 100 anomalies (i.e., $|\mathcal{C}_{\mathrm{N}}| = 50$, $|\mathcal{C}_{\mathrm{A}}| = 100$).

The second step is to apply stratified ten-fold cross validation (cf. Subsection 2.8.3). Cross validation increases the reliability of the value of the meta-feature and prevents over-fitting (cf. Kohavi, 1995). In each of the ten iterations of the cross-validation, we have 90% of the original data points available. For data set 501, this corresponds to 45 normalities and 90 anomalies.

The third step is to apply one of the three preprocessing methods discussed in Section 5.5.

The fourth step is to vary the number of anomalies. Initially, we let the data set contain only normalities, i.e., $\mathcal{D} = \mathcal{C}_{\mathrm{N}}$. Subsequently, from the set of available anomalies $\mathcal{C}_{\mathrm{A}}$, we add a number of anomalies that equals a fraction λ of the number of normalities. More precisely, we let $|\mathcal{C}_{\mathrm{A}}|$ be equal to $\lambda |\mathcal{C}_{\mathrm{N}}|$ for each $\lambda \in \{0.0, 0.1, 0.2\}$. When $\lambda = 0$, no anomalies are present in the data set. When $\lambda = 1$, the number of anomalies is equal to

the number of normalities, which corresponds more to a binary meta-learning setting. The anomalies are chosen randomly from the set of available anomalies. For data set 501, $\lambda = 0.2$, corresponds to randomly selecting nine anomalies from the 90 available anomalies. Because the anomalies are chosen randomly, we repeat this step ten times. Here we remark, for completeness, that we aim for a low value of λ that still produces reliable meta-features.

The fifth step is actually to apply the meta-feature μ to the one-class data set $\mathcal{D}$ that was produced by the steps two, three, and four. Because of the ten-fold cross validation from step two, and the randomly selected anomalies from step four, this step is performed 100 times, and thus produces 100 values. The final value of the meta-feature of the original one-class data set is the average of the 100 values.

5.6.2 Elementary meta-features

The first category of meta-features are the elementary meta-features. These meta-features are employed by most references (see King et al., 1995; Sohn, 1999; Köpf, Taylor, and Keller, 2000; Ali and Smith, 2006). Below we list five of them. They are defined as follows.

1. Number of normalities

This meta-feature returns the number of normalities in the data set.

$$\mu_{\text{number-of-normalities}}(\mathcal{D}) = |\mathcal{C}_{\text{N}}|\,, \tag{5.1}$$

where $\mathcal{C}_{\text{N}}$ is the set of normalities.

2. Number of anomalies

This meta-feature returns the number of anomalies in the data set.

$$\mu_{\text{number-of-anomalies}}(\mathcal{D}) = |\mathcal{C}_{\text{A}}|\,, \tag{5.2}$$

where $\mathcal{C}_{\text{A}}$ is the set of anomalies.

3. Dimensionality

This meta-feature is also known as the number of features or attributes in the data set.

$$\mu_{\text{dimensionality}}(\mathcal{D}) = m, \tag{5.3}$$

where m is the number of features of the one-class data set.

4 & 5. Ratio of binary and categorical features

We assume that each feature within a one-class data set belongs to one of three types: (1) binary, (2) categorical, and (3) continuous. We define the following two meta-features, that measure the ratio of the binary and categorical features.

$$\mu_{\text{binary-features}}(\mathcal{D}) = \frac{m_b}{m}, \tag{5.4}$$

$$\mu_{\text{categorical-features}}(\mathcal{D}) = \frac{m_c}{m}, \tag{5.5}$$

where m_b, and m_c denote the number of binary and categorical features, respectively.

5.6.3 Statistical meta-features

The second category of meta-features is based on descriptive statistics (see Sohn, 1999; Köpf et al., 2000; Ali and Smith, 2006). Below we list eleven of them. (Please note we number the meta-features throughout.)

6. Mean

The meta-feature mean computes for each feature j in the one-class data set the mean of the values, and returns the average of those means.

$$\mu_{\text{mean}}(\mathcal{D}) = \frac{1}{nm} \cdot \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij}, \tag{5.6}$$

where x_{ij} is the j^{th} feature value of data point $\mathbf{x}_i$.

7, 8 & 9. Standard deviation

The meta-feature standard deviation computes for each feature j in the one-class data set the standard deviation of the values, and returns the average of all of them.

$$\mu_{\text{std-mean}}(\mathcal{D}) = \frac{1}{m} \cdot \sum_{j=1}^{m} \sqrt{\frac{\sum_{i=1}^{n} x_{ij}^2}{n} - \left(\frac{\sum_{i=1}^{n} x_{ij}}{n}\right)^2}. \tag{5.7}$$

We also compute the maximum standard deviation,

$$\mu_{\text{std-max}}(\mathcal{D}) = \max_{j=1}^{m} \sqrt{\frac{\sum_{i=1}^{n} x_{ij}^2}{n} - \left(\frac{\sum_{i=1}^{n} x_{ij}}{n}\right)^2}, \tag{5.8}$$

and the standard deviation of the standard deviations of each feature,

$$\mu_{\text{std-std}}(\mathcal{D}) = \sqrt{\frac{\sum_{i=1}^{m} \mathbf{s}_i^2}{m} - \left(\frac{\sum_{i=1}^{m} \mathbf{s}_i}{m}\right)^2}, \tag{5.9}$$

where $\mathbf{s}$ is a vector of length m that contains the standard deviation of each feature.

10. Normality ratio

This meta-feature computes the ratio of normality of the one-class data set. For each feature in the data set, it performs the Lilliefors test that tests the hypothesis whether the values could have been generated from a Normal distribution. The Lilliefors test is a 2-sided goodness-of-fit test suitable when a fully-specified null distribution is unknown and its parameters must be estimated (Lilliefors, 1967). This meta-feature ranges from 0 (no features are Normally distributed) to 1 (all features are Normally distributed).

$$\mu_{\text{normality-ratio}}(\mathcal{D}) = \frac{1}{m} \cdot \sum_{j=1}^{m} \text{lillie}\Big(\big[x_{1j}, x_{2j}, x_{3j}, \ldots, x_{nj}\big]\Big). \tag{5.10}$$

11 & 12. Kurtosis

The meta-feature kurtosis computes for each feature j in the one-class data set the kurtosis of the values, and returns the average of them. Kurtosis is a measure of 'peakedness' of a distribution of values. For example, when the values of one feature are distributed normally, then the kurtosis is 3.

$$\mu_{\text{kurtosis-mean}}(\mathcal{D}) = \frac{1}{m} \cdot \sum_{j=1}^{m} \text{kurt}\Big(\big[x_{1j}, x_{2j}, x_{3j}, \ldots, x_{nj}\big]\Big). \tag{5.11}$$

Besides the average kurtosis, we measure the maximum kurtosis over all the features in the one-class data set.

$$\mu_{\text{kurtosis-max}}(\mathcal{D}) = \max_{j=1}^{m} \text{kurt}\Big(\big[x_{1j}, x_{2j}, x_{3j}, \ldots, x_{nj}\big]\Big). \tag{5.12}$$

13. Skewness

The meta-feature skewness computes for each feature j in the one-class data set the absolute skewness of the values, and returns the maximum of them. Skewness is a measure of asymmetry of the distribution of values. Because values may be skewed

to the left or right, we take the absolute skewness.

$$\mu_{\text{skewness}}(\mathcal{D}) = \max_{j=1}^{m} ||\text{skew}\Big(\big[x_{1j}, x_{2j}, x_{3j}, \ldots, x_{nj}\big]\Big)||. \tag{5.13}$$

14, 15 & 16. Eigenvalues

To compute the meta-features concerning eigenvalues, we first compute the sample covariance matrix of the one-class data set. Then we compute the eigenvalues of the covariance matrix. The solutions $\lambda_1, \lambda_2, \lambda_3, \ldots, \lambda_r$ are called the eigenvalues, where $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_r$, and r is the rank of the covariance matrix. The largest and smallest eigenvalues (cf. Ali and Smith, 2006) are

$$\mu_{\text{eig-max}}(\mathcal{D}) = \lambda_1 \,, \tag{5.14}$$

$$\mu_{\text{eig-min}}(\mathcal{D}) = \lambda_r \,. \tag{5.15}$$

The relative importance of the largest eigenvalue (cf. Köpf et al., 2000) reads

$$\mu_{\text{eig-max-rel}}(\mathcal{D}) = \lambda_1 / \sum_{i=1}^{r} \lambda_i. \tag{5.16}$$

Here, we note that the eigenvalues are non-negative, so the sum is strictly positive.

5.6.4 Decision tree-based meta-features

The third category of meta-features is based on decision trees. A decision tree is structure where leaves represent labels $\{\text{anomaly, normality}\}$ and branches represent conjunctions of features that lead to those labels (Quinlan, 1986). Figure 5.4 shows a decision tree that is induced from data set 501N. For example, a data point with feature vector $\mathbf{x} = [5.1, 3.5, 1.4, 0.2]$ is classified as a normality. We note that the absence of the second feature x_2 may indicate that it does not play an important role (see Peng et al., 2002).

The algorithm that constructs the decision tree works top-down by choosing at each step, the feature with the highest Gini index of diversity (Mingers, 1989). The Gini index of diversity is a measure of how often a randomly chosen data point from the data set would be incorrectly labelled if it were randomly labelled according to the distribution of labels in the subset.

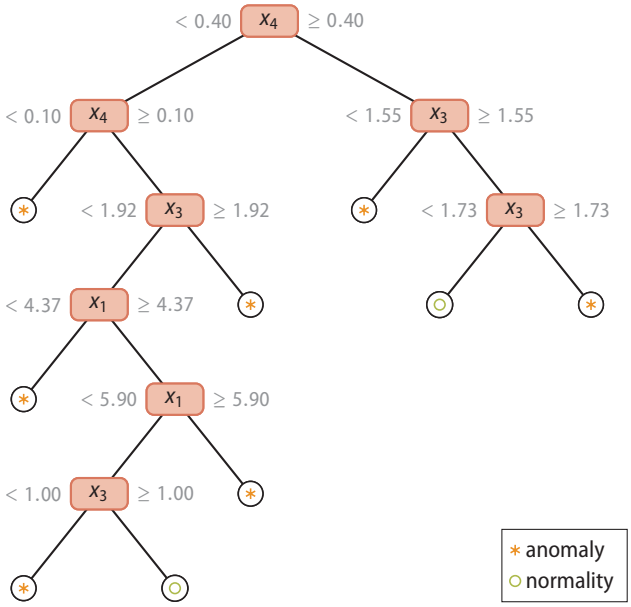


Figure 5.4 A decision tree induced from data set 501. From the decision tree we calculate several properties that serve as meta-features.

The idea is that the properties of the tree (e.g., shape and complexity) are a predictor of the complexity of the one-class data set (cf. Bensusan, Giraud-Carrier, and Kennedy, 2000; Peng et al., 2002). A larger tree, for example, may indicate that the one-class data set is more complex and that it may degrade the performance of certain one-class classifiers. Bensusan et al. (2000) found that the depth and the number of leaf nodes depend on (1) the amount of noise, (2) the number of irrelevant features, and (3) the class distribution.

We assume, similar to Bensusan et al. (2000), that the decision trees are not pruned, that is, no branches are cut. We remark that the decision tree is thus constructed anew for each time the meta-feature is computed, i.e., 100 times. Twelve decision tree-based meta-features are defined below.

17. Number of nodes

The number of nodes in the decision tree is defined as follows:

$$\mu_{\text{number-of-nodes}}(\mathcal{D}) = |\mathcal{T}|, \tag{5.17}$$

where $\mathcal{T}$ denotes the decision tree induced from one-class data set.

18. Number of leaves

The number of leaves in the decision tree is defined as follows:

$$\mu_{\text{number-of-leaves}}(\mathcal{D}) = |\mathcal{L}|, \tag{5.18}$$

where $\mathcal{L}$ is the set of leaf nodes.

19. Nodes per class

The number of nodes per class is defined as follows:

$$\mu_{\text{nodes-per-class}}(\mathcal{D}) = \frac{|\mathcal{T}|}{2}. \tag{5.19}$$

20. Nodes per data point

The meta-feature nodes per data point computes the ratio of the number of tree nodes to the number of data points.

$$\mu_{\text{nodes-per-data-point}}(\mathcal{D}) = \frac{|\mathcal{T}|}{n}, \tag{5.20}$$

where $|\mathcal{T}|$ denotes the number of nodes in the tree.

21. Depth

The meta-feature depth measures the longest path from the root node to a leaf node. The length of a path is the measured by the number of nodes it contains. The depth indicates how difficult it is to represent the one-class data set by a decision tree.

$$\mu_{\text{depth}}(\mathcal{D}) = \max |P|, \ \forall P \in \mathcal{P}(\mathcal{T}), \tag{5.21}$$

were $\mathcal{P}(\mathcal{T})$ denotes the set of all paths in $\mathcal{T}$, and $|P|$ denotes the number of nodes in the path. Paths are assumed to start at the root node. The depth of the decision tree in Figure 5.4 is 7.

22. Width

The meta-feature width computes the width of the decision tree, which is defined as the maximum number of nodes in a level:

$$\mu_{\text{width}}(\mathcal{D}) = \max \left| \mathcal{T} \text{ s.t. level}(N) = i \mid \forall i \in \left\{1, 2, \ldots, \mu_{depth}\right\}\right. . \tag{5.22}$$

The width of the decision tree in Figure 5.4 is 4.

23. Shape

The meta-feature shape is a function of the probabilities of arriving at the various leaf nodes given a random walk down the tree. The probability of arriving at node N_i among the m sibling nodes from the ancestor N_a is given by $p(N_i) = p(N_a)/m$ where $p(N_a) = 1$ if it is the root node. The shape of the decision tree with is then measured using the probability of arriving at the leaf nodes $p(L)$ by the following equation.

$$\mu_{\text{shape}}(\mathcal{D}) = -\sum_{L \in \mathcal{L}} p(L) \log_2 p(L). \tag{5.23}$$

24. Number of repeated nodes

The meta-feature number of repeated nodes measures how many features are represented by more than one node. A high value indicates the need for feature re-description.

$$\mu_{\text{number-of-repeated-nodes}}(\mathcal{D}) = \sum_{i=1}^{m} \mathbb{1}\left\{\left|\left[\mathcal{T} \middle| f(N) = i\right]\right| > 1\right\}, \tag{5.24}$$

where $f(N)$ indicates the feature that node N is concerned with.

25. Leaf corroboration

The meta-feature leaf corroboration computes the mean support of each leaf node. The support of a leaf node is defined as the number of data points (both normalities and anomalies) terminating in that leaf node.

$$\mu_{\text{leaf-corroboration}}(\mathcal{D}) = \frac{1}{|\mathcal{L}|} \sum_{L \in \mathcal{L}} s(L), \tag{5.25}$$

where $\mathcal{L}$ is the set of leaf nodes and $s(L)$ denotes the support of leaf node L.

26. Impurity

The meta-feature impurity computes the impurity of the tree, which is defined as the average Gini's Diversity Index at each node in the tree:

$$\mu_{\text{impurity}}(\mathcal{D}) = \frac{1}{|\mathcal{T}|} \sum_{\mathcal{T}} p_a^2(N) p_n^2(N), \tag{5.26}$$

where $p_a^2(N)$ and $p_n^2(N)$ denote the probability that a data point will be classified as anomaly and normality, respectively.

27. Mean error

The meta-feature mean error computes the average error of the decision tree, where error is defined as the probability that a data point will be mis-classified.

$$\mu_{\text{mean-error}}(\mathcal{D}) = \frac{1}{|\mathcal{L}|} \sum_{L \in \mathcal{L}} e(L), \tag{5.27}$$

where $e(L)$ denotes the error of leaf node L.

28. Total error

The meta-feature total error computes the total error of the decision tree, where error is defined as the probability that a data point will be mis-classified.

$$\mu_{\text{total-error}}(\mathcal{D}) = \sum_{L \in \mathcal{L}} e(L). \tag{5.28}$$

5.6.5 Information-theory-based meta-features

29. Correlation between meta-features

The meta-feature correlation between meta-features computes the average correlation between all pairs of meta-features. More precisely, it is the pairwise Pearson's linear correlation coefficient between each pair of meta-features in the one-class data set, and is mathematically defined as follows:

$$\mu_{\text{correlation}}(\mathcal{D}) = \frac{2}{m(m-1)} \cdot \sum_{i=1}^{m} \sum_{j=i+1}^{m} \frac{\text{cov}(\mathbf{f}_i, \mathbf{f}_j)}{\sigma_{\mathbf{f}_i} \sigma_{\mathbf{f}_j}}, \tag{5.29}$$

where $\text{cov}(\mathbf{f}_i, \mathbf{f}_j)$ is the covariance between meta-feature i and j and $\sigma_{\mathbf{f}_i}$ is the variance of meta-feature i.

5.6.6 Euclidean-distance-based meta-features

The fifth category of meta-features calculates statistics of the Euclidean distances between all pairs of data points in the data set. The rationale is that many of the one-class classifiers use the Euclidean distances as the dissimilarity measure. To the best of our knowledge, this category of meta-features has not yet been explored.

Let $\mathbf{e}$ be a vector of length $n(n-1)/2$ that contains the Euclidean distances between all pairs of data points:

$$\mathbf{e} = \left[\sqrt{\sum_{k=1}^{m} \left(x_{jk} - x_{ik} \right)^2} \;\middle|\; i \in \{1, \cdots, n\}, j \in \{1, \cdots, i-1\} \right].$$

The vector $\mathbf{e}$ can also be thought of as the flattened upper triangle of the dissimilarity matrix $\mathbf{D}$ as discussed in Subsection 4.2.1.

We compute five statistics of the Euclidean distance vector $\mathbf{e}$: (1) the mean, (2) the median, (3) the standard deviation, (4) the skewness, and (5) the kurtosis. These five statistics serve as five meta-features to the one-class data set $\mathcal{D}$ and are defined as follows.

30. Mean of Euclidean distances

The mean of the Euclidean distances is defined as follows:

$$\mu_{\text{euclidean-mean}}(\mathcal{D}) = \text{mean}(e). \tag{5.30}$$

31. Median of Euclidean distances

The median of the Euclidean distances is defined as follows:

$$\mu_{\text{euclidean-median}}(\mathcal{D}) = \text{median}(e). \tag{5.31}$$

32. Standard deviation of Euclidean distances

The standard deviation of the Euclidean distances is defined as follows:

$$\mu_{\text{euclidean-std}}(\mathcal{D}) = \text{std}(e). \tag{5.32}$$

33. Kurtosis of Euclidean distances

The kurtosis of the Euclidean distances is defined as follows:

$$\mu_{\text{euclidean-kurtosis}}(\mathcal{D}) = \text{kurt}(e). \tag{5.33}$$

34. Skewness of Euclidean distances

The skewness of the Euclidean distances is defined as follows:

$$\mu_{\text{euclidean-skewness}}(\mathcal{D}) = \text{skew}(e)\,. \tag{5.34}$$

5.6.7 Miscellaneous meta-features

The sixth category contains two miscellaneous meta-features that do not belong to any of the other categories: hardness and variance curve.

35. Hardness

The meta-feature hardness is a defined as the misclassification rate of a 1-Nearest Neighbour classifier.

$$\mu_{\text{hardness}}(\mathcal{D}) = \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}\left\{y_i \neq l_i\right\}, \tag{5.35}$$

where $\mathbb{1}\left\{y_i \neq l_i\right\}$ is 0 when data point i has been correctly classified, and 1 otherwise. It is known as a landmarking meta-feature because by using a fast algorithm, such as the 1-Nearest Neighbour classifier, we may get an idea of the difficulty, or hardness, of the one-class data set. When there are no anomalies present in the one-class data set, we randomly insert them, making it possible to use this meta-feature.

36. Variance curve

The meta-feature variance curve intends to inform us on how much difference in variance there is between each feature in the one-class data set.

$$\mu_{\text{variance-curve}}(\mathcal{D}) = \frac{\sum_{i=1}^{m} \sigma_{\mathbf{f}_i}}{m \cdot \max_i \sigma_{\mathbf{f}_i}}, \tag{5.36}$$

where $\sigma_{\mathbf{f}_i}$ denotes the variance of feature i and m denotes the number of features. The meta-feature variance curve is defined between 0 and 1. Its value is 1 when all features in the one-class data set have the same variance. The lower the value is, the more difference there is in the variance between features. A high difference in variance between features may indicate badly scaled features, rendering the Euclidean-distance measure inadequate for one-class classification.

5.7 Results and discussion

In this section we present the main result of the chapter, which are the 36 meta-features. To investigate the 36 meta-features more closely, we visualise the distribution of the values they produce for the 255 one-class data sets with density plots. A density plot shows how the values are distributed across the one-class data sets and may reveal whether the values are suitable for the task of adequately predicting a one-class classifier for a given data set (so the x-axis is the value and the y-axis is the density). The density is estimated using Kernel Density Estimation (KDE). We may see negative values because KDE applies some smoothing to the values. Figure 5.5 shows 36 density plots, one for each meta-feature. We plot the distribution for three values of λ, namely, 0.0, 0.1, and 0.2. This allows us to investigate whether the distribution of meta-feature values changes as the number of anomalies is increased.

When the meta-feature value is 0 for all one-class data sets, the density plot displays a flat line. This is the case for the meta-features 'number-of-anomalies' and 'impurity', for $\lambda = 0.0$.

For some meta-features, the values can become disproportionally high. This results in a badly scaled meta-feature which makes meta-learning more difficult. Consider, for example, the meta-features 'dimensionality', 'std-std', and 'eig-min'. When we look at their distributions in Figure 5.5 then we can see that the majority of the values are very low, and that there are a few outlying values. (We do not need any outlier-selection algorithm to select those outlying values.) In order to model appropriately the meta-feature on the entire range of values, we apply the natural logarithm to each value of the meta-feature.

Table 5.2 provides an overview of all meta-features and indicates on which meta-features we apply the log transformation.

Figure 5.6 shows, just like Figure 5.5, 36 distributions, but then for each log transformed meta-feature. By looking at the 36 distributions, we can see that all meta-features are well-behaved. We note that we only apply the log transformation to those meta-features that need that transformation.

Figure 5.7 shows a two-dimensional scatter plot of the 255 one-class data sets. Each marker in the scatter plot represents a one-class data set. The code of data set can be found next to each marker. The shape of the marker, i.e., a circle, a square, or triangle,

Table 5.2 Overview of the 36 meta-features used in our experiment. The column 'Apply log' indicates whether we apply the log transformation to the values of that meta-feature.

Equation	Meta-feature name	Apply log	Equation	Meta-feature name	Apply log
5.1	number-of-normalities	✓	5.19	nodes-per-class	
5.2	number-of-anomalies	✓	5.20	nodes-per-data-point	
5.3	dimensionality	✓	5.21	depth	✓
5.4	binary-features		5.22	width	
5.5	categorical-features		5.23	shape	
5.6	mean	✓	5.24	number-of-repeated-nodes	✓
5.7	std-mean	✓	5.25	leaf-corroboration	✓
5.8	std-max	✓	5.26	impurity	✓
5.9	std-std	✓	5.27	mean-error	
5.10	normality-ratio		5.28	total-error	✓
5.11	kurtosis-mean	✓	5.29	correlation	
5.12	kurtosis-max	✓	5.30	euclidean-mean	✓
5.13	skewness	✓	5.31	euclidean-median	✓
5.14	eig-max	✓	5.32	euclidean-std	✓
5.15	eig-min	✓	5.33	euclidean-kurtosis	✓
5.16	eig-max-rel		5.34	euclidean-skewness	
5.17	number-of-nodes	✓	5.35	hardness	
5.18	number-of-leaves	✓	5.36	variance-curve	

indicates whether the data set is preprocessed with None, PCA, or Variance technique, respectively.

The position of each marker is based on the 36 meta-feature values of the corresponding one-class data set. Each one-class data set is therefore represented by a vector with a length of 36. So, we actually have a meta-data set of 255 instances and with a dimensionality of 36. To visualise the 36-dimensional meta-data set, we employ t-SNE, which is a non-linear dimensionality reduction technique (van der Maaten and Hinton, 2008). t-SNE reduces the dimensionality of the meta-data set to two, while preserving the local structure. As a result, one-class classifiers that have similar meta-features will appear close together in the scatter plot. Figure 5.7 is thus a visualisation of one of the meta-data sets that we will use in Chapter 6.

The meta-features used in Figure 5.7 are standardised and have the log transformation applied to them (as indicated in Table 5.2). Moreover, λ is set to 0.1, so the meta-features are based on one-class data sets with some anomalies.

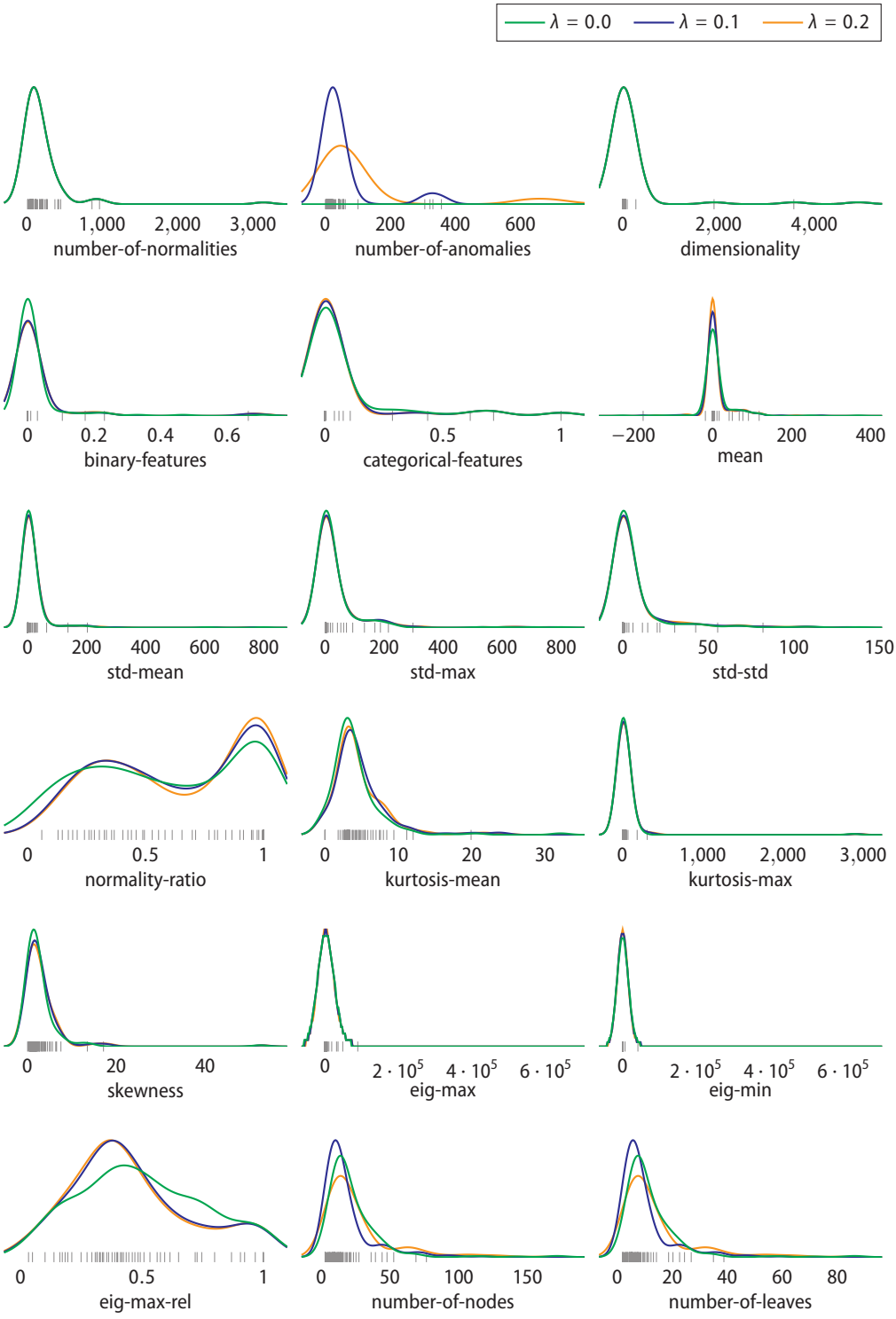


Figure 5.5 Density plots of the raw meta-features for $\lambda = 0.0, 0.1$, and 0.2.

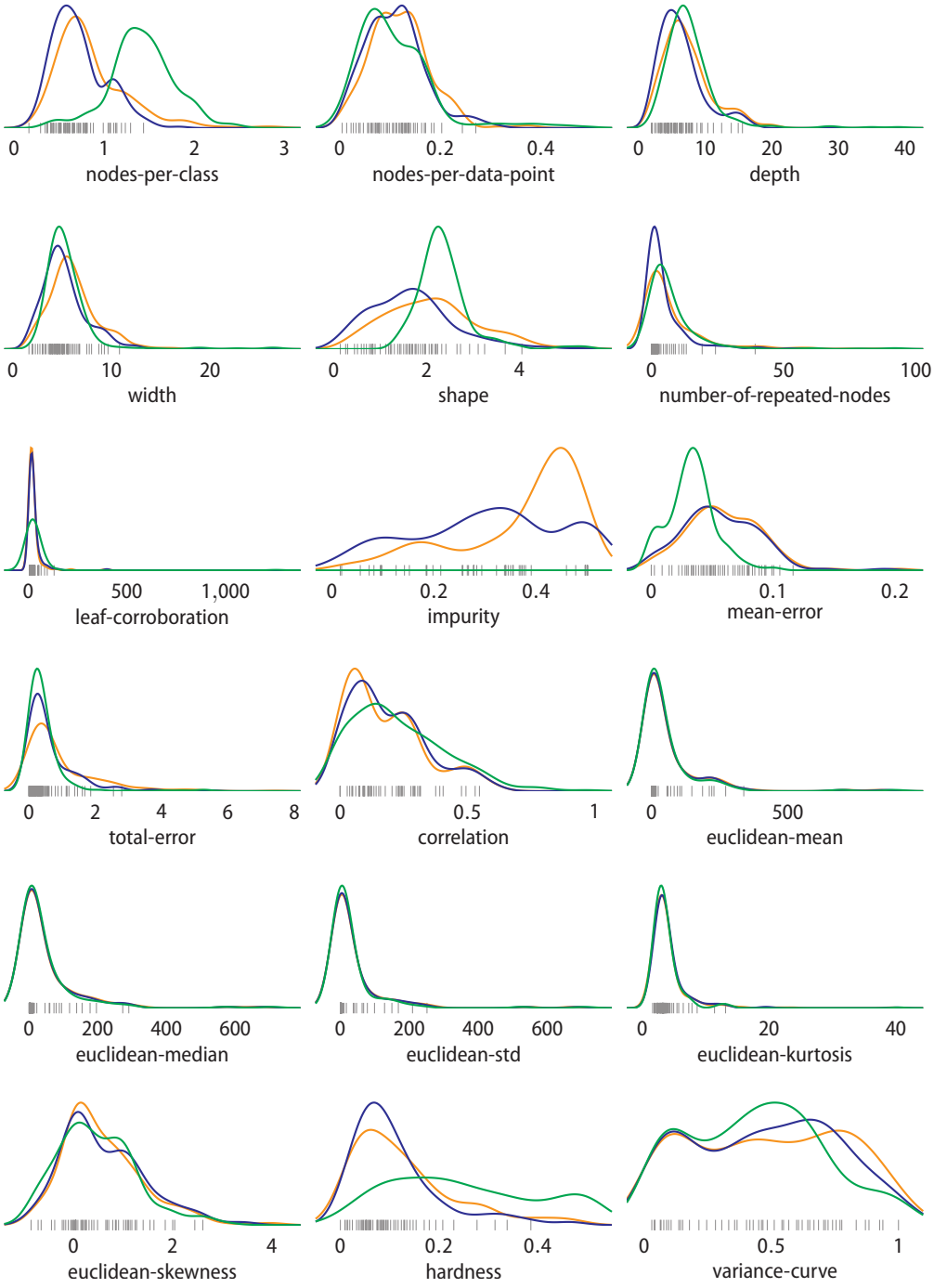


Figure 5.5 (Continued)

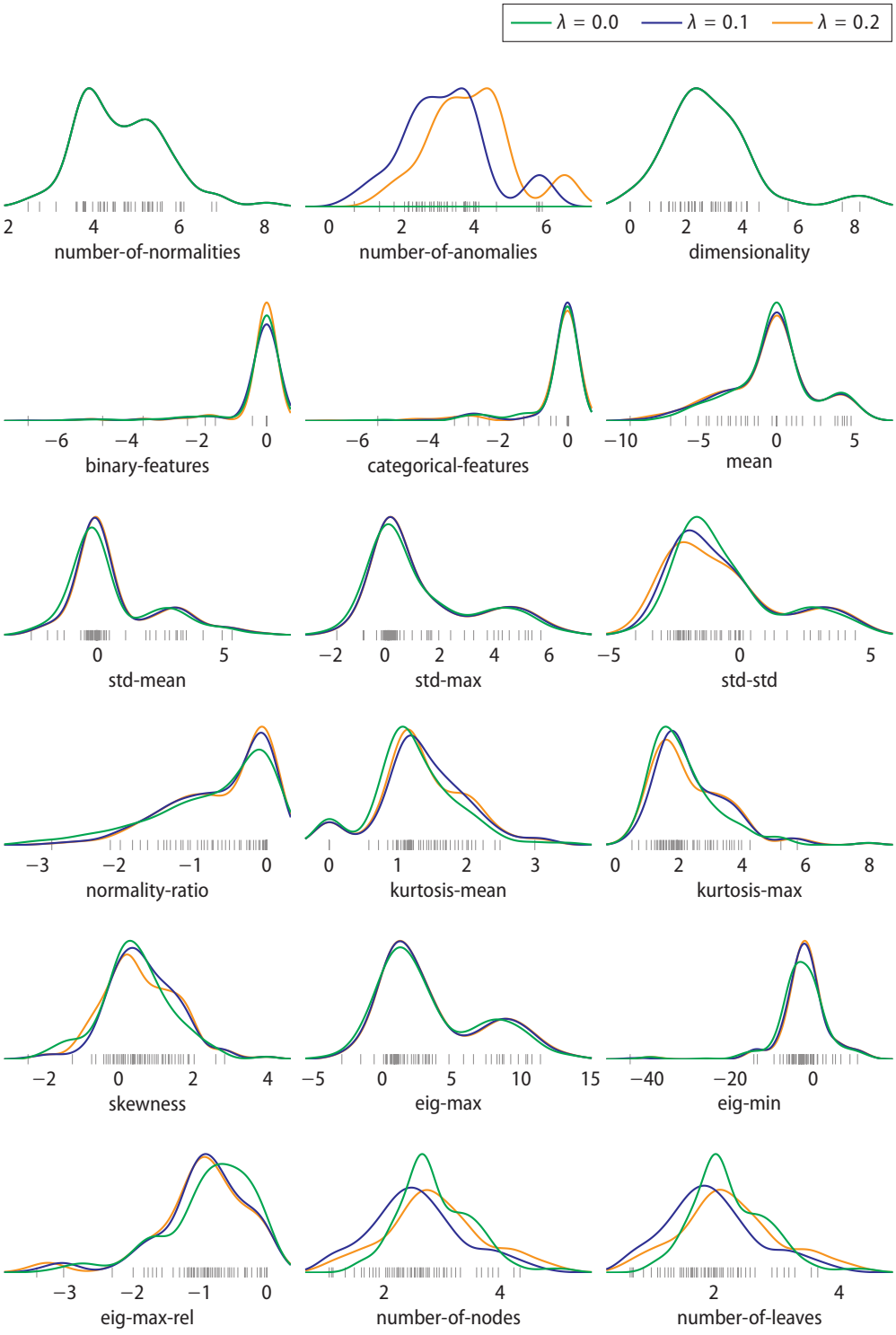


Figure 5.6 Density plots of the log-transformed meta-features for $\lambda = 0.0$, 0.1, and 0.2.

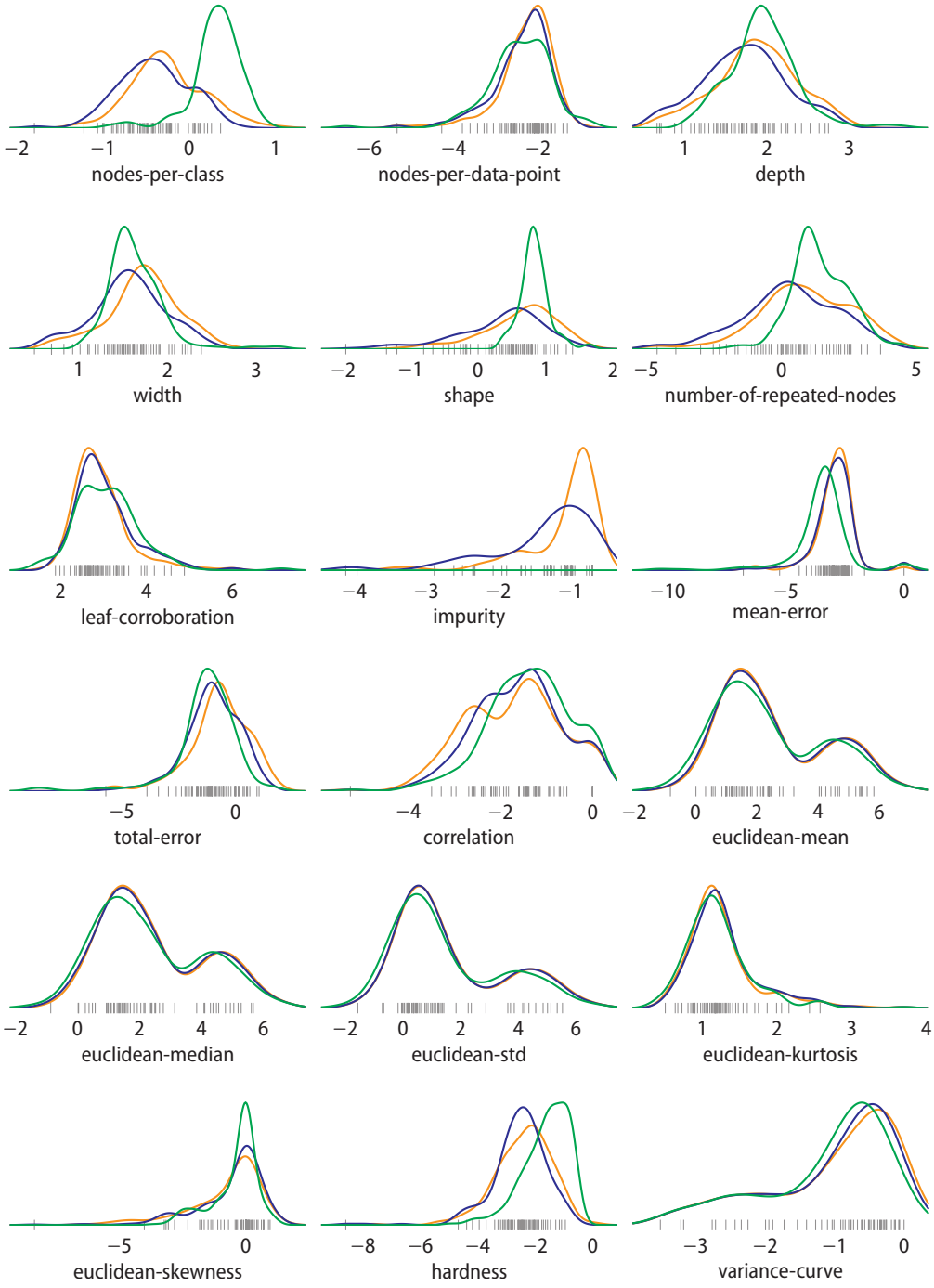


Figure 5.6 (Continued)

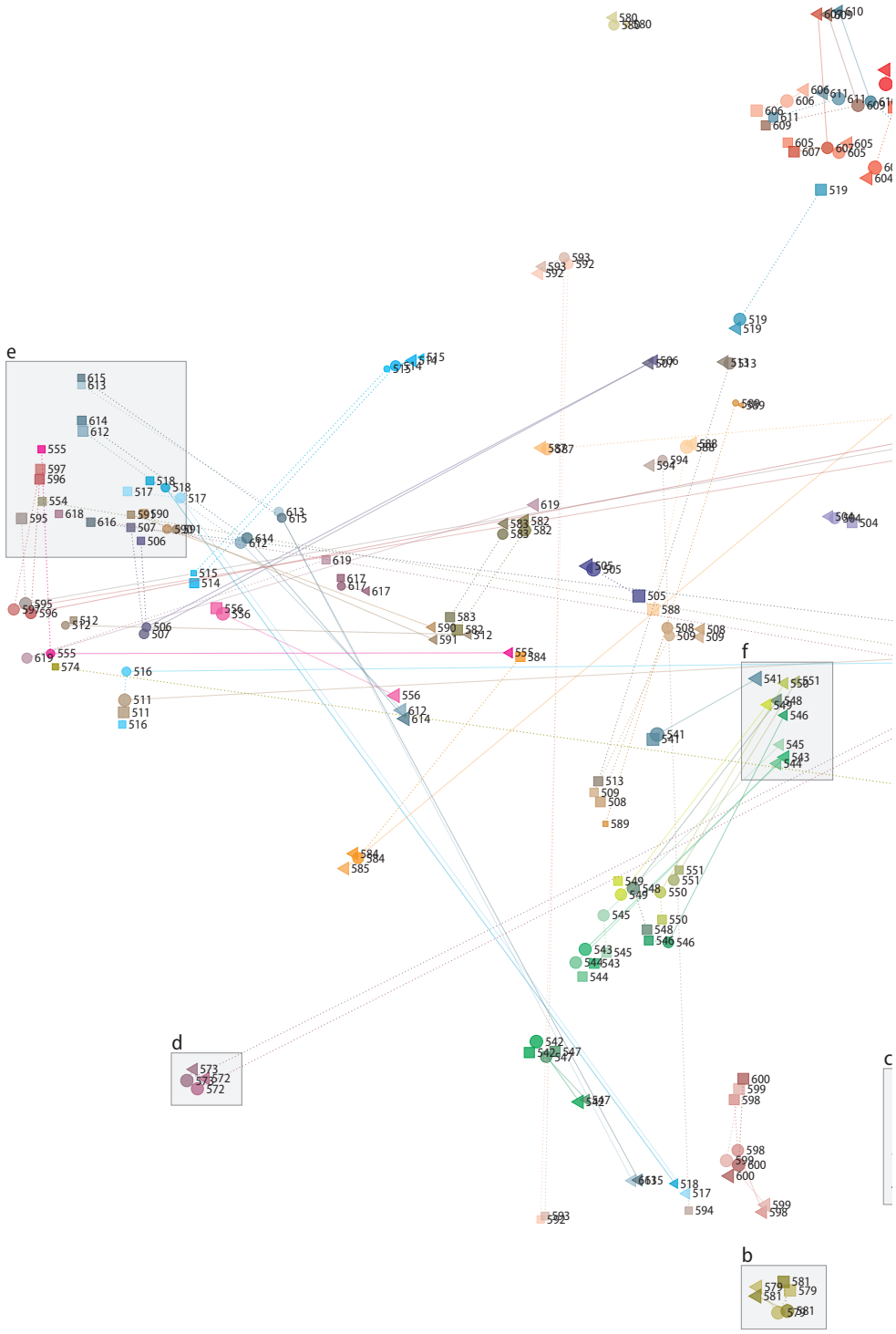


Figure 5.7 Two-dimensional t-SNE plot of 255 one-class data sets based on meta-feature values.

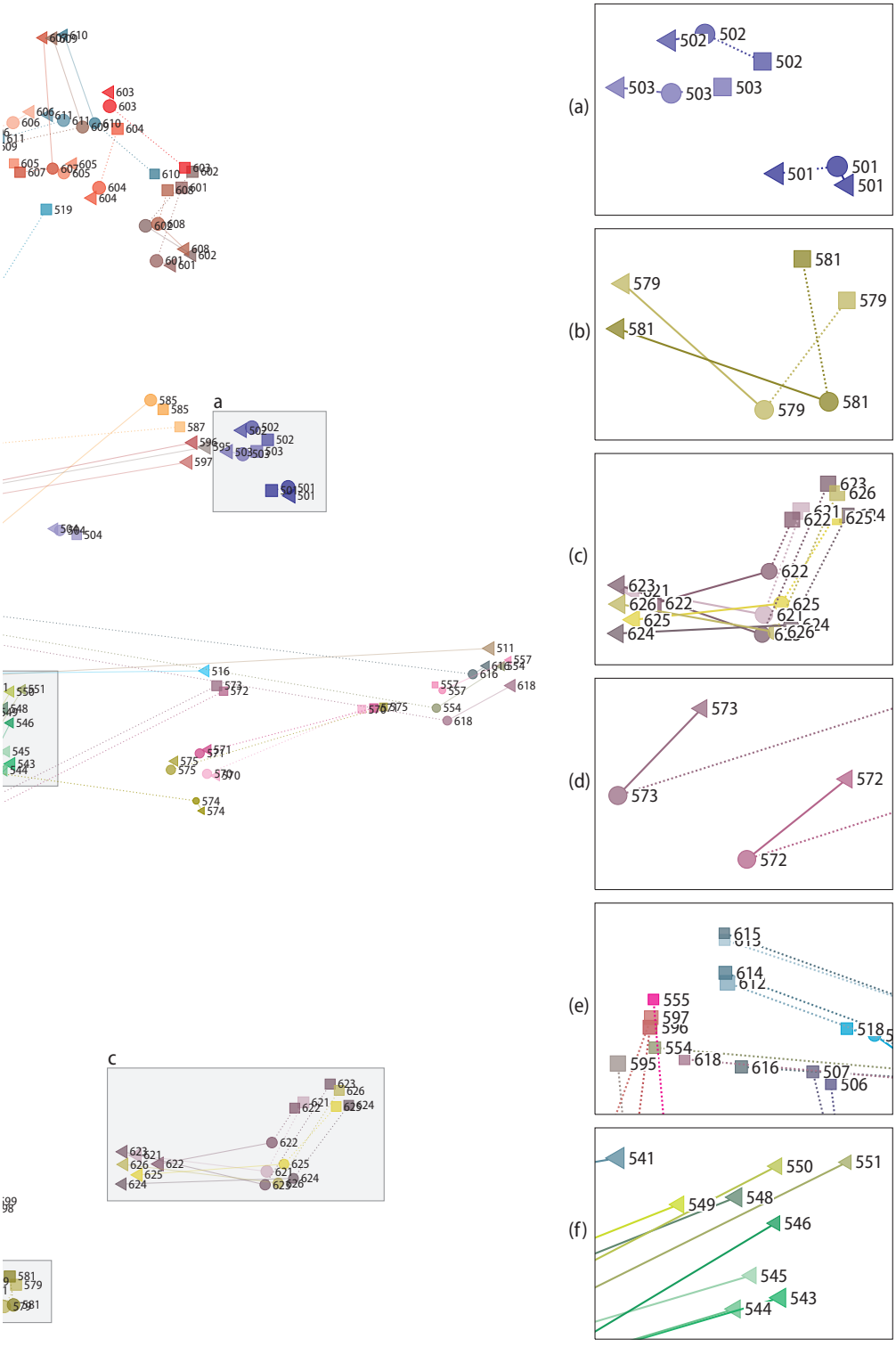


Figure 5.7 (Continued)

The one-class data sets belonging to the same group (i.e., 501N, 501P, and 501V) all have the same colour. None-processed one-class data sets (circles) are connected to the PCA data sets (squares) with a solid line and to the Variance data sets with a dashed line.

The figure shows that for most one-class data sets, preprocessing drastically changes the values of the meta-features. This results in data sets within the same group to be at very different positions in the scatter plot. However, some groups of data sets do not seem to be affected as they appear close together. For example, the Iris data sets (501, 502, and 503), as shown in inset (a) all appear close together. The same holds for the group of one-class data sets in insets (b) and (c). Insets (d), (e), and (f) highlight data sets where one preprocessing greatly affects the meta-features. Moreover, we can see in insets (e) and (f) that many data sets preprocessed with the same technique are located together. From the presence of clusters indicates we may conclude that the meta-data set contains structure. (Whether this structure corresponds to one-class classifier performance is investigated in Chapter 6.)

The size of each marker is actually a preview of what is yet to come—it represents the average AUC that is obtained by applying the 19 one-class classifiers of Chapter 6 on the associated one-class data set.

5.8 Chapter summary

In this chapter we started with our quest to answer RQ3: *To what extent can we solve the one-class classifier selection problem using a meta-learning approach?* The rationale of the quest was that we observed that the 'No Free Lunch' theorem also applies to one-class classification. The theorem implies that there is no single best one-class classifier. In other words, for each one-class data set, there may be a different one-class classifier that performs best. Our goal is to select automatically the best one-class classifier using a meta-learning approach.

We stated four prerequisites that needed to be satisfied in order to solve the one-class classifier selection problem using a meta-learning approach: (1) a large number of one-class data sets, (2) suitable meta-features, (3) a large number of one-class classifiers, and (4) a performance measure. In this chapter, we satisfied prerequisites (1) and (2) as follows. (Prerequisite (3) will be satisfied in Chapter 6 and prerequisite (4) was already satisfied in Chapter 2 as the AUC performance measure.)

We presented 85 one-class data sets. Because of two preprocessing techniques, (1) variance and (2) PCA, we actually had 255 one-class data sets. We defined 36 meta-features from six categories: (1) elementary meta-features, (2) statistical meta-features, (3) decision tree-based meta-features, (4) information-theory-based meta-features, (5) Euclidean-distance-based meta-features, and (6) miscellaneous meta-features. We applied the 36 meta-features to the 255 one-class data sets, which produced a set of meta-feature values. The $3 \times 36 \times 255$ meta-feature values are used as input for the next chapter, where we use meta-learning to relate the meta-features to one-class classifier performance.

Chapter 6
Meta-learning for one-class classifiers

Contents:

In this chapter we will complete the answer to RQ3: *To what extent can we solve the one-class classifier selection problem using a meta-learning approach?* In order to achieve that, we will us the 36 meta-features computed in Chapter 5. We apply 19 one-class classifiers to the 255 one-class data sets. We start describing our meta-learning strategy of learning when to select which one-class classifier. Three baseline strategies are introduced to put our strategy into perspective. The quality of the four strategies is compared and discussed. Finally, we provide our conclusions to RQ3.

Outline:

6.1 Defining mapping and strategy. **6.2** Overview of one-class classifiers. **6.3** Applying 19 one-class classifiers to 255 one-class data sets. **6.4** A meta-learning strategy for a selection mapping. **6.5** Three baseline selection strategies. **6.6** Results and discussion. **6.7** Chapter conclusions.

In this chapter we continue to answer RQ3: *To what extent can we solve the one-class classifier selection problem using a meta-learning approach?* We recall from Chapter 5 the four prerequisites to solving the one-class classifier selection problem using a meta-learning approach. For clarity, we reiterate them here.

1. A large number of one-class data sets of various complexities.
2. Existence of suitable meta-features to characterise the one-class data sets.
3. A large number of diverse one-class classifiers to apply to the one-class data sets.
4. A performance measure to evaluate one-class classifier performance.

Prerequisites (1) and (2) have been satisfied in Chapter 5 by introducing 255 one-class data sets and 36 meta-features. Prerequisite (4) is satisfied as it is the AUC performance measure. In the current chapter we start defining the one-class classifier selection mapping and a corresponding strategy in Section 6.1. In Section 6.2 we introduce the 19 one-class classifiers used in our experiments. In Section 6.3 we explain how we apply the one-class classifiers to the 255 one-class data sets. In Section 6.4 we describe our meta-learning-based one-class classifier selection strategy. In Section 6.5 we introduce three one-class selection strategies that do not involve meta-learning that are used to put our strategy into perspective. In Section 6.6 we present and discuss the results and compare the four one-class classifier selection strategies. In Section 6.7 we provide our chapter conclusions and answer RQ3.

6.1 Defining mapping and strategy

We employ the four prerequisites to find the selection mapping $S(f(\mathcal{D}))$ (cf. Definition 5.1). Below we give two important definitions, one on one-class classifier selection mapping and one on one-class classifier selection strategy.

Definition 6.1 (One-class classifier selection mapping)**.** *A one-class classifier selection mapping is a function $S : \mathcal{D} \rightarrow f$ that takes as input a one-class data set, $\mathcal{D}$, and returns a one-class classifier, f, that is to be applied to that one-class data set.*

A one-class classifier selection mapping is obtained by employing a strategy, which we define as follows.

Definition 6.2 (One-class classifier selection strategy)**.** *A one-class classifier selection strategy is any procedure that results in a one-class classifier selection mapping.*

6.2 Overview of one-class classifiers

In our experiments we employ 19 different one-class classifiers. Seven one-class classifiers, such as SOS and KNNDD have been employed and discussed in previous chapters. Discussing every one-class classifier is beyond the scope of this chapter. Any reader who wishes to learn more on a certain one-class classifier (from the remaining twelve) is kindly referred to the literature referenced. Table 6.1 lists the 19 one-class classifiers that we use in our experiments. For each one-class classifier the table states: (1) the codename we use in the text, (2) the full name, and (3) the reference to the literature in which more information can be found.

Table 6.1 Overview of the 19 one-class classifiers.

Code	Name	Reference
Auto	Auto-Encoder Neural Network data description	Japkowicz, Myers, and Gluck (1995)
Ball	L_p-Ball data description	Tax, Juszczak, Pękalska, and Duin (2006)
Gauss	Gaussian density data description	Parra, Deco, and Miesbach (1996)
IncSV	Incremental Support Vector data description	Tax and Duin (1999)
KCenter	K-Center data description	Ypma and Duin (1998)
KMeans	K-means data description	Bishop (1995)
KNN	K-Nearest Neighbour data description	Tax (2001)
LOF	Local Outlier Factor data description	Breunig, Kriegel, Ng, and Sander (2000)
LP	Linear Programming data description	Pękalska, Tax, and Duin (2003)
MPM	Minimax Probability Machine data description	Lanckriet, Ghaoui, Bhattacharyya, and Jordan (2002)
MST	Minimum Spanning Tree data description	Juszczak, Tax, Pękalska, and Duin (2009)
MoG	Mixture of Gaussians data description	Bishop (1995)
NN	Nearest Neighbour data description	Tax (2001)
NParzen	Naive Parzen density data description	Parzen (1962)
OKNN	Optimised K-Nearest Neighbour data description	Tax (2001)
PCA	Principal Component data description	Bishop (1995)
Parzen	Parzen density data description	Kraaijveld and Duin (2001)
SOM	Self-Organising Map data description	Kohonen (2001)
SOS	Stochastic Outlier Selection data description	*this thesis*

6.3 Applying 19 one-class classifiers to 255 one-class data sets

In Section 5.2 we presented a formal model for the one-class classifier selection problem (see Figure 5.1 on page 102). The fourth component of that model is the performance space $\mathcal{AUC}$, which represents the mapping of each one-class classifier to the AUC performance measure. We are able to obtain a set of AUC performances (i.e., a subset of the performance space) by applying one-class classifiers to one-class data sets.

In this section we apply the 19 one-class classifiers to the 255 one-class data sets. This results in 19 × 255, i.e., 4,845, AUC performance values.

Figures 6.1 and 6.2 show the AUC performances in four matrices. The first matrix shows the AUC performances that the 19 one-class classifiers obtained on the 85 None-preprocessed one-class datasets. The one-class classifiers are shown in alphabetical order along the top side of the matrix, and the one-class data sets are indicated along the left side of the matrix. The colour of each cell in the matrix corresponds to the AUC performance. The colour bar on the right shows the mapping from AUC performances to colours. (Please note that AUC performances between 0 and 0.5 all map to the same colour, i.e., dark red.) For instance, the top row of the matrix, which corresponds to data set 501N (cf. Table 5.1 on page 106), shows that all 19 one-class classifiers obtained an AUC performance of at least 95%.

A black-coloured cell indicates that the AUC performance is missing either because (1) the one-class classifier was not able to process the one-class data set, (2) the processing took too long, or (3) something else went wrong.

The second and third matrix show the AUC performances which were obtained on the one-class data sets that were preprocessed by the Variance and PCA procedures, respectively. The order of both the one-class data sets and the one-class classifiers in these two matrices are the same as in the first matrix.

The fourth matrix shows again the AUC performances for the None preprocessed one-class data sets, but then in a different order. First of all, the columns, i.e., the one-class classifiers, are ordered by the average AUC performance which is obtained on the 85 one-class data sets. Second, the AUC performances in each column are sorted separately, in non-ascending order. So, there is no single one-class data set per row any more. From this matrix we can see, for instance, that MST has the best average performance.

If we compare the colours in the first three matrices, we can see, on a highly detailed level, how the preprocessing procedures affect the performance of the one-class classifiers. As a case in point, the third matrix has quite some more of the red colour than the first and second matrices, which indicates that the PCA preprocessing is not always beneficial for the AUC performance.

Figures 6.3 and 6.4 show how the Variance and PCA preprocessing procedures affect the AUC performance of one-class classifiers, per one-class data set. The data sets are ordered by the median AUC performance. For each data set, there are two overlapping

box plots. The grey box plot describes the distribution of AUC performances for the None-preprocessed data sets. The colour of the other box plot is either green or red. The colour depends on whether median AUC performances of the preprocessed data set is higher (green) or lower (red) than the None-preprocessed one-class data set.

From the two figures, we can see (1) that the overall performance on, for example, data set 597 is improved by preprocessing it by the Variance procedure, but also (2) that the overall performance will be degraded if preprocessed by the PCA procedure.

The computed AUCs are used for evaluating the one-class classifier selection strategies. Since we know exactly the performance that a one-class classifier may obtain because all values have been precomputed, we have an upper bound on the AUC performance. The strategy used selects a one-class classifier for each one-class data set. Of course, we will compare the precomputed AUC of the selected one-classifier with, for example, the average or the maximum AUC for that one-class data set.

6.4 A meta-learning strategy for a selection mapping

We recall from Chapter 5 that we use meta-learning to find a selection mapping $S(f(\mathcal{D}))$ such that we obtain a maximum AUC performance on a given data set $\mathcal{D}$. Thus, the selection mapping is basically a function that maps one-class data sets to one-class classifiers. (In other words, the one-class data sets are the domain of S and the one-class classifiers are the codomain of S).

In this section we describe our suggested solution to the one-class classifier selection problem. In other words, we explain how we employ a meta-learning strategy in order to obtain a one-class classifier selection mapping.

The section is structured as follows. In Subsection 6.4.1, we describe how the meta-data set looks like. In Subsection 6.4.2, we explain how we prepare the meta-data set for binary classification. In Subsection 6.4.3, we present the general classification approach. In Subsection 6.4.4, we discuss how we select a subset of features. In Subsection 6.4.5, we present the seven variants of the classification approach.

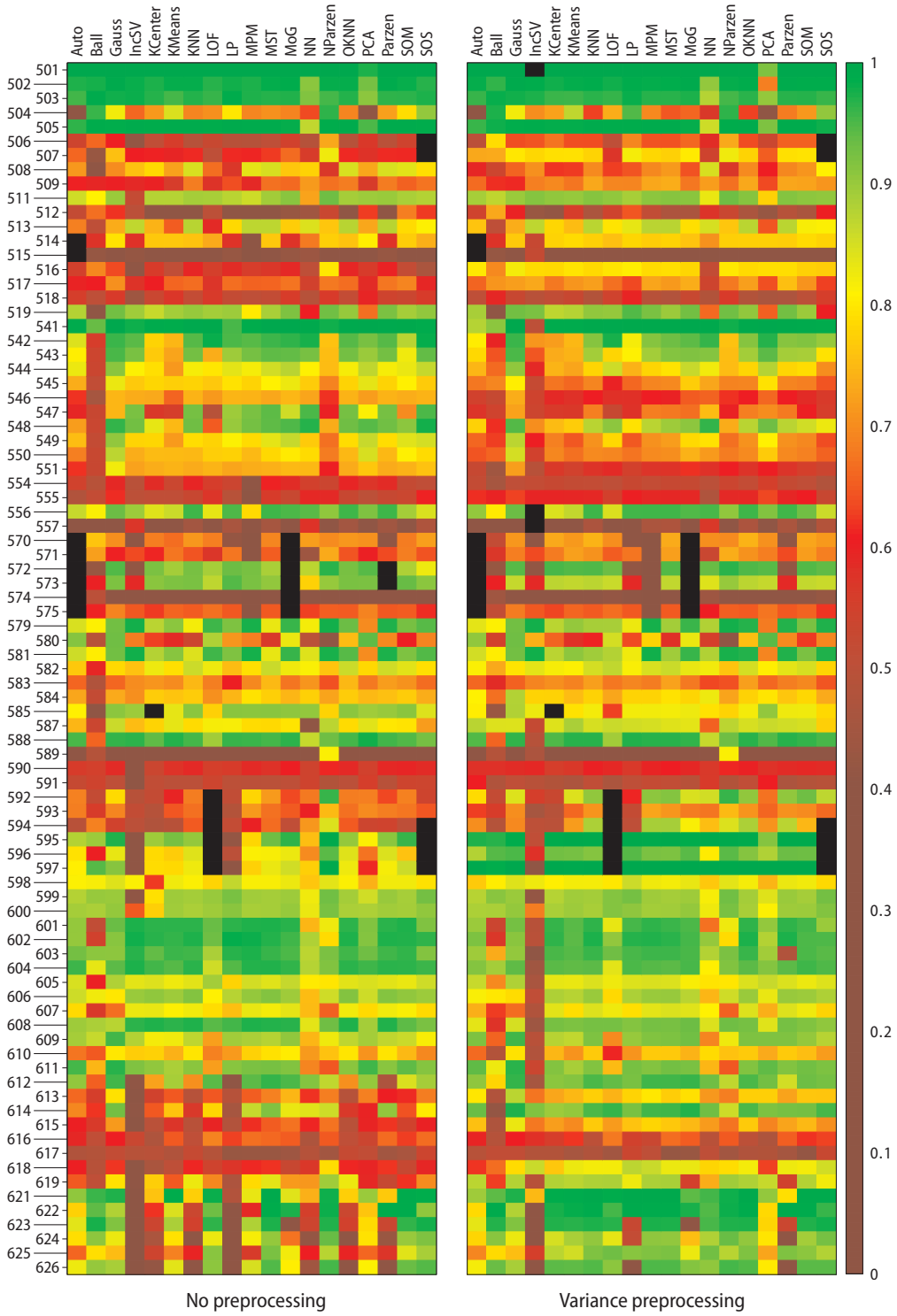


Figure 6.1 AUC matrices for preprocessing techniques None (left) and Variance (right).

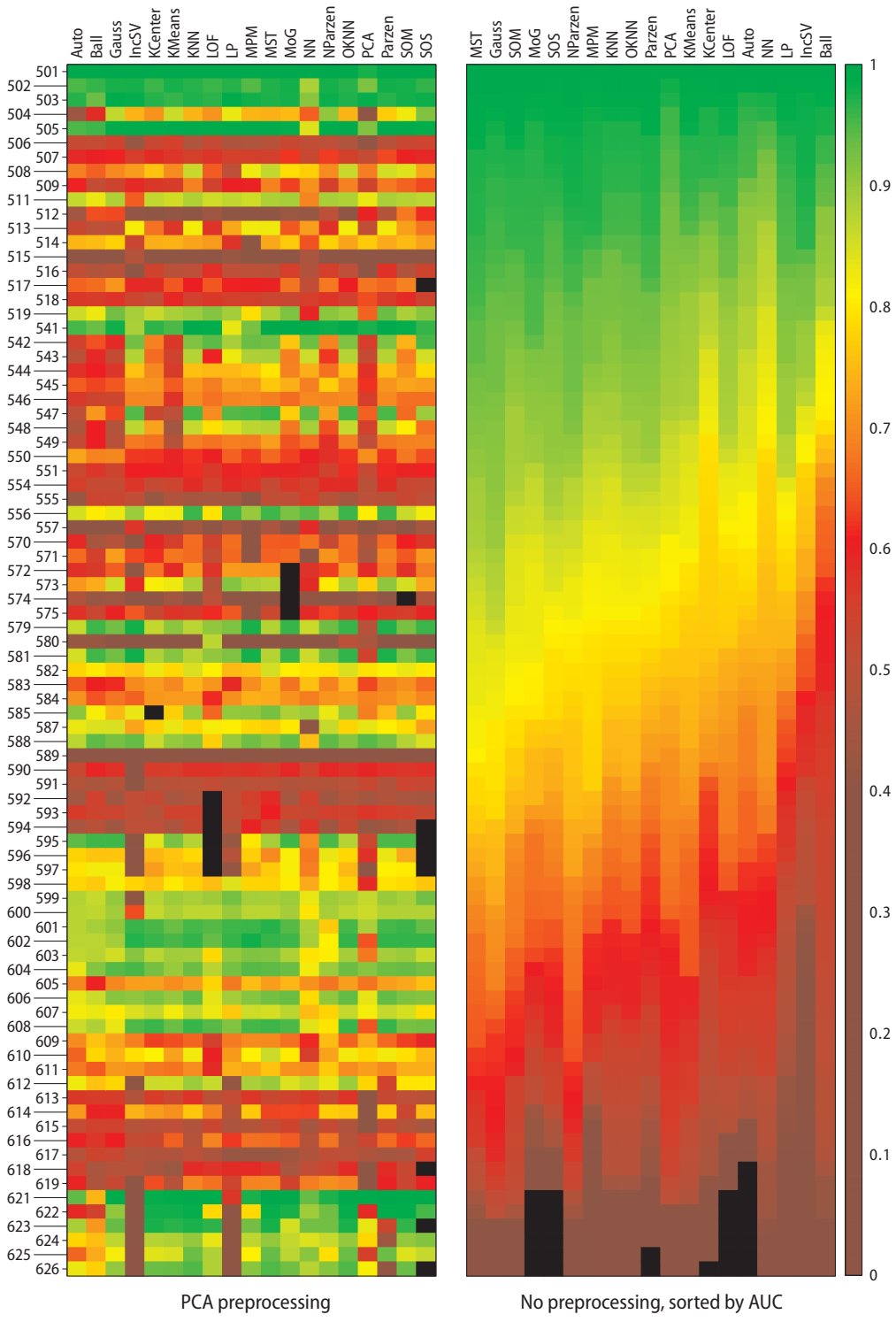


Figure 6.2 AUC matrices for the preprocessing technique PCA (left) and None, ordered by AUC (right).

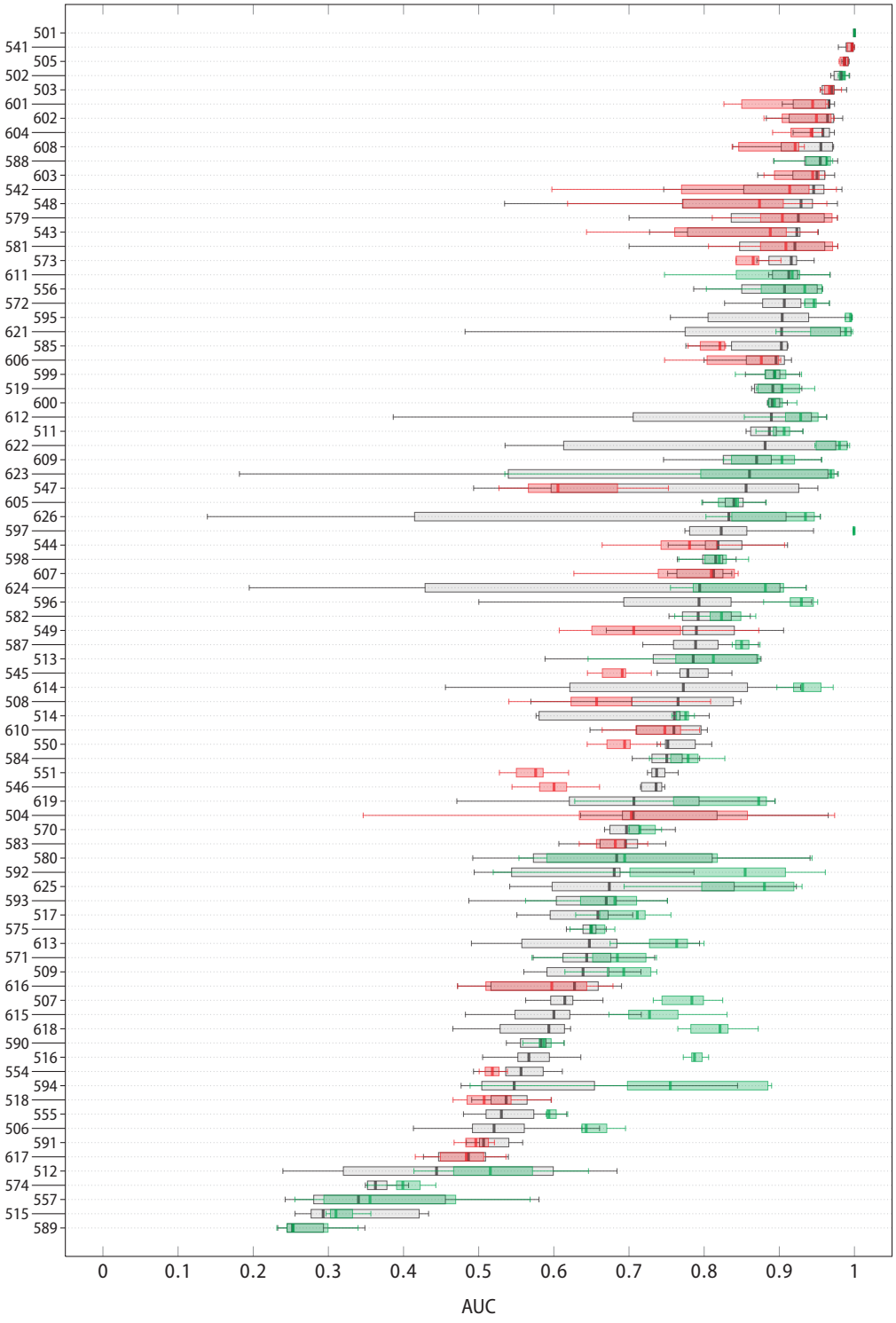


Figure 6.3 Boxplots indicating the influence of preprocessing Variance on the AUC performance.

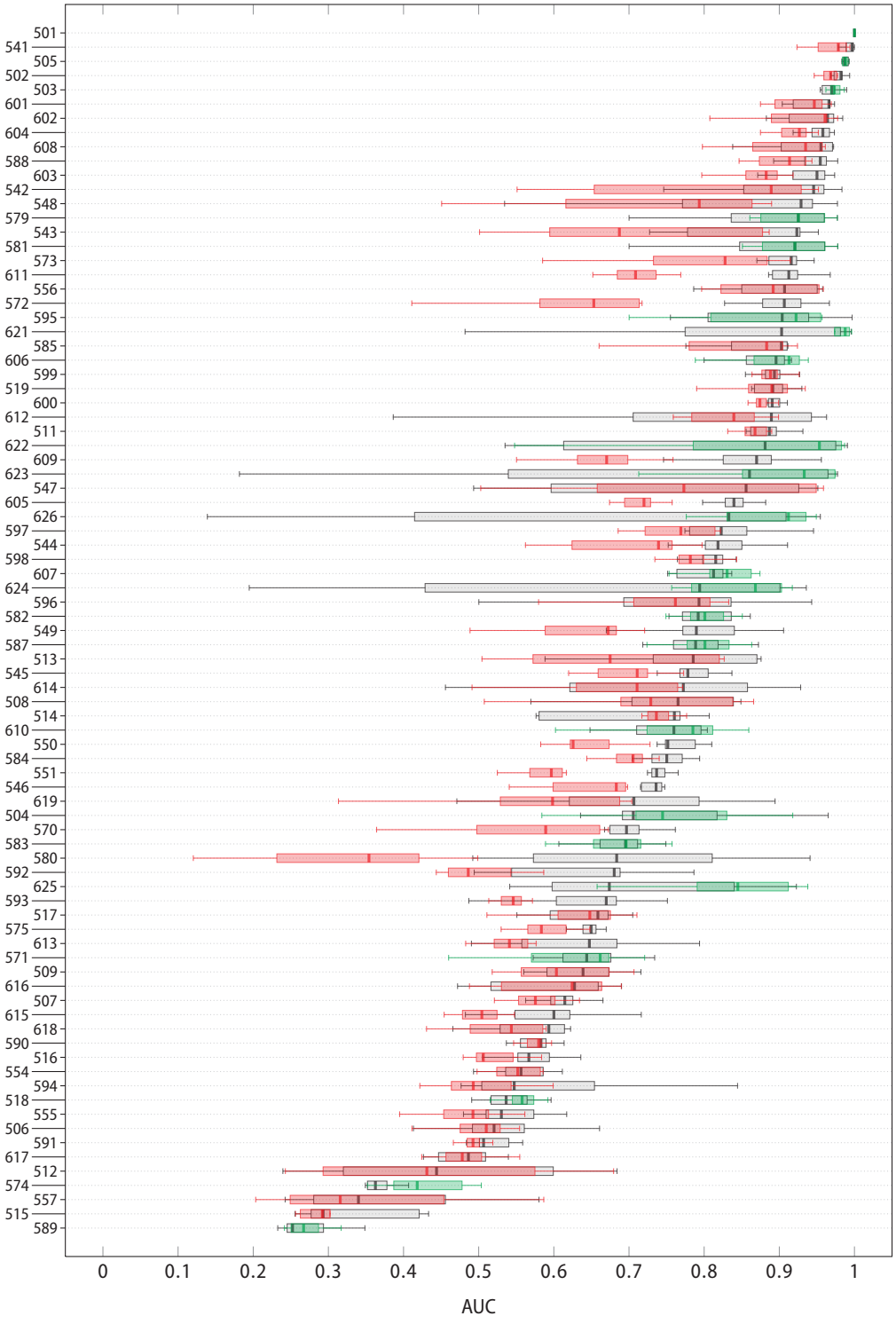


Figure 6.4 Boxplots indicating the influence of preprocessing PCA on the AUC performance.

6.4.1 The meta-data set

As mentioned in Chapter 5, a meta-learning strategy involves machine learning. More precisely we use a form of classification to obtain a mapping from data sets to one-class classifiers.

The data set that we use in our machine learning procedure was earlier obtained by (1) computing 36 meta-features of 255 one-class data sets in Section 5.6 and (2) applying 19 one-class classifiers on the same 255 one-class data sets. Thus, the data set has 255 data points (the one-class data sets) where each data point is a feature vector of length 36 (the meta-features). The data set is labelled, but the labels do not have a binary form as seen thus far. Each data point (one-class data set) has 19 AUC values associated with it.

6.4.2 Preparing the meta-data set for classification

We cannot straightforwardly apply any classifier to such a labelled data set. Therefore, we employ the following classification procedure to obtain a mapping from one-class data sets to one-class classifiers.

At this point we need to handle the data set as follows. The features of each data point (i.e., the meta-features of the one-class data set) can remain the same. The labels (i.e., the 19 AUC values) have to be transformed. Instead of trying to learn a mapping from the set of one-class data sets to the set of 19 one-class classifiers, we perform a simplified classification task. Given two one-class classifiers, say i and j, we learn which one-class classifier is the better one. The new label for the data point is 1 if the AUC performance of one-class classifier i is higher than that of j, and 0 otherwise. On the transformed meta-data set we apply a binary classifier.

6.4.3 General classification approach

As stated above we are ready to apply a binary classifier on the transformed meta-data set. Since the meta-data set has only 255 data points, and since we want to train the binary classifier with as much data points as possible, the leave-one-out cross-validation is preferred over the 10-fold cross-validation.

In leave-one-out cross-validation, the classifier is trained on 254 data points and tested on the remaining data point. As we know, two thirds of the data points are actually preprocessed versions of the first 85 data sets. So, the binary classifier may be at an unfair

advantage when, for example, the training meta-data set contains one-class datasets 501N and 501P, and we test it for 501V. (These three one-class data sets may have too many meta-features in common.) The solution is to employ leave-*three*-out cross-validation instead of leave-one-out cross validation. This ensures a fair training and testing environment. Now, the binary classifier is being trained on 252 one-class data points and tested on the remaining three data points (for example, 501N, 501P, 501V). This is repeated 85 times, such that the binary classifier has tested all data points.

We have 19 one-class classifiers, which results in $19 \times 18 / 2 = 171$ pairs of one-class classifiers. (It does not make sense to compare a one-class classifier with itself and comparing one-class classifiers i with j is the same as comparing j with i.) The meta-data set is therefore transformed 171 times and we apply a binary classifier on each of them.

Since the classifying procedure is quite involved with 171 binary classifiers and leave-three-out cross-validation, it would be adequate to employ a binary classifier that is cheap to use. In our experiments we have chosen to employ the K-Nearest Neighbour binary classifier, where we set K arbitrarily to 3.

We count, per data point, i.e., for each one-class data set, how often each one-class classifier has been selected by the 171 binary classifiers. The one-class classifier that has been selected the most often is the one-class classifier that we let our one-class classifier selection mapping select for that one-class data set. The final result is a mapping of 255 one-class data sets to 255 times one of the 19 one-class classifiers.

In Section 5.7 we noticed that some meta-features have very high values for certain one-class data sets. To mitigate this, we applied the log transformation.

6.4.4 Feature selection

In total we have 36 meta-features. Not all 36 meta-features are informative, and some may even be degrading the quality of the selection mapping. Earlier, for our one-class data sets, we used PCA as a preprocessing technique to solve this problem (cf. Subsection 5.5.3). However, if we use PCA on the meta-data set, then we will not know which meta-feature is the most informative. In other words, we would lose interesting information. So, rather than performing PCA, we select a subset of the meta-features using a technique called Recursive Feature Elimination (RFE) (Guyon, Weston, Barnhill, and Vapnik, 2002).

RFE works as follows. A Support Vector Machine (SVM) with a linear kernel is applied to the meta-data set with all 36 features. The SVM finds an optimal linear separation (i.e., discriminant function) between the two classes (i.e., 1 if classifier i is better than j and 0 otherwise). The discriminant function consists of 36 weights, where each weight determines the importance of a feature. The feature with the least importance (i.e., lowest weight) is eliminated from the meta-data set. The SVM is applied again, but then on the remaining 35 features—hence the name Recursive Feature Elimination.

RFE is performed 171 times, once for each pair of one-class classifiers. The 20 features that are deemed most important after 171 instantiations of RFE are selected to become the features of the meta-data set that we will use for the meta-learning approach.

6.4.5 Seven variants

For our meta-data set, we may choose whether we wish to use meta-features that are (1) standardised or not, (2) log transformed or not, and (3) selected or not. There is no meta-data set with 'selected meta-features' because applying feature selection to raw meta-features did not work, i.e., the feature ranking algorithm fails to process the raw meta-features. This is most likely due to the large values in the raw meta-features. Together with the raw meta-features, we have seven different meta-data sets which we will employ for our meta-learning approach. They are:

- raw meta-features;
- standardised meta-features;
- standardised and selected meta-features;
- log meta-features;
- log and selected meta-features;
- log and standardised meta-features; and
- log, standardised, and selected meta-features.

Subsequently, we apply each variant to each meta-data set with anomaly fraction $\lambda \in \{0.0, 0.1, 0.2.\}$. This results in 21 selection mappings.

In Section 6.5 we introduce three baseline selection strategies for reason of comparison. Finally, in Section 6.6 we evaluate the performance of our strategy and the seven variants.

6.5 Three baseline selection strategies

To put the performance of our strategy into a proper perspective, we introduce three baseline strategies for selecting a one-class classifier. These three strategies only depend on the precomputed AUC performances and do not employ any meta-learning. They are called: random (Subsection 6.5.1), best-on-average (Subsection 6.5.2), and oracle (Subsection 6.5.3).

6.5.1 Random strategy

The first strategy is the 'random' strategy. As the name suggests, this strategy selects randomly one of the 19 one-class classifiers. This is the same as having no strategy, or in other words, having no prior knowledge about any of the one-class classifiers.

6.5.2 Best-on-average strategy

The second strategy is the 'best-on-average' strategy. This strategy selects the one-class classifier that obtained the overall best AUC performance on all data sets. It therefore always selects the same one-class classifier, independent of the one-class data set at hand. In our case, this corresponds to the one-class classifier MST. This strategy is useful if one does have prior knowledge of the one-class classifiers, but does not know when to select which one-class classifier.

6.5.3 Oracle strategy

The third strategy is the 'oracle' strategy. The oracle strategy always selects the one-class classifier with the highest AUC performance possible. This is possible because we have all the AUC performances at our disposal. If we assume that the 19 one-class classifiers used in our experiment are the only ones that exist, then this is the optimal strategy. However, it is also cheating because, in practice, we do not have the AUC performances for a new data set available—hence the name oracle. It is worthwhile to include this strategy into our comparisons because this allows us to see how close our meta-learning strategy is to the optimal performance.

6.6 Results and discussion

In Section 6.4 we introduced our meta-learning strategy to solve the one-class classifier selection problem. Our meta-learning strategy has seven variants and three settings of the λ parameter, producing 21 selection mappings. In Section 6.5 we described the random, best-on-average, and oracle baseline selection strategies, which each produce one mapping. In the current section we compare the four baseline selection strategies by evaluating the quality of the mappings they produce.

The quality of a mapping is determined by the one-class classifier it selects for each one-class data set. Given a one-class data set, and the selected one-class classifier, we look up the precomputed AUC performance (see Section 6.3). Because we have 255 one-class data sets, each mapping has 255 AUC values associated with it. These 255 values determine the quality of a mapping.

The meta-learning, best-on-average, and oracle strategies are deterministic in the sense that they produce the same mapping for the same one-class data sets. The random strategy is, as its name implies, stochastic, meaning that it produces a different mapping every time, even if the one-class data sets remain the same. For the random strategy we use the expected AUC performance on a given data set, which equals the average of all 19 AUC performances obtained for that data set.

The remainder of the section is structured as follows. In Subsection 6.6.1 we compare the meta-learning strategy to the other three strategies on a high level. In Subsection 6.6.2 we look more closely at how the seven variants of the meta-learning strategy compare to each other.

6.6.1 Meta-learning strategy better than random

In this subsection we aim to obtain an overview of the qualities of the mappings that the four strategies have produced. This allows us to see whether there are any remarkable differences. We recall that, in total, we have 24 mappings (21 from our meta-learning strategy, and 3 from the other strategies). In order to obtain an adequate overview of the qualities of 21 mappings, we visualise the distributions of the associated AUC performances. Figure 6.5 contains, for each mapping, a graph that represents the distribution of the AUC performances. We recall that the AUC performance range from 0 to 1. The graphs are generated by KDE, which allows us look at the AUC performances

as distributions. Please note that KDE smooths the graphs that causes the values to go beyond 1. This is a side-effect of most mappings leading to a large number of high AUC performances.

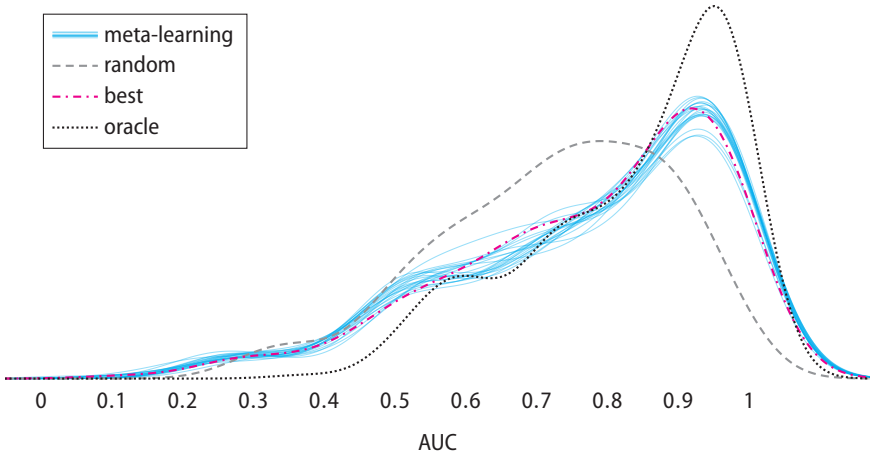


Figure 6.5 Distribution of the AUCs of the mappings generated by the four strategies.

The 21 mappings produced by the meta-learning strategy are grouped together for purpose of establishing a good overview. (We have a closer look at the differences among the 21 meta-learning mappings in the next Subsection.) The meta-learning mappings are indicated by light-blue lines. The random, best-on-average, and oracle strategies are indicated by a grey dashed line, a purple dash-dotted line, and a black dotted line, respectively. If one line is higher than another line for some AUC performance, then the corresponding mapping has selected more one-class classifiers that obtained that AUC performance.

Using Figure 6.5, we now discuss the random, oracle, best-on-average, and meta-learning strategies respectively. The mapping produced by the random strategy has led to relatively more AUCs between roughly 0.5 and 0.8 than any other mapping (i.e., expected AUCs, since the random strategy is stochastic). A consequence is that the random mapping leads to few AUCs that are higher than 0.8. From the figure we may conclude that the random strategy has the lowest quality of the four strategies.

We recall that the oracle strategy is to choose the best one-class classifier available per one-class data set. Thus, the graph shows the distribution of the highest AUC

performance obtained on the 255 one-class data sets. While this strategy is impossible in practice, it provides us a perspective.

The random and oracle strategies may be regarded as two extremes at the spectrum of one-class classifier selection strategies. Selecting a random one-class classifier, on the one hand, is what we would do if we knew absolutely nothing about one-class data sets, meta-features, and one-class classifiers. Selecting the best one-class classifier for every one-class data set, on the other hand, is what we could do if we were omniscient regarding one-class data sets, meta-features, and one-class classifiers. All mappings produced by the best-on-average and meta-learning strategies are situated between these two extremes.

The mapping produced by the best-on-average strategy effectively shows the AUC performances obtained by the MST one-class classifier. We can clearly see that this mapping has a much higher quality than the average random mapping.

The 21 mappings produced by our meta-learning strategy all have an AUC distribution similar to that of the best-on-average mapping. This indicates that we are at least looking at the right direction. However, there appears to some variety between the 21 mappings. Some mappings have a higher quality than the best-on-average, and others have lower quality.

Still, the question that remains is, which of the seven variants combined with which setting of λ produces the best mapping?

6.6.2 Comparing the seven meta-learning strategy variants

In this subsection we have a closer look at the seven variants of our meta-learning one-class classifier selection strategy. Figure 6.6 shows seven sub-plots, where each sub-plot corresponds to one of the seven variants. Each sub-plot contains a graph of the meta-learning variant with λ set to 0.0, 0.1, and 0.2. Moreover, we have plotted the random, best-on-average, and oracle mappings. The graphs of these three mappings are the same in each of the seven sub-plots and therefore provide a good anchoring point for comparison with the seven variants.

Because we have already seen in Figure 6.5 that all the mappings lie between the random and the oracle mappings, it makes sense to have a look at the AUC performances relative to those two mappings.

The y-axis in each sub-plot denotes the relative AUC, and is computed as follows.

The absolute AUC associated with a particular one-class classifier for a particular one-class data set $\mathcal{D}$ is transformed into the relative AUC, which is defined as follows.

Definition 6.3 (Relative AUC). *The relative AUC of a one-class classifier is computed by subtracting the random AUC from the absolute AUC divided by the difference between the oracle and the random AUCs:*

$$\mathrm{AUC}_{\mathrm{rel}}\left(\phi_M, \mathcal{D}\right) = \frac{\mathrm{AUC}\left(\phi_M, \mathcal{D}\right) - \mathbb{E}\left[\mathrm{AUC}\left(\phi_R, \mathcal{D}\right)\right]}{\mathrm{AUC}\left(\phi_O, \mathcal{D}\right) - \mathbb{E}\left[\mathrm{AUC}\left(\phi_R, \mathcal{D}\right)\right]}, \tag{6.1}$$

where ϕ, subscripted by M, R, and O indicate the one-class classifier selected by the mapping to be transformed, the random mapping, and the oracle mapping, respectively. The $\mathbb{E}$ indicates the expected value of the AUC of the random mapping.

The relative AUC is 0 when it is equal to the AUC produced by the random mapping and 1 when it is equal to the oracle mapping. The random and oracle mapping therefore appear as two horizontal lines in the seven sub-plots. The relative AUC is lower than 0 if it is lower than the random mapping. The lower bound of the relative AUC depends on both random and oracle AUCs. We only show the relative AUCs between −1 and 1.

The x-axis represents the 255 data sets. Please note that the data sets are ordered per mapping, so we cannot compare the performance per data set. We have arranged matters in this way because otherwise it would be very difficult to compare the mappings. Moreover, we are most interested in the overall quality of the mappings.

From Figure 6.6 we can see that even the best-on-average strategy is worse at selecting than the random strategy for some number of one-class data sets. (This is another confirmation that there is no free lunch for one-class classification. In fact, we may ask ourselves whether there exists a one-class classifier selection strategy, other than the oracle strategy, that always performs better than random?)

To compare the seven variants in a better way we define the Average Relative AUC (ARA).

Definition 6.4 (Average Relative AUC (ARA)). *The Average Relative AUC of a one-class classifier selection mapping S is defined as the average of 255 relative AUCs as obtained on the 255 one-class datasets.*

$$\mathrm{ARA}(S) = \frac{1}{255} \times \sum_{i=1}^{255} \mathrm{AUC}_{\mathrm{rel}}\left(S(\mathcal{D}_i), \mathcal{D}_i\right) \tag{6.2}$$

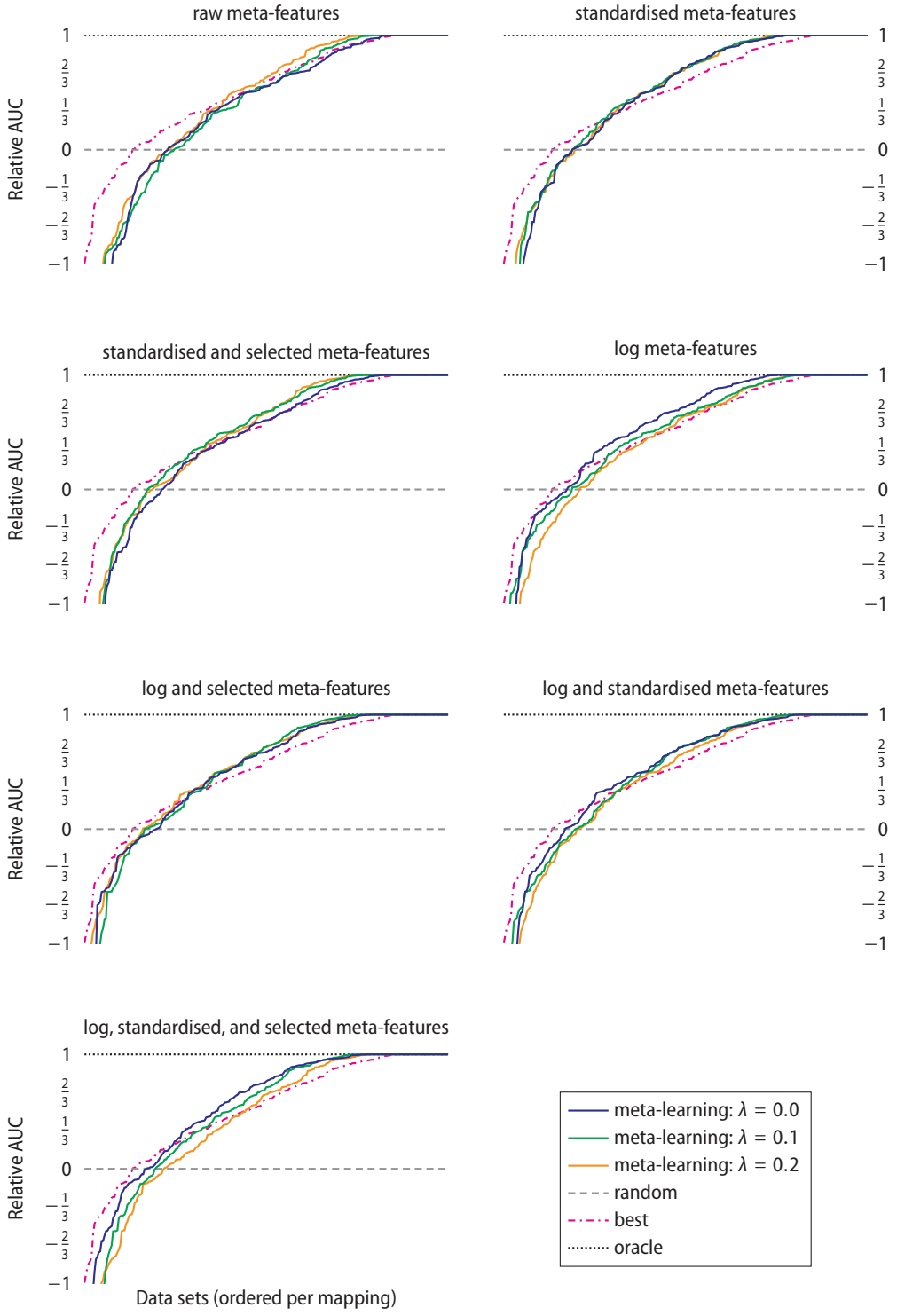


Figure 6.6 Comparing the seven variants of the meta-learning strategy with the other strategies.

where $S(\mathcal{D}_i)$ denotes the one-class classifier as selected by the one-class classifier selection mapping S.

Table 6.2 shows the ARA of each meta-learning variant. The rows are sorted by the first column ($\lambda = 0.0$). The last two columns show the differences that $\lambda = 0.1$ and $\lambda = 0.2$ have with respect to $\lambda = 0.0$.

Table 6.2 ARA of each meta-learning variant.

		ARA			ΔARA	
Meta-learning variant	$\lambda =$	0.0	0.1	0.2	0.1	0.2
raw		.33	.34	.39	+.01	+.06
standardised, selected		.39	.44	.43	+.05	+.04
standardised		.43	.47	.47	+.04	+.04
log, standardised		.49	.47	.43	−.02	−.06
log		.49	.48	.41	−.01	−.09
log, selected		.50	.48	.51	−.02	+.01
log, standardised, selected		.52	.42	.38	−.09	−.14

Table 6.2 confirms that all seven variants of the meta-learning strategy perform better than the random strategy (because they all have a positive ARA). The random and oracle selection mappings have an ARA of 0 and 1, respectively (see Table 6.3). The best-on-average strategy has an ARA of 0.51, which puts it between the 'log, selected' and the 'log, standardised, selected' meta-learning strategies for $\lambda = 0.0$.

Table 6.3 ARA of each baseline selection strategy.

Non-meta-learning variant	ARA
random	0.00
best-on-average	0.51
oracle	1.00

Below we discuss the influence of the two dimensions: (1) the setting of λ and (2) the variants. In Chapter 5 we expected that including anomalies in the one-class data sets for the computation of the meta-features would increase the quality of the meta-learning mappings. However, from the table we can see that setting λ to 0.1 or 0.2, for the four best performing variants (i.e, those that include the option 'log'), decreases the mapping quality. One reason why $\lambda = 0.1$ and $\lambda = 0.2$ perform worse, is that the meta-features do not distinguish between the two classes. So, including anomalies may distort the values of

the meta-features in a negative manner. Thus, increasing the fraction of anomalies used to compute the meta-features, generally decreases the performance.

The variant 'log, standardised, selected' employs all the three options and it performs best. Its ARA is 58% higher than the raw variant, which employ none of the three options. This means, in a general sense, that the quality of the meta-learning strategy can be improved by transforming and selecting the raw meta-features.

Now the time is ripe to discuss the influence of the three options (i.e., log, standardised, and selected) on the quality of the meta-learning mapping. The variants that employ the log option appear to be the best four variants. Just employing the log option to raw meta-features increases the ARA from .33 to .49. The influence of the standardised option is not consistent. On the one hand, employing the standardised option on the raw meta-features increases the quality to .043 and to the 'log, selection' variant it adds .02 ARA. On the other hand, employing it in combination with the log option, it does not appear to make a difference. The selected option, in general, does improve the quality, but not too much. It increases the 'log' variant by .01 and the 'log, standardised' variant by .03. It decreases the quality of the 'standardised' variant by .04.

6.7 Chapter conclusions

In this chapter, we continued with our journey to answer RQ3: *To what extent can we solve the one-class classifier selection problem using a meta-learning approach?* First, we applied 19 one-class classifiers to the 255 one-class data sets presented in Chapter 5. Second, we defined the terms one-class classifier selection strategy and one-class classifier selection mapping. Third, we presented our suggestion to solve the one-class classifier selection problem, by a meta-learning strategy. The aim of the meta-learning strategy was to learn a mapping from one-class data sets to one-class classifiers, such that the AUC was maximised. To learn the mapping, the 36 meta-features of Chapter 5 were used to form a meta-data set. The meta-data set had one parameter, λ, which controlled the fraction of anomalies present, when computing the meta-features. The meta-learning strategy had seven variants, which, together with three settings for λ, resulted in 21 different one-class classifier selection mappings. Fourth, three baseline selection strategies were introduced: random, best-on-average, and oracle. These three strategies were meant to put the performance of our meta-learning strategy into perspective. Fifth, we compared

and discussed the quality of the four strategies. We are now ready to formulate our conclusion to RQ3.

To answer RQ3 we need to define what it means to solve the one-class classifier selection problem completely. According to the model in Chapter 5, the question reads: when do we arrive at a one-class classifier selection mapping that maximises the AUC?

In Subsection 6.5.1, we defined that the random one-class classifier selection strategy is actually equal to having no strategy. It therefore makes sense to define this as the lower bound. Moreover, the oracle strategy delivers what is the maximal achievable result. We know this for sure, because we had precomputed all AUC performances by applying 19 one-class classifiers to 255 one-class data sets. In Subsection 6.6.2, we defined the ARA, which is, by definition, 0 for the random strategy and 1 for the oracle.

So, when we consider the variant of our meta-learning strategy with the highest ARA, namely the 'log, standardised, selected' variant, then we may conclude that, given the 255 one-class data sets and 19 one-class classifiers, we can solve the one-class classifier selection problem using a meta-learning approach for 52%.

In our quest to answer RQ3, which spans Chapters 5 and 6, we have made the following choices regarding the four prerequisites and our meta-learning approach:

- 255 one-class data sets,
- 36 meta-features,
- 19 one-class classifiers,
- AUC performance measure,
- Optional log transformation of certain meta-features,
- Optional standardised of meta-features,
- Optional selecting top 20 meta-features,
- Feature selection using a Support Vector Machine with a linear kernel, and
- Using 191 3-NN binary classifiers for multi-label classification using.

These choices together offered a complete exercise into applying meta-learning for the one-class classifier selection problem. Some of these choices were based on availability (e.g., the data sets and one-class classifiers), some were supported by scientific literature (e.g., the meta-features), and some were guided by preliminary results (e.g., log transformation, feature selection).

We do not claim that these choices are all optimal. Many choices could have been made differently, which may have led to better solutions of the one-class classifier selection problem. However, it is beyond the scope of this thesis to provide a complete exploration of all choices made; that would take too much time.

On the basis of our results as shown in Chapters 5 and 6, we believe that the one-class classifier selection cannot be solved completely. In other words, we believe that there is no strategy that obtains an ARA of 1.0 (unless the number of one-class data sets and one-class classifiers is trivially low). The reason for our belief is that we experience that, just as with one-class classification, there is no free lunch for meta-learning.

Chapter 7
Conclusions

Contents:

This chapter provides answers to the three research questions posed in Chapter 1. Moreover, a definitive conclusion to the problem statement is formulated. Finally, four directions for future research are suggested.

Outline:

7.1 Answers to the research questions. **7.2** Answer to the problem statement. **7.3** Future research.

In this chapter we answer the three research questions on the basis of the work presented in the thesis (Section 7.1). Subsequently, we formulate our conclusion to the problem statement (Section 7.2). Finally, we suggest four directions for future research (Section 7.3).

7.1 Answers to the research questions

> **Research question 1:** *How should we evaluate and compare the performance of outlier-selection algorithms?*

The answer to the first research question is derived from Chapters 2 and 3. In Chapter 2 we described four methods for the evaluation of outlier-selection algorithms and one-class classifiers: (1) derivation of one-class data sets from multi-class data sets, (2) simulation of anomalies, (3) cross-validation with training and test data sets, and (4) computation of the AUC performance measure. Moreover, we described three techniques to compare the performances: (5) the Friedman test, (6) the post-hoc Nemenyi statistical test, and (7) the critical difference diagram.

In Chapter 3, we aimed to evaluate and compare five outlier-selection algorithms from the fields of ML and KDD. However, prior to the evaluation, i.e., applying the outlier selection to one-class data sets, we framed LOF and LOCI into the one-class classifier setting. By doing so, each outlier-selection algorithm was treated under the same circumstances. Subsequently, we were able to employ the four evaluation methods and three comparison techniques listed above.

Although AUC is the most widely-used performance measure within Machine Learning, it is certainly not the only way to evaluate outlier-selection algorithms and one-class classifiers. In any case, we strongly discourage the use of visualizing the classifications by the algorithms only, since that may lead to a subjective evaluation. Instead, we advise to employ (1) a performance measure that has a proven statistical validity and (2) statistical tests to compare the performances and to test whether there is any significant difference.

From our own experimental approach and results we may conclude that outlier-selection algorithms should be evaluated and compared by: (1) ensuring that they are treated under the same circumstances, (2) measuring their performance in an objective manner using statistically valid techniques, and (3) comparing their performance with statistical tests.

Research question 2: *Can an effective outlier-selection algorithm be devised that employs the concept of affinity?*

The answer to the second research question is derived from Chapter 4. We developed and evaluated the Stochastic Outlier Selection (SOS) algorithm, a novel, unsupervised algorithm for classifying outliers in a data set. SOS employs the concept of affinities to compute for each data point an outlier probability.

From our empirical results we observe that (1) SOS is an effective algorithm for classifying outliers in a data set and that (2) SOS compares favourably to state-of-the-art outlier-selection algorithms. We may therefore conclude that the concept of affinities, which forms the basis SOS, is successfully applied to the problem of outlier selection.

Research question 3: *To what extent can we solve the one-class classifier selection problem using a meta-learning approach?*

The answer to the third research question is derived from Chapters 5 and 6. We defined that the one-class classifier selection problem is completely solved when we have found a one-class classifier selection mapping that maximises the AUC. We employed the ARA to evaluate to what extent we solve the one-class classifier selection problem.

On the one hand, we have the random one-class classifier selection strategy, which is equal to having no strategy, i.e., an ARA of 0. On the other hand, we have the oracle strategy, which is what is maximal achievable, i.e., an ARA of 1. In other words, the random and the oracle strategies solve the one-class classifier selection problem for 0% and 100%, respectively.

The 'log, standardised, selected' variant of our meta-learning strategy obtained the highest ARA, namely the 0.52. It was slightly higher than the best-on-average strategy, which had an ARA of 0.51. We may therefore conclude that, given the 255 one-class data sets and 19 one-class classifiers used in our experiments, we can solve the one-class classifier selection problem using a meta-learning approach for 52%.

Because (1) the ARA has not been employed by any previous study and (2) we were, to the best of our knowledge, the first to attempt to solve the one-class classifier selection problem using a meta-learning approach, it is difficult to put the 52% into a proper perspective. Still, we may consider that our result, according to our definition, indicates that we are beyond half-way of solving the one-class classifier selection problem. In Section 7.3 we provide additional research directions that may improve the result.

7.2 Answer to the problem statement

In this section we provide an answer to the problem statement. Our answer is based on the answers to the three research questions as presented in the previous section.

> **Problem statement:** *To what extent can outlier-selection algorithms support domain experts with real-world anomaly detection?*

In Chapter 1 we stated three conditions that ought to be satisfied in order for a domain expert to be supported by outlier-selection algorithms. The three conditions, and how we addressed them, are as follows:

First, a domain expert should know how to evaluate and compare the performance of the algorithms. We have described and shown how to evaluate and compare the performance of outlier-selection algorithms and one-class classifiers.

Second, a domain expert should have one or more algorithms available that are considered state-of-the-art. We developed a new state-of-the art outlier-selection algorithm called SOS. Besides SOS, we have performed experiments with five algorithms in Chapter 3, four algorithms in Chapter 4, and nineteen algorithms in Chapter 6. These algorithms are all available to the domain expert.

Third, a domain expert should know when to apply which algorithm. We have shown that, to a certain extent, a meta-learning approach can provide an expert insight into when to apply which one-class classifier given a certain data set.

Seeing that the above three conditions have been satisfied, we may conclude that the domain expert can be supported to some extent.

7.3 Future research

The research presented in this thesis was complex and promising. Our results were not always the last verdict. Promising areas of future work remain. In this section, we mention four of the most interesting directions.

First, in our research we concentrated on unsupervised outlier-selection algorithms and semi-supervised one-class classifiers that make use of normal data points only. There are also algorithms that can make use of anomalies in the training data set, such as SVDD. We expect that the AUC increases when there are anomalies included in the data set. The success of such an inclusion may be dependent on the number of anomalies.

So, possible future research in the first direction is to investigate whether SOS can be extended in such a way that it can make use of any anomalies in the data set. Then, SOS could be evaluated using the evaluation techniques presented in the thesis and compared with the few other outlier-selection algorithms that do make use of anomalies.

Second, continuing on the first research direction, one could investigate the use of active learning for anomaly detection. Active learning is a research field within Machine Learning that deals with algorithms that are able to select unlabelled data points from the training data set and present these to the domain expert for labelling. The algorithm is then able to make use of the extra label such that its performance will increase. In this scenario, the domain expert is involved in the classification of the data points. In other words, we have a human-in-the-loop system.

Third, in our research regarding meta-learning we focused on learning a mapping from a one-class data set to one-class classifiers. Our experiments involved many choices regarding the data sets, the one class classifiers, and the meta-features. Moreover, our meta-learning approach in itself has many parameters, such as the feature selection aspect and the use of 191 binary classifiers. As mentioned in Section 6.7, a complete exploration of these aspects is beyond the scope of the thesis.

So, an obvious research direction is to continue to vary these choices and to investigate other settings for the parameters of the meta-learning approach. We expect that the ARA may be further improved by changing one or more of the following seven aspects: (1) one-class data sets, (2) meta-features, (3) one-class classifiers, (4) preprocessing techniques, (5) meta-feature transformations, (6) meta-feature selection, and (7) classifier setting for multi-label classification.

However, we should note that, this way, the meta-learning approach may remain a black box to the domain expert. In order to gain more insight into the one-class classifiers and when to use them, one could also mine so-called association rules. Association rules typically take the form of 'if meta-feature x has value y, use one-class classifier z'. When sufficient consistent association rules have been mined, the domain expert could apply them to a new unseen data set.

Fourth, in our research we concentrated on the Euclidean-distance measure. The reason we that did so was to be able to use many feature-based data sets in our experiments. While this is fair and practical from an experimental point-of-view, we believe that often, additional research is required when applying an outlier-selection algorithm to a real-world problem, such as anomaly detection in the maritime domain.

In many real-world problems, especially ones that involve time-series or trajectories, the Euclidean-distance measure may be insufficient or even impossible to use. Then, the research should be focused on representing correctly the real-world objects or events through a suitable dissimilarity measure. Furthermore, it would be interesting to see which algorithm performs best with such a dissimilarity measure.

References

C.C. Aggarwal and P.S. Yu. Outlier detection for high dimensional data. *ACM SIGMOD Record*, 30(2):37–46, 2001. (Cited on page 56)

D.W. Aha. Generalizing from case studies: A case study. In D.H. Sleeman and P. Edwards, editors, *Proceedings of the 9th International Conference on Machine Learning*, pages 1–10, San Francisco, CA, 1992. Morgan Kaufmann Publishers. (Cited on pages 100, 103, and 104)

D.W. Aha, D. Kibler, and M.K. Albert. Instance-based learning algorithms. *Machine Learning*, 6:37–66, 1991. (Cited on page 84)

S. Ali and K.A. Smith. On learning algorithm selection for classification. *Applied Soft Computing*, 6:119–138, 2006. (Cited on pages 100, 103, 104, 114, 115, and 117)

U. Alon, N. Barkai, D.A. Notterman, K. Gish, S. Ybarra, D. Mack, and A.J. Levine. Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. *Proceedings of the National Academy of Sciences*, 96(12):6745–50, June 1999. ISSN 0027-8424. (Cited on page 89)

A. Asuncion and A. Frank. UCI Machine learning repository, 2010. URL http://archive.ics.uci.edu/ml. (Cited on pages 29, 52, 89, and 105)

M.J. Bayarri and J.O. Berger. The interplay of Bayesian and Frequentist analysis. *Statistical Science*, 19(1):58–80, Feb. 2004. ISSN 0883-4237. (Cited on page 75)

H. Bensusan, C. Giraud-Carrier, and C. Kennedy. A higher-order approach to meta-learning. In *Proceedings of the ECML-00 Workshop on Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination*, pages 109–117, June 2000. (Cited on page 118)

C.M. Bishop. *Neural networks for pattern recognition*. Oxford University Press, 1995. (Cited on page 137)

A.P. Bradley. The use of the area under the ROC curve in the evaluation of machine learning algorithms. *Pattern Recognition*, 30(7):1145–1159, 1997. (Cited on pages 31, 32, and 43)

P. Brazdil, C Giraud-Carrier, C. Soares, and R. Vilalta. *Metalearning: Applications to data mining*. Cognitive Technologies. Springer-Verlag Berlin Heidelberg, 2009. (Cited on pages 100 and 103)

M.M. Breunig, H.P. Kriegel, R.T. Ng, and J. Sander. LOF: Identifying density-based local outliers. *ACM SIGMOD Record*, 29(2):93–104, 2000. (Cited on pages 10, 43, 46, 84, 87, and 137)

V. Chandola, A. Banerjee, and V. Kumar. Outlier detection: A survey. *ACM Computing Surveys*, 41(3):1–72, July 2009. (Cited on page 7)

T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein. *Introduction to algorithms*. MIT Press, third edit edition, Sept. 2009. (Cited on page 78)

D.R. Davies and R. Parasuraman. *The psychology of vigilance*. Academic Press, London, United Kingdom, 1982. (Cited on page 3)

D. de Ridder, D.M.J. Tax, and R.P.W. Duin. An experimental comparison of one-class classification methods. In *Proceedings of the 4th Annual Conference of the Advanced School for Computing and Imaging*, pages 213–218, Delft, The Netherlands, 1998. ASCI. (Cited on pages 44 and 45)

G.K.D. de Vries. *Kernel methods for vessel trajectories*. Ph.D. Thesis, Universiteit van Amsterdam, Amsterdam, 2012. (Cited on page 22)

G.K.D. de Vries, V. Malaisé, M. van Someren, P.W. Adriaans, and G. Schreiber. Semi-automatic ontology extension in the maritime domain. In A. Nijholt, M. Pantic, M. Poel, and H. Hondorp, editors, *Proceedings of The 20th Belgian-Netherlands Conference on Artificial Intelligence*, pages 265–272, Enschede, The Netherlands, 2008. (Cited on page 179)

J. Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006. (Cited on pages 34, 35, 36, and 92)

R.P.W. Duin, M. Loog, E. Pękalska, and D.M.J. Tax. *Feature-based dissimilarity space classification*. Lecture Noes in Computer Science. Springer Berlin / Heidelberg, 2010. (Cited on pages 21 and 22)

Eclipse. *Rich Client Platform*, available from: http://www.eclipse.org. Accessed June 2010. (Cited on page 179)

Embedded Systems Institute. *The Poseidon project: System evolvability and reliability of systems of systems*, available from: http://www.esi.nl/projects/poseidon. Accessed June 2010. (Cited on pages 177 and 179)

T. Fawcett. ROC graphs: Notes and practical considerations for researchers. *Machine Learning*, 31:1–38, Mar. 2004. (Cited on page 33)

U.M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth. Knowledge discovery and data mining: Towards a unifying framework. *Knowledge Discovery and Data Mining*, pages 82–88, 1996a. (Cited on page 43)

U.M. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. *Advances in knowledge discovery and data mining*. MIT Press, Cambridge, MA, 1996b. (Cited on page 21)

R.A. Fisher. The use of measurements in taxonomic problems. *Annals of Eugenics*, 7(2): 179–188, 1936. (Cited on page 29)

B.J. Frey and D. Dueck. Clustering by passing messages between data points. *Science (New York, N.Y.)*, 315(5814):972–6, Feb. 2007. ISSN 1095-9203. (Cited on pages 7 and 61)

M. Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association*, 32(200):675–701, Dec. 1937. (Cited on pages 34 and 43)

J. Gao and P.N. Tan. Converting output scores from outlier detection algorithms into probability estimates. In *Proceedings of the 6th International Conference on Data Mining*, volume 6, pages 212–221, Hong Kong, China, Dec. 2006. Ieee. ISBN 0-7695-2701-7. doi: 10.1109/ICDM.2006.43. (Cited on pages 61, 62, 75, and 95)

A.B. Gardner, A.M. Krieger, G. Vachtsevanos, and B. Litt. One-class novelty detection for seizure analysis from intracranial EEG. *Journal of Machine Learning Research*, 7: 1025–1044, June 2006. (Cited on page 4)

T. Gärtner, J.W. Lloyd, and P.A. Flach. Kernels and distances for structured data. *Machine Learning*, 57(3):205–232, Dec. 2004. (Cited on page 20)

C. Giraud-Carrier. Beyond predictive accuracy. In *Proceedings of the ECML-98 Workshop on Upgrading Learning to Meta-Level: Model Selection and Data*, 1998. (Cited on page 104)

J. Goldberger, S.T. Roweis, G.E. Hinton, and R. Salakhutdinov. Neighbourhood components analysis. In L.K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, pages 513–520, Cambridge, MA, 2005. MIT Press. (Cited on page 65)

A. González-Sanchez, É. Piel, and H.G. Groß. RiTMO: A method for runtime testability measurement and optimisation. In B. Choi, editor, *Proceedings of the 9th International Conference on Quality Software*, pages 377–382, Jeju, South Korea, 2009. IEEE Reliability Society. (Cited on page 179)

I. Guyon, J. Weston, S. Barnhill, and V.N. Vapnik. Gene selection for cancer classification using Support Vector Machines. *Machine Learning*, 46(1-3):389–422, 2002. (Cited on page 145)

K. Hempstalk and E. Frank. Discriminating against new classes: One-class versus multi-class classification. In *Proceedings of the 21st Australian Joint Conference on Artificial Intelligence*, pages 325–336, Auckland, New Zealand, 2008. Springer. (Cited on page 34)

S. Hido, Y. Tsuboi, H. Kashima, M. Sugiyama, and T. Kanamori. Inlier-based outlier detection via direct density ratio estimation. In *Proceedings of the 8th International Conference on Data Mining*, pages 223–232, Los Alamitos, CA, 2008. IEEE Computer Society. (Cited on pages 43, 84, and 88)

G.E. Hinton and S. Roweis. Stochastic Neighbor Embedding. In *Advances in Neural Information Processing Systems*, volume 17, pages 857–864, 2003. (Cited on pages 7, 61, 65, 66, and 81)

V.J. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22(2):85–126, Oct. 2004. (Cited on page 43)

R.L. Iman and J.M. Davenport. Approximations of the critical region of the Friedman statistic. *Communications in Statistics-Theory and Methods*, 9(6):571–595, 1979. (Cited on page 35)

N. Jankowski, W. Duch, and K. Grąbczewski, editors. *Meta-learning in computational intelligence*. Studies in Computational Intelligence. Springer-Verlag Berlin Heidelberg, 2011. (Cited on pages 100, 103, and 104)

J.H.M. Janssens. Outlier detection. *De Connectie*, 2009. (Cited on page 196)

J.H.M. Janssens. Ranking images on semantic attributes using human computation. In *NIPS Workshop on Computational Social Science and the Wisdom of Crowds*, Whistler, Canada, Dec. 2010. (Cited on pages 4 and 195)

J.H.M. Janssens and E.O. Postma. One-class classification with LOF and LOCI: An empirical comparison. In M.G.J. van Erp, J.H. Stehouwer, and M.M. van Zaanen, editors, *Proceedings of the 18th Annual Belgian-Dutch Conference on Machine Learning*, pages 56–64, Tilburg, The Netherlands, June 2009. (Cited on pages 46 and 196)

J.H.M. Janssens and E.O. Postma. Ranking images on semantic attributes using human computation. In *SNN Symposium: Intelligent Machines*, Nijmegen, The Netherlands, Nov. 2010. (Cited on page 196)

J.H.M. Janssens, I. Flesch, and E.O. Postma. Outlier detection with one-class classifiers from ML and KDD. In M.A. Wani, M. Kantardzic, V. Palade, L. Kurgan, and Y. Qi, editors, *Proceedings of the 8th International Conference on Machine Learning and Applications*, pages 147–153, Miami, FL, Dec. 2009. (Cited on pages 177, 179, and 196)

J.H.M. Janssens, H. Hiemstra, and E.O. Postma. Creating artificial vessel trajectories with Presto. In *Proceedings of the 22nd Benelux Conference on Artificial Intelligence*, Luxembourg, Luxembourg, Oct. 2010a. (Cited on page 196)

J.H.M. Janssens, E.O. Postma, and H.J. van den Herik. Unsupervised outlier selection with pairwise affinities. In *SNN Symposium: Intelligent Machines*, Nijmegen, The Netherlands, Nov. 2010b. (Cited on page 195)

J.H.M. Janssens, H. Hiemstra, and E.O. Postma. PRESTO: A Poseidon research tool to create artificial vessel trajectories. In J.H.M. Janssens and E.O. Postma, editors, *Proceedings of the 1st International Workshop on Maritime Anomaly Detection (MAD 2011)*, pages 43–44, Tilburg, The Netherlands, June 2011a. (Cited on page 195)

J.H.M. Janssens, E.O. Postma, and J.W. Hellemons. *Proceedings of the first international workshop on Maritime Anomaly Detection (MAD 2011)*. Tilburg, The Netherlands, 2011b. URL http://mad.uvt.nl/program. (Cited on pages 184 and 196)

J.H.M. Janssens, F. Huszar, E.O. Postma, and H.J. van den Herik. Stochastic Outlier Selection. Technical Report TiCC TR 2012-001, Tilburg University, Tilburg, The Netherlands, 2012. (Cited on page 195)

J.H.M. Janssens, E.O. Postma, and H.J. van den Herik. Density-based anomaly detection in the maritime domain. In P. van de Laar, J. Tretmans, and M. Borth, editors, *Situation Awareness with Systems of Systems*, pages 119–131. Springer Science+Business Media, New York, NY, 2013. ISBN 978-1-4614-6229-3. (Cited on page 195)

N. Japkowicz, C. Myers, and M. Gluck. A novelty detection to approach to classification. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, pages 518–523, 1995. (Cited on page 137)

F. Jelinek, R.L. Mercer, L.R. Bahl, and J.K Baker. Perplexity–a measure of the difficulty of speech recognition tasks. *Journal of the Acoustal Society of America*, 62(S1):S63, 1977. (Cited on page 81)

P. Juszczak, D.M.J. Tax, E. Pękalska, and R.P.W. Duin. Minimum spanning tree based one-class classifier. *Neurocomputing*, 72(7-9):1859–1869, Mar. 2009. ISSN 09252312. doi: 10.1016/j.neucom.2008.05.003. URL http://linkinghub.elsevier.com/retrieve/pii/S0925231208003238. (Cited on page 137)

A. Kalousis, J. Gama, and M. Hilario. On data and algorithms: Understanding inductive performance. *Machine Learning*, 54:275–312, 2004. (Cited on pages 100 and 103)

T. Kanamori, S. Hido, and M. Sugiyama. A least-squares approach to direct importance estimation. *Journal of Machine Learning Research*, 10:1391–1445, 2009. (Cited on pages 84 and 88)

R.D. King, C. Feng, and A. Sutherland. StatLog: Comparison of classification algorithms on large real-world problems. *Applied Artificial Intelligence*, 9(3):289–333, 1995. (Cited on pages 100, 103, and 114)

R. Kohavi. A study of cross-validation and bootstrap for accuracy estimation and model selection. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1995. (Cited on pages 38 and 113)

T Kohonen. *Self-organizing maps*. Springer-Verlag Berlin Heidelberg, third edition, 2001. (Cited on page 137)

C. Köpf, C.C. Taylor, and J. Keller. Meta-analysis: From data characterisation for meta-learning to meta-regression. In *Proceedings of the PKDD-00 Workshop on Data Mining, Decision Support, Meta-Learning and ILP: Forum for Practical Problem Presentation and Prospective Solutions*, pages 15–26, 2000. (Cited on pages 114, 115, and 117)

M.A. Kraaijveld and R.P.W. Duin. A criterion for the smoothing parameter for Parzen-estimators of probability density functions. Technical report, Delft University of Technology, 1991. (Cited on page 137)

H.P. Kriegel, P. Kröger, E. Schubert, and S.A. Zimek. LoOP: Local outlier probabilities. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pages 1649–1652. ACM, 2009. ISBN 9781605585123. (Cited on page 61)

H.P. Kriegel, P. Kröger, E. Schubert, and S.A. Zimek. Interpreting and unifying outlier scores. In *Proceedings of the 11th SIAM International Conference On Data Mining*, 2011. (Cited on pages 62 and 75)

G.R.G. Lanckriet, L. El Ghaoui, C. Bhattacharyya, and M.I. Jordan. Minimax probability machine. *Advances in Neural Information Processing Systems*, 2002. (Cited on page 137)

J. Lee and C. Giraud-Carrier. Predicting algorithm accuracy prediction with a small set of effective meta-features. In *Proceedings of the 7th International Conference on Machine Learning and Applications*, pages 808–812, 2008. (Cited on page 104)

J.W. Lee and C. Giraud-Carrier. A metric for unsupervised metalearning. *Intelligent Data Analysis*, 15:827–841, 2011. (Cited on page 104)

H.W. Lilliefors. On the Kolmogorov-Smirnov test for normality with mean and variance unknown. *Journal of the American Statistical Association*, 62:399–402, 1967. (Cited on page 116)

D.J.C. MacKay. *Information theory, inference, and learning algorithms*. Cambridge University Press, 2003. ISBN 0521642981. (Cited on page 81)

MÄK. *VR-Forces: The complete simulation toolkit*, available from: http://www.mak.com/products/vrforces.php. Accessed June 2010. (Cited on page 177)

R.B. Marimont and M.B. Shapiro. Nearest neighbour searches and the curse of dimensionality. *IMA Journal of Applied Mathematics*, 24(1):59–70, 1979. (Cited on page 110)

J. Mingers. An empirical comparison of selection measures for decision-tree induction. *Machine Learning*, 3:319–342, 1989. (Cited on page 117)

T.M. Mitchell. *Machine learning*. Series in Computer Science. McGraw-Hill, Columbus, OH, 1997. (Cited on page 25)

D. Mitchie, D.J. Spiegelhalter, and C.C. Taylor, editors. *Machine learning, neural and statistical classification*. Ellis Horwood Series in Artificial Intelligence. Prentice Hall, London, United Kingdom, 1994. (Cited on pages 100, 103, and 104)

National Aeronautics and Space Administration. *NASA World Wind: Java SDK*, available from: http://worldwind.arc.nasa.gov/java. Accessed June 2010. (Cited on page 179)

P.F. Neményi. *Distribution-free multiple comparisons*. Ph.D. Thesis, Princeton, 1963. (Cited on pages 35, 43, and 92)

S. Papadimitriou, H. Kitagawa, P.B. Gibbons, and C. Faloutsos. LOCI: Fast outlier detection using the local correlation integral. In *Proceedings of the 19th International Conference on Data Engineering*, pages 315–326, Bangalore, India, Mar. 2003. (Cited on pages 10, 43, 46, 48, 49, 51, 84, 87, and 94)

L. Parra, G. Deco, and S. Miesbach. Statistical independence and novelty detection with information preserving nonlinear maps. *Neural Computation*, 8(2):260–269, 1996. (Cited on page 137)

E. Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, pages 1065–1076, 1962. (Cited on pages 43, 44, 82, and 137)

K. Pearson. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine*, 2(11):559–572, 1901. (Cited on page 110)

Y. Peng, P.A. Flach, C. Soares, and P. Brazdil. Improved dataset characterisation for meta-learning. In *Discovery Science*, Lecture Notes in Computer Science, pages 193–208. Springer Berlin / Heidelberg, 2002. (Cited on pages 100, 103, 104, 117, and 118)

E. Pękalska and R.P.W. Duin. Dissimilarity representations allow for building good classifiers. *Pattern Recognition Letters*, 23:943–956, 2002. (Cited on page 22)

E. Pękalska and R.P.W. Duin. *The dissimilarity representation for pattern recognition: Foundations and applications*. World Scientific, Singapore, 2005. (Cited on page 21)

E. Pękalska, D.M.J. Tax, and R.P.W. Duin. One-class LP classifier for dissimilarity representations. In *Advances in Neural Information Processing Systems*, Cambridge, MA, 2003. MIT Press. (Cited on page 137)

J.R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986. (Cited on page 117)

L.A. Rendell and H.H. Cho. Empirical learning as a function of concept character. *Machine Learning*, 5(3):267–298, 1990. (Cited on page 103)

J.R. Rice. The algorithm selection problem. *Advances in Computers*, 15:65–118, 1976. (Cited on pages 8, 100, and 102)

A. Robin, L. Moisan, and S. Le Hégarat-Mascle. An a-contrario approach for subpixel change detection in satellite imagery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(11):1977–1993, Nov. 2010. (Cited on page 4)

S.M. Ross. *Introduction to probability models*. Elsevier, 9th editio edition, 2007. (Cited on page 78)

S.T. Roweis and L.K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–6, Dec. 2000. ISSN 0036-8075. (Cited on page 61)

J.S. Russel and P. Norvig. *Artificial intelligence: A modern approach*. Pearson Education, third inte edition, 2010. (Cited on page 3)

B. Schölkopf and A.J. Smola. *Learning with kernels*. MIT Press, Cambridge, MA, 2002. (Cited on pages 43 and 45)

C.E. Shannon. The mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948. ISSN 0724-6811. (Cited on page 81)

K.A. Smith-Miles. Cross-disciplinary perspectives on meta-learning for algorithm selection. *ACM Computing Surveys*, 41(1):6:1–6:24, Dec. 2008. (Cited on pages 8, 100, 103, and 104)

S.Y. Sohn. Meta analysis of classification algorithm for pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1137–1144, Nov. 1999. (Cited on pages 100, 103, 114, and 115)

Q. Song, G. Wang, and C. Wang. Automatic recommendation of classification algorithms based on data set characteristics. *Pattern Recognition*, 45:2672–2689, 2012. (Cited on pages 100, 101, 103, and 104)

D.M.J. Tax. *One-class classification: Concept-learning in the absence of counter-examples*. Ph.D. Thesis, Delft University of Technology, Delft, The Netherlands, June 2001. (Cited on pages 6, 29, 37, 43, 44, 45, 56, 84, 87, 112, and 137)

D.M.J. Tax. DDtools, The Data Description Toolbox for MATLAB. Version 1.9.1, 2012. URL http://prlab.tudelft.nl/david-tax/dd_tools.html. (Cited on pages 10 and 113)

D.M.J. Tax and R.P.W. Duin. Support vector domain description. *Pattern Recognition Letters*, 20(11-13):1191–1199, 1999. (Cited on pages 42, 43, 45, and 137)

D.M.J. Tax, P. Juszczak, E. Pękalska, and R.P.W. Duin. Outlier detection using ball descriptions with adjustable metric. In D.-Y. Yueng, J.T. Kwok, A.L.N. Fred, F. Roli, and D. de Ridder, editors, *Structural, Syntactic, and Statistical Pattern Recognition, Proc. SSSPR2006 (Hong Kong, China, August 2006)*, volume 4109 of *Lecture Notes in Computer Science*, pages 587–595. Springer Verlag, Berlin, 2006. (Cited on page 137)

J.B. Tenenbaum, V. de Silva, and J.C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–23, Dec. 2000. ISSN 0036-8075. (Cited on page 61)

L.J.P. van der Maaten. *Feature extraction from visual data*. Ph.D. Thesis, Tilburg University, Tilburg, The Netherlands, June 2009a. (Cited on page 21)

L.J.P. van der Maaten. Learning a parametric embedding by preserving local structure. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, volume 5, pages 384–391, 2009b. (Cited on pages 7 and 61)

L.J.P. van der Maaten and G.E. Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. (Cited on pages 61, 65, 66, 82, and 125)

M.G.J. van Erp. *Accessing natural history: Discoveries in data cleaning, structuring, and retrieval*. Ph.D. Thesis, Tilburg University, Tilburg, The Netherlands, June 2010. (Cited on page 21)

S. Vanderlooy. *Ranking and reliable classification*. Ph.D. Thesis, Universiteit Maastricht, Maastricht, The Netherlands, 2009. (Cited on pages 23 and 32)

V.N. Vapnik. *The nature of statistical learning theory*. Springer, New York, NY, 1995. (Cited on page 45)

R. Vilalta and Y. Drissi. A perspective view and survey of meta-learning. *Artificial Intelligence Review*, 18:77–95, 2002. (Cited on pages 103 and 104)

G.I. Webb. Multiboosting: A technique for combining boosting and wagging. *Machine Learning*, 40:159–197, 2000. (Cited on page 34)

P. Welinder, S. Branson, S. Belongie, and P. Perona. The multidimensional wisdom of crowds. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2424–2432, 2010. (Cited on page 4)

C. Whitrow, D.J. Hand, P. Juszczak, D. Weston, and N.M. Adams. Transaction aggregation as a strategy for credit card fraud detection. *Data Mining and Knowledge Discovery*, 18(1):30–55, 2009. (Cited on page 4)

C.M.E. Willems, W.R. van Hage, G.K.D. de Vries, J.H.M. Janssens, and V. Malaisé. An integrated approach for visual analysis of a multi-source moving objects knowledge base. *International Journal of Geographical Information Science*, 24(10):1543–1558, 2010a. (Cited on pages 179 and 195)

C.M.E. Willems, W.R. van Hage, G.K.D. de Vries, J.H.M. Janssens, and V. Malaisé. An integrated approach for visual analysis of a multi-source moving objects knowledge base. In G. Andrienko, N. Andrienko, J. Dykes, M-J. Kraak, and H. Schumann, editors, *Geospatial Visual Analytics: Focus on Time*, Guimarães, Portugal, May 2010b. (Cited on page 196)

I.H. Witten and E. Frank. *Data mining: Practical machine learning tools and techniques*. Series in Data Management Systems. Morgan Kaufmann Publishers, San Francisco, CA, 2nd edition, 2005. (Cited on pages 20 and 25)

D.H. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1341–1390, 1996a. (Cited on page 101)

D.H. Wolpert. The existence of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1391–1420, 1996b. (Cited on page 101)

D.H. Wolpert. The supervised learning no-free-lunch theorems. In *Proceedings of the 6th Online World Conference on Soft Computing in Industrial Applications*, 2001. (Cited on page 101)

D.H. Wolpert and W.G. Macready. No free lunch theorems for search. Technical report, The Santa Fe Institute, Santa Fe, NM, 1995. (Cited on pages 11 and 101)

A. Ypma. *Learning methods for machine vibration analysis and health monitoring*. Ph.D. Thesis, Delft University of Technology, Delft, The Netherlands, 2001. (Cited on pages 52, 89, and 105)

A. Ypma and R.P.W. Duin. Support objects for domain approximation. In L.F. Niklassen, M.B. Bodén, and T. Ziemke, editors, *Proceedings of the 8th International Conference on Artificial Neural Networks*, pages 719–724, Skövde, Sweden, 1998. Springer. (Cited on page 137)

A. Zellner. Bayesian estimation and prediction using asymmetric loss functions. *Journal of the American Statistical Association*, 81(394):446–451, June 1986. ISSN 01621459. (Cited on page 80)

Appendices

Contents:

During the time spent at Tilburg University we completed two projects that are related to the Poseidon project: (1) Presto and (2) MAD 2011. Because we could not fit the projects in any of the chapters, we highlight them in the following two appendices. In Appendix A we describe a software application called Presto. Presto allows experts from the maritime domain to create artificial anomalies. Appendix B discusses the first international workshop on Maritime Anomaly Detection, in short: MAD 2011. The workshop was held on June 17, 2011, and featured international keynote speakers, presentions, posters, and attendants from eleven countries.

Based on:

- C.M.E. Willems, W.R. van Hage, G.K.D. de Vries, J.H.M. Janssens, and V. Malaisé. An integrated approach for visual analysis of a multi-source moving objects knowledge base. *International Journal of Geographical Information Science*, 24(10):1543–1558, 2010a.
- J.H.M. Janssens, H. Hiemstra, and E.O. Postma. Creating artificial vessel trajectories with Presto. In *Proceedings of the 22nd Benelux Conference on Artificial Intelligence*, Luxembourg, Luxembourg, Oct. 2010a.
- J.H.M. Janssens, H. Hiemstra, and E.O. Postma. PRESTO: A Poseidon research tool to create artificial vessel trajectories. In J.H.M. Janssens and E.O. Postma, editors, *Proceedings of the 1st International Workshop on Maritime Anomaly Detection (MAD 2011)*, pages 43–44, Tilburg, The Netherlands, June 2011a.

Outline:

A Presto: A Poseidon research tool to create maritime anomalies. **B** MAD 2011: The first international workshop on maritime anomaly detection.

Presto: a Poseidon research tool to create maritime anomalies

The automatic detection of anomalies in the maritime domain requires representative anomalous instances. We developed the Poseidon Research Tool (Presto), an application that enables maritime-domain experts to create artificial anomalous vessel trajectories, characteristic of traffic violations, illegal fishing activities, drug smuggling, or piracy. When merged with existing real-world data the artificial trajectories enable the application and evaluation of machine learning algorithms for the automatic detection of anomalies.

A.1 The need for maritime anomalies

A Maritime Safety and Security (MSS) system aims at guiding a surveillance operator to find maritime anomalies such as vessel traffic violations, illegal fishing activities, and drug smuggling (Embedded Systems Institute, available from: http://www.esi.nl/projects/poseidon). In real-world data, serious maritime anomalies rarely occur. Because the MSS system might be deployed in any maritime circumstance, we need to evaluate if our machine learning algorithms (Janssens, Flesch, and Postma, 2009) can properly detect such anomalies. To this end, we have developed Presto, an application which enables maritime domain-experts to create easily artificial scenarios. In contrast to existing simulation applications, such as VR-Forces (MÄK, available from: http://www.mak.com/products/vrforces.php), which impose restrictive behaviour models, our application gives the expert full control over the vessel trajectories.

A.2 Overview of Presto

The main concept in Presto is a scenario, which can contain one or more trajectories. Each trajectory is defined by several waypoints. A waypoint is a location on the world map and contains three additional parameters: velocity, time, and curvature. Figure A.1 on the next page shows the user interface of Presto, which consists of three main elements: (A) the world map, (B) the timeline, and (C) the property editor. Using the world map, the user can edit the position of the trajectories and their waypoints. The timeline gives an

overview of the trajectories within the scenario and is used to navigate through time. The property editor with its three tabs "scenario", "track", and "waypoint", is used to edit element-specific parameters (e.g., for a scenario: name and description; for a trajectory: name and flag; and for a waypoint: location and velocity).

A trajectory is generated from the waypoints using the following iterative procedure. Given a location, velocity, and time as provided by the current waypoint, the vessel starts sailing towards the next waypoint for a user-defined time step. Thereafter, the location, velocity, and the bearing of the vessel are updated. This is repeated until the vessel reaches the next waypoint. When the vessel passes the waypoint within the time step, all variables are adjusted such that the vessel continues exactly at the waypoint. Rather than using a linear interpolation for the position between two waypoints, which result in unnaturally sharp turns, we employ Bézier interpolation in the Euclidean space mapped back to the Earth's surface. The current and next waypoints are on the path in the Bézier interpolation, whereas the (invisible) control points are placed at a location defined by the curvature variable. A lower curvature value causes the control points to be placed further away from the waypoints, resulting in a smoother and larger turn of the vessel. The velocity between waypoints is interpolated using exponential easing.

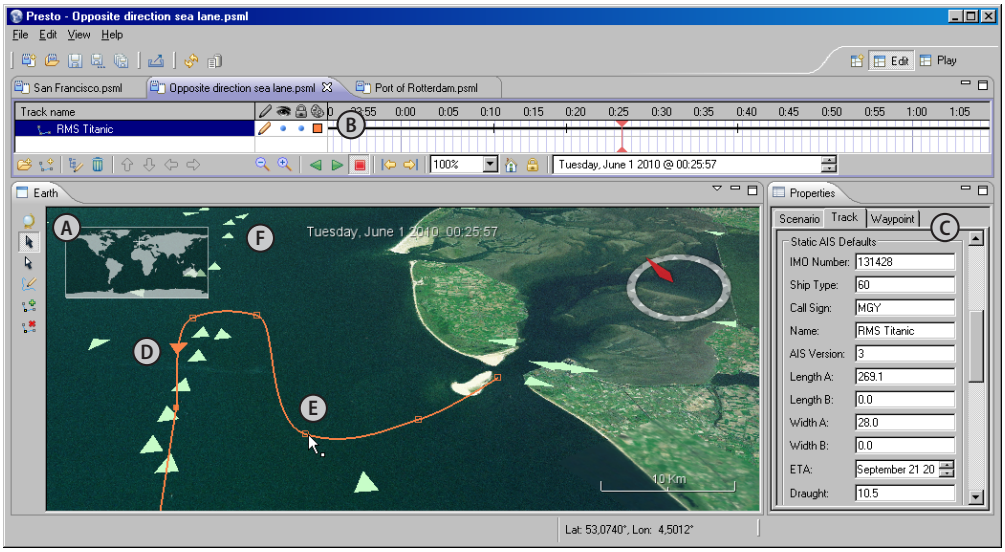


Figure A.1 A screenshot illustrating the different user interface elements and concepts of Presto: (a) the world map, (b) the timeline showing the artificial trajectory in the scenario, (c) the property editor showing the trajectory properties, (d) the current location of the artificial vessel along the created trajectory, (e) one waypoint of the trajectory, and (f) the background data at the current moment.

Presto has the ability to load existing, real vessel-trajectory data as so-called background data. This background data contains the positions and bearings of all vessels for a certain period (e.g., a day). It serves as a reference when creating new artificial trajectories, so that, for instance, collisions can be avoided or enforced. The generated data is used to create real-world vessel-trajectory data by filtering the appropriate variables such as location, velocity, and bearing. The filtering is performed in such a way that data points mimic messages according the Automatic Identification System (AIS) protocol. The exported data is fused with real-world data, such that the artificial data cannot be distinguished up front from the original data.Besides the AIS format, Presto can export to CSV, GPX, and KML for integration with existing GIS software, such as Google Earth.

Presto is entirely programmed in Java, and makes extensive use of the Eclipse Rich Client Platform (Eclipse, available from: http://www.eclipse.org). It is open source and can be downloaded from https://github.com/jeroenjanssens/presto. The world map is based on NASA World Wind, an open-source alternative to Google Earth (National Aeronautics and Space Administration, available from: http://worldwind.arc.nasa.gov/java). The use of these software solutions makes Presto suitable for recent versions of Windows, Mac OS X, and Linux-based operating systems. The world map requires a hardware-accelerated 3D graphics card and an internet connection for downloading the high-resolution terrain images, although these can also be cached locally.

A.3 Academic and industrial adoption of Presto

Presto is successfully used by researchers within the Poseidon project (Embedded Systems Institute, available from: http://www.esi.nl/projects/poseidon) and domain-experts at the Maritime Safety and Security division of Thales Netherlands. It has proved to be a useful research tool for tasks including: anomaly detection (Janssens et al., 2009), ontology engineering (de Vries, Malaisé, van Someren, Adriaans, and Schreiber, 2008), runtime testing (González-Sanchez, Piel, and Groß, 2009), and visual vessel analysis (Willems, van Hage, de Vries, Janssens, and Malaisé, 2010a). Additionally, Presto has enabled domain-experts to discuss and exchange potential maritime situations, and to challenge researchers with anomalies that are not present in real-world data. In future work we plan to implement an additional import and export functionality such that Presto can be integrated more seamlessly with existing applications that are used within the MSS domain.

MAD 2011: the first international workshop on maritime anomaly detection

As part of the Poseidon project, the author organised, together with Eric Postma and Joke Hellemons from the Tilburg center for Cognition and Communication (TiCC), the first international workshop on Maritime Anomaly Detection (MAD 2011). The workshop was kindly sponsored by the School for Information and Knowledge Systems (SIKS). Figure B.1 shows the logo of the workshop.

Figure B.1 Logo of the MAD internal workshop.

The goal of MAD 2011, which was held on June 17 2011, was to bring together researchers from academia and industry in a stimulating discussion on state-of-the-art algorithms for maritime anomaly detection and future directions of research and development.

The field of maritime anomaly detection encompasses the detection of suspicious or abnormal vessel behaviour in potentially very large datasets or data streams (e.g., radar and AIS data streams). To support operators in the detection of anomalies, methods from machine learning, data mining, statistics, and knowledge representation may be used. The scientific study of anomaly detection brings together researchers in computer science, artificial intelligence, and cognitive science. For the maritime domain, anomaly detection plays a crucial role in supporting maritime surveillance and in enhancing situation awareness. Typical applications include the prediction of collisions, the detection of vessel traffic violations, and the assessment of potential threats.

Table B.1 Program of the MAD 2011.

Start	Duration	Title	Authors
9:00	0:30	*Registration*	
9:30	0:15	Opening	Jaap van den Herik, Tilburg center for Cognition and Communication, The Netherlands
9:45	0:45	Keynote: Towards Maritime Behavior Recognition and Anomaly Detection	David Aha, Naval Research Laboratory, USA
10:30	0:30	Density Based, Visual Anomaly Detection	Roeland Scheepens, Niels Willems, Huub van de Wetering, and Jarke van Wijk, Eindhoven University of Technology, The Netherlands
11:00	0:30	ConTraffic: Maritime container traffic anomaly detection	Aristide Varfis, Evangelos Kotsakis, Aris Tsois, Maxym Sjachyn, Alberto Donati, Elena Camossi, Paola Villa, Tatyana Dimitrova, and Muriel Pellissier, European Commission, Joint Research Centre, Italy
11:30	0:30	Comparing Vessel Trajectories using Geographical Domain Knowledge and Alignments	Gerben de Vries, Universiteit van Amsterdam, The Netherlands; Willem Robert van Hage, Vrije Universiteit Amsterdam, The Netherlands; and Maarten van Someren, Universiteit van Amsterdam, The Netherlands
12:00	1:00	*Lunch*	
13:00	0:30	Keynote: Towards improving situation awareness for operators in the maritime domain	Maurice Glandrup, Thales Naval, The Netherlands
13:30	0:30	Spatio-Temporal Visualisation of Outliers	Laurent Etienne and Cyril Ray, Naval Academy Research Institute, France; and Gavin Mcardle, National Universtiy of Ireland Maynooth, Ireland
14:00	0:30	Maritime Anomaly Detection using Stochastic Outlier Selection	Jeroen Janssens, Eric Postma, and Jaap van den Herik, Tilburg University, The Netherlands
14:30	0:30	Maritime Route Anomaly Detection	Richard Lane, QinetiQ, United Kingdom
15:00	0:30	*Break*	
15:30	0:30	Applying V-Analytics to AIS Data	Gennady Andrienko and Natalia Andrienko, Fraunhofer Institute IAIS, Germany
16:00	0:30	An Evaluation of Fractal/Velocity Pattern Extraction	René Enguehard, Rodolphe Devillers, and Orland Hoeber, Memorial University of Newfoundland, Canada
16:30	0:30	Incremental Stream Clustering for Anomaly Detection in Maritime Surveillance	Anders Holst and Jan Ekman, Swedish Institute of Computer Science, Sweden
17:00	0:10	Poster pitches	
17:10	1:50	Poster session and drinks	
19:00	?	*Workshop banquet*	

We welcomed contributions in the form of case studies and contributions concerning anomaly detection in other domains that may be of relevance to the maritime domain.

Topics of interests included, but were not limited to:

- Anomaly, Outlier, and Novelty Detection
- Machine Learning / Pattern Recognition / Data Mining
- (Online) Active Learning / Human-in-the-Loop Approaches
- Feature Selection / Metric learning
- (Dis)similarity measures / kernels for vessel trajectories
- Generating artificial anomalous vessel trajectories
- Incorporating / Representing / Visualizing Maritime Domain Knowledge

The workshop program (see Table B.1) included two invited talks: one by David Aha (U.S. Naval Research Laboratory, Navy Center for Applied Research in Artificial Intelligence) and one by Maurice Glandrup (Thales Naval Netherlands, Above Water Systems). The contributed talks and posters were selected on the basis of quality and originality. Table B.2 lists the posters that were presented during the poster session.

Table B.2 Posters presented at the poster session.

Poster	Title	Authors
A	A Bayesian Network Approach to Maritime Situation Assessment	Yvonne Fischer, Karlsruhe Institute of Technology, Germany; and Joris Ijsselmuiden, Fraunhofer IOSB, Germany
B	Maritime Anomaly Detection by Fusing Sensor Information and Intelligence	Bert van den Broek, Martijn Neef, Patrick Hanckmann, Arthur Smith, TNO, The Netherlands; and Fok Bolderheij, NLDA, The Netherlands
C	Web-based Geographical Visualization of Container Itineraries	Tatyana Dimitrova and Evangelos Kotsakis, European Commission, Joint Research Centre, Italy
D	Finding Fraud in Health Insurance Data with Two-Layer Outlier Detection Approach	Rob Konijn and Wojtek Kowalczyk, Vrije Universiteit Amsterdam, The Netherlands
E	Semantic-based Anomalous Pattern Detection from Maritime Trajectories	Paola Villa and Elena Camossi, European Commission, Joint Research Centre, Italy
F	Detection of near misses and undesired encounters on the North Sea	Erwin van Iperen, MARIN, The Netherlands
G	PRESTO: A Poseidon Research Tool To Create Artificial Vessel Trajectories	Jeroen Janssens, Tilburg University, The Netherlands; Hans Hiemstra, Thales Naval, The Netherlands; and Eric Postma, Tilburg University, The Netherlands

The accepted abstracts have been published in the MAD workshop proceedings (Janssens, Postma, and Hellemons, 2011b), which can be found on-line at http://mad.uvt.nl/program. With eighteen contributions from ten different countries, from both academia and industry, we may conclude that the MAD 2011 was a successful first international workshop on maritime anomaly detection.

Summary

An anomaly is a real-world observation or event that deviates from what is considered to be normal. Examples of anomalies are: a terrorist attack, a forged painting, and a rotten apple. Because anomalies can be dangerous or expensive, detecting them is of utmost importance. A domain expert may suffer from three cognitive limitations that hamper the detection of anomalies: (1) fatigue, (2) information overload, and (3) emotional bias.

Real-world observations can often be measured and recorded to form a data set. A suitable representation translates anomalies into outliers, which are data points that deviate quantitatively from the majority of other data points. Outlier-selection algorithms are capable of automatically classifying outliers in large data sets. In the thesis we study the problem statement: *To what extent can outlier-selection algorithms support domain experts with real-world anomaly detection?*

We consider a domain expert to be supported when the following three conditions are satisfied. First, a domain expert should know how to evaluate and compare the performance of the outlier-selection algorithms. Second, a domain expert should have one or more algorithms available that are considered state-of-the-art. Third, a domain expert should know when to apply which algorithm. The three conditions lead to three accompanying research questions: (RQ1) *How should we evaluate and compare the performance of outlier-selection algorithms?*, (RQ2) *Can an effective outlier-selection algorithm be devised that employs the concept of affinity?*, and (RQ3) *To what extent can we solve the one-class classifier selection problem using a meta-learning approach?* The answers to these three research questions enable us to formulate an answer to the problem statement.

In Chapter 2 we introduce the main concepts and explain the experimental set-up that is employed for answering RQ1, RQ2, and RQ3. This is partly based on a review and the analysis of the relevant literature concerning anomaly detection, outlier selection, and machine learning. The chapter consists of four parts. First, we present two outlier-selection settings that we consider in the thesis: (1) unsupervised outlier selection and (2) one-class classification. We highlight their similarities and differences, and we make clear when which setting is appropriate. Second, we describe two types of data representations that outlier-selection algorithms can process: (1) feature vector representation and (2)

similarity matrix representation. Third, we explain in detail the experimental set-up. It relies on statistical techniques such as (1) the Area Under the ROC Curve (AUC) performance measure, (2) the post-hoc Neményi statistical test, and (3) cross validation. Fourth, we explain how to transform ordinary multi-class data sets into one-class data sets so that they can be used for our comparative experiments.

In Chapter 3 we answer RQ1. We describe how anomalies are detected in the fields of Machine Learning (ML) and Knowledge Discovery in Databases (KDD). Both fields have their own anomaly-detection algorithms and corresponding evaluation procedures. Two well-known outlier-selection algorithms from the field of KDD are framed into the one-class classification framework so that these can be compared with three algorithms from the field of ML in a statistically valid way. We perform a comparative evaluation of the ML and KDD outlier-selection algorithms on real-world datasets and discuss the results.

In Chapter 4, we provide an answer to RQ2. We present a novel, unsupervised algorithm for classifying outliers, called Stochastic Outlier Selection (SOS). SOS uses the concept of affinity to compute for each data point an outlier probability. The probabilities that are computed by SOS provide several advantages with respect to the unbounded outlier scores as computed by related algorithms. Using outlier-score plots, we illustrate and discuss the qualitative performances of SOS and four related algorithms. Then we evaluate SOS and the four algorithms on eighteen real-world data sets and seven synthetic data sets. The results obtained on these 25 data sets show that SOS (1) has a significantly higher performance and (2) is more robust to data perturbations and varying densities than the four related algorithms. From these results we may conclude that SOS is an effective algorithm for classifying outliers.

The research required to formulate an answer to RQ3 is split over the Chapters 5 and 6. In Chapter 5, we discuss the No Free Lunch theorem, which states that there is no single best one-class classifier. For each one-class data set, there may be a different one-class classifier that performs best. The performance of a one-class classifier is greatly determined by the characteristics, or meta-features, of the one-class data set. We present three methods for preprocessing data that may improve the performance of certain one-class classifiers on certain one-class data sets. The results in this chapter are obtained in three steps. First, we implement 36 meta-features. Second, we apply the meta-features to 255 data sets. Third, we select empirically the most informative meta-features. The selected meta-features are

used as input for Chapter 6, where we use meta-learning to relate the meta-features to the one-class classifier performance.

In Chapter 6, we provide an answer to RQ3 by using the output from Chapter 5. First, we describe 19 one-class classifiers. Subsequently, we perform a comparative experiment by applying the 19 one-class classifiers on the 255 one-class data sets from Chapter 5. To answer RQ3, we aim to understand the relationship between the previously computed meta-features and the one-class classifier performances. To this end, we set up a meta-learning experiment, where we aim to solve the one-class classifier selection problem, i.e., to predict the most appropriate one-class classifier for a given, unseen, one-class data set. The results indicate to what extent we can solve the one-class classifier selection problem by using a meta-learning approach.

In Chapter 7 we complete the thesis. We discuss the findings of the thesis on a general level and suggest four recommendations for future research. By reviewing the answers to the three research questions, we arrive at three conclusions. First, we demonstrated how to evaluate and compare the performance of outlier-selection algorithms and one-class classifiers. Second, we developed a new state-of-the art outlier-selection algorithm called SOS. Third, we have shown that a meta-learning approach can provide a domain expert with insight into when to apply which one-class classifier given a certain data set. Seeing that the three conditions mentioned in paragraph 3 have been satisfied, we may now answer our problem statement. Our conclusion reads that the domain expert can be supported by outlier-selection algorithms to some extent.

Samenvatting

Een anomalie is een natuurlijke observatie of gebeurtenis die afwijkt van wat als normaal wordt beschouwd. Voorbeelden van anomalieën zijn: een terroristische aanslag, een vervalst schilderij en een rotte appel. Omdat anomalieën gevaarlijk of duur kunnen zijn, is het van groot belang dat deze gedetecteerd worden. Een domeinexpert kan last hebben van drie cognitieve beperkingen die de detectie van anomalieën bemoeilijken: (1) vermoeidheid, (2) informatieoverbelasting en (3) emotionele vooringenomenheid.

Observaties kunnen vaak gemeten en opgenomen worden om zo een dataverzameling te vormen. Een geschikte representatie vertaalt anomalieën als uitschieters. Dat zijn datapunten die kwantitatief afwijken van de meeste andere datapunten. Uitschieterselectiealgoritmen kunnen automatisch uitschieters classificeren uit grote dataverzamelingen. In het proefschrift bestuderen we de probleemstelling: *In hoeverre kunnen uitschieterselectiealgoritmen domeinexperts ondersteunen bij het detecteren van natuurlijke anomalieën?*

We beschouwen een domeinexpert als ondersteund wanneer aan de volgende drie voorwaarden wordt voldaan. Ten eerste zal een domeinexpert moeten weten hoe uitschieterselectiealgoritmen geëvalueerd en vergeleken dienen te worden. Ten tweede zal een domeinexpert één of meerdere effectieve algoritmen tot zijn of haar beschikking moeten hebben. Ten derde zal een domeinexpert moeten weten wanneer welk algoritme gebruikt dient te worden. Deze drie voorwaarden leiden tot drie bijbehorende onderzoeksvragen: (OV1) *Hoe dienen we de prestaties van uitschieterselectiealgoritmen te evalueren en vergelijken?*, (OV2) *Kan er een effectief uitschieterselectiealgoritme worden ontwikkeld dat het concept van affiniteit gebruikt?* en (OV3) *In hoeverre kunnen we het enkelklasseclassificeerderselectieprobleem oplossen met behulp van een metaleeraanpak?* De antwoorden op deze drie onderzoeksvragen stellen ons in staat om een antwoord te formuleren op de probleemstelling.

In Hoofdstuk 2 introduceren we de voornaamste concepten en leggen we de experimentele aanpak uit die wordt gehanteerd voor het beantwoorden van OV1, OV2 en OV3. Dit is gedeeltelijk gebaseerd op een studie en analyse van de relevante wetenschappelijke literatuur aangaande anomaliedetectie, uitschieterselectie en machinaal leren. Het hoofdstuk bestaat uit vier delen. Ten eerste presenteren we twee uitschieterselectiesituaties die we in het proefschrift behandelen: (1) ongecontroleerde

uitschieterselectie en (2) enkelklasseclassificatie. We leggen de nadruk op hun gelijkenissen en verschillen, en leggen uit wanneer welke situatie van toepassing is. Ten tweede beschrijven we twee typen datarepresentaties die uitschieterselectiealgoritmen kunnen verwerken: (1) eigenschapvectorrepresentatie en (2) gelijkenismatrixrepresentatie. Ten derde leggen we in detail de experimentele opzet uit. Die berust op statistische technieken zoals (1) de AUC prestatiemaat, (2) de post-hoc Nemányi statistische test en (3) kruisvalidatie. Ten vierde leggen we uit hoe gewone multiklassedataverzamelingen in meerdere enkelklassedataverzamelingen te transformeren zijn zodat deze te gebruiken zijn in onze vergelijkende experimenten.

In Hoofdstuk 3 beantwoorden we OV1. We beschrijven hoe anomalieën gedetecteerd worden in de onderzoeksgebieden Machinaal Leren (ML) en Kennis Ontdekking in Databases (KOD). Beide onderzoeksgebieden hebben hun eigen anomaliedetectiealgoritmen en bijbehorende evaluatieprocedures. Twee bekende uitschieterselectiealgoritmen van het onderzoeksgebied KOD zijn in het enkelklasseclassificatieraamwerk gevat zodat deze op een statistisch valide wijze vergeleken kunnen worden met drie algoritmen van het onderzoeksgebied ML. We voeren een vergelijkende evaluatie van de ML en KOD uitschieterselectiealgoritmen uit op natuurlijke dataverzamelingen en bespreken de resultaten.

In Hoofdstuk 4 geven we een antwoord op OV2. We presenteren een nieuw, ongecontroleerd algoritme voor het classificeren van uitschieters, genaamd *Stochastic Outlier Selection* (SOS). SOS gebruikt het concept affiniteit om voor ieder datapunt een uitschieterkans te berekenen. De kansen die berekend worden door SOS bieden verscheidene voordelen ten opzichte van de ongelimiteerde uitschieterscores die berekend worden door andere algoritmen. Door middel van uitschieterscorediagrammen illustreren en bespreken we de kwalitatieve prestaties van SOS en vier andere algoritmen. Daarna evalueren we SOS en de vier algoritmen op achttien natuurlijke dataverzamelingen en zeven kunstmatige dataverzamelingen. De behaalde resultaten op deze 25 dataverzamelingen laten zien dat SOS (1) een significant hogere prestatie heeft en (2) beter kan omgaan met dataverstoringen en variërende datadichtheden dan de vier andere algoritmen. Met deze resultaten mogen we concluderen dat SOS een effectief algoritme is voor het classificeren van uitschieters.

Het onderzoek dat nodig is om een antwoord te formuleren voor OV3 is verdeeld over de Hoofdstukken 5 en 6. In Hoofdstuk 5 bespreken we de *no-free-lunch* stelling, welke inhoudt dat er geen beste enkelklasseclassificeerder is. Voor iedere

enkelklassedataverzameling kan er een andere enkelklasseclassificeerder zijn die het beste presteert. De prestatie van een enkelklasseclassificeerder wordt sterk bepaald door de karakteristieken, of meta-eigenschappen, van de enkelklassedataverzameling. We presenteren drie datavoorverwerkingsmethoden die de prestaties van bepaalde enkelklasseclassificeerders op bepaalde enkelklassedataverzamelingen kunnen verbeteren. De resultaten in dit hoofdstuk worden in drie stappen behaald. Ten eerste implementeren we 36 meta-eigenschappen. Ten tweede passen we de meta-eigenschappen toe op 255 dataverzamelingen. Ten derde selecteren we op empirische wijze de meest informatieve meta-eigenschappen. De geselecteerde meta-eigenschappen dienen als invoer voor Hoofdstuk 6, waar we metaleren gebruiken om de meta-eigenschappen te relateren aan de prestaties van de enkelklasseclassificeerders.

In Hoofdstuk 6 geven we een antwoord op OV3 door gebruik te maken van de resultaten van Hoofdstuk 5. Ten eerste beschrijven we negentien enkelklasseclassificeerders. Daarna voeren we een vergelijkend experiment uit door de negentien enkelklasseclassificeerders toe te passen op de 255 enkelklassedataverzamelingen van Hoofdstuk 5. Om OV3 te beantwoorden, zetten we een metaleerexperiment op waarmee we trachten om het enkelklasseclassificeerderselectieprobleem op te lossen, dat betekent, om de meest geschikte enkelklasseclassificeerder voor een gegeven, ongeziene, enkelklassedataverzameling te selecteren. Voorafgaande experimenten hebben aangetoond dat meta-eigenschappen die enkel op de normaalklasse zijn gebaseerd, niet-informatief zijn voor de prestatie van een enkelklasseclassificeerder. Daarom herberekenen we de meta-eigenschappen op basis van een variërend aantal anomalieën. De resultaten indiceren in hoeverre we het enkelklasseclassificeerderselectieprobleem kunnen oplossen met behulp van een metaleeraanpak.

In Hoofdstuk 7 besluiten we het proefschrift. We bespreken de bevindingen van het proefschrift op een algemeen niveau en suggereren vier aanbevelingen voor toekomstig onderzoek. Door het bezien van de antwoorden op de drie onderzoeksvragen komen we tot drie conclusies. Ten eerste hebben we gedemonstreerd hoe de prestaties van uitschieterselectiealgoritmen geëvalueerd en vergeleken dienen te worden. Ten tweede hebben we een nieuw, tot-de-top-behorend, uitschieterselectiealgoritme ontwikkeld, genaamd Stochastic Outlier Selection. Ten derde hebben we laten zien dat een metaleeraanpak een domeinexpert inzicht kan geven in wanneer welke enkelklasseclassificeerder moet worden toegepast op een gegeven dataverzameling. Gezien hebbende dat aan de voorwaarden beschreven in paragraaf 3 kan worden voldaan, mogen we nu onze

probleemstelling beantwoorden. Onze conclusie is dat de domeinexpert tot op zekere hoogte ondersteund kan worden door uitschieterselectiealgoritmen.

Curriculum vitae

Jeroen Janssens was born on June 6, 1983 in Helmond, the Netherlands. He completed his secondary education (HAVO, natuur en techniek) at the Carolus Borromeus College in Helmond. In 2002, Jeroen obtained his propedeutics in Computer Science at the Fontys Hogeschool in Eindhoven. The propedeutics degree, together with a colloquium doctum in Physics, allowed him to switch to the Eindhoven University of Technology to study Industrial Design.

Because Industrial Design did not appear to be the study Jeroen had hoped for, he decided to restart in September 2003 at the University College Maastricht (UCM). UCM is a so-called liberal arts college where the curriculum is composed by the student under guidance of an academic advisor. Jeroen chose to concentrate on subjects such as Computer Science, Psychology, and Mathematics. Several Artificial Intelligence courses were followed at the Department of Knowledge Engineering of Maastricht University. Moreover, six months were spent in Australia, where Jeroen studied one semester of Computer Science at the University of Adelaide. In June 2006, Jeroen obtained his Bachelor of Science degree, cum laude. He continued with a Master in Artificial Intelligence at Maastricht University, and obtained his degree in January 2008, also cum laude. His Master's thesis was titled "Collaborative Image Ranking: Bridging the Semantic Gap using Human Computation".

Immediately thereafter, Jeroen started a BSIK-funded Ph.D. project at the same university under the supervision of prof.dr. Eric Postma and prof.dr. Jaap van den Herik. In August 2008, he accompanied his supervisors to the Tilburg University and was appointed as Ph.D. researcher at TiCC. For his research, he developed techniques to detect automatically anomalous patterns in data. This research was conducted in the Poseidon project hosted by the Embedded Systems Institute in Eindhoven and in close cooperation with the industry represented by Thales Nederland in Hengelo and Noldus in Wageningen. He defended his Ph.D. thesis in 2013.

Currently, Jeroen is working as a Data Scientist at Visual Revenue, an Outbrain company, in New York, NY. Visual Revenue offers news publishers a platform that provides real-time recommendations on how to make the most of their front page and social media channels. Jeroen is mainly responsible for the predictive analytics aspect of the platform. For example, he develops machine learning algorithms that predict how often a news article will be read.

Publications

The scientific work performed during the author's Ph.D. research resulted in the following publications.

Journal articles and reports

1. J.H.M. Janssens, F. Huszar, E.O. Postma, and H.J. van den Herik. Stochastic Outlier Selection. Technical Report TiCC TR 2012-001, Tilburg University, Tilburg, The Netherlands, 2012
2. C.M.E. Willems, W.R. van Hage, G.K.D. de Vries, J.H.M. Janssens, and V. Malaisé. An integrated approach for visual analysis of a multi-source moving objects knowledge base. *International Journal of Geographical Information Science*, 24(10): 1543–1558, 2010a

Book chapters

3. J.H.M. Janssens, E.O. Postma, and H.J. van den Herik. Density-based anomaly detection in the maritime domain. In P. van de Laar, J. Tretmans, and M. Borth, editors, *Situation Awareness with Systems of Systems*, pages 119–131. Springer Science+Business Media, New York, NY, 2013. ISBN 978-1-4614-6229-3

Conference and workshop proceedings

4. J.H.M. Janssens, H. Hiemstra, and E.O. Postma. PRESTO: A Poseidon research tool to create artificial vessel trajectories. In J.H.M. Janssens and E.O. Postma, editors, *Proceedings of the 1st International Workshop on Maritime Anomaly Detection (MAD 2011)*, pages 43–44, Tilburg, The Netherlands, June 2011a
5. J.H.M. Janssens. Ranking images on semantic attributes using human computation. In *NIPS Workshop on Computational Social Science and the Wisdom of Crowds*, Whistler, Canada, Dec. 2010
6. J.H.M. Janssens, E.O. Postma, and H.J. van den Herik. Unsupervised outlier selection with pairwise affinities. In *SNN Symposium: Intelligent Machines*, Nijmegen, The Netherlands, Nov. 2010b

7. J.H.M. Janssens and E.O. Postma. Ranking images on semantic attributes using human computation. In *SNN Symposium: Intelligent Machines*, Nijmegen, The Netherlands, Nov. 2010
8. J.H.M. Janssens, H. Hiemstra, and E.O. Postma. Creating artificial vessel trajectories with Presto. In *Proceedings of the 22nd Benelux Conference on Artificial Intelligence*, Luxembourg, Luxembourg, Oct. 2010a
9. C.M.E. Willems, W.R. van Hage, G.K.D. de Vries, J.H.M. Janssens, and V. Malaisé. An integrated approach for visual analysis of a multi-source moving objects knowledge base. In G. Andrienko, N. Andrienko, J. Dykes, M-J. Kraak, and H. Schumann, editors, *Geospatial Visual Analytics: Focus on Time*, Guimarães, Portugal, May 2010b
10. J.H.M. Janssens, I. Flesch, and E.O. Postma. Outlier detection with one-class classifiers from ML and KDD. In M.A. Wani, M. Kantardzic, V. Palade, L. Kurgan, and Y. Qi, editors, *Proceedings of the 8th International Conference on Machine Learning and Applications*, pages 147–153, Miami, FL, Dec. 2009
11. J.H.M. Janssens and E.O. Postma. One-class classification with LOF and LOCI: An empirical comparison. In M.G.J. van Erp, J.H. Stehouwer, and M.M. van Zaanen, editors, *Proceedings of the 18th Annual Belgian-Dutch Conference on Machine Learning*, pages 56–64, Tilburg, The Netherlands, June 2009

Edited volumes

12. J.H.M. Janssens, E.O. Postma, and J.W. Hellemons. *Proceedings of the first international workshop on Maritime Anomaly Detection (MAD 2011)*. Tilburg, The Netherlands, 2011b. URL http://mad.uvt.nl/program

Popular scientific

13. J.H.M. Janssens. Outlier detection. *De Connectie*, 2009

Acknowledgements

In the past five years, I have been supported by many people. Without those people I would not have been able to complete this thesis. Now that we have arrived at the end, I would like to take the opportunity to express my gratitude to them.

First and foremost I wish to express my very great appreciation to my supervisors prof.dr. Eric Postma and prof.dr. Jaap van den Herik. Their never-ending support and belief in me has been a crucial factor for the successful completion of the thesis. It has been said many times before, but their qualities are truly complementary to each other.

Eric's excellent guidance and creativity have helped me defining the area of my research. He always listened patiently to my ideas, no matter how crazy they sometimes were. At that moment, Eric was a master of turning a crazy idea into a concrete project. I enjoy recalling the many Poseidon adventures we, together with Ildiko Flesch, have embarked upon. Over the years we developed a common sense of maritime humour that was incomprehensible to outliers. His infinitely positive attitude and energy have cheered me up on more than one occasion.

Jaap has thought me many things, including how to plan, write, and listen. One specific thing that I have learned is that everything should come in three—not in two or four. However, when it comes down to enumerating Jaap's qualities I must make an exception: Jaap is the most hard-working, disciplined, precise, and quick-witted person that I know. He is an expert at detecting textual and grammatical anomalies. I am very grateful to have been supervised by both Eric and Jaap and it has been a pleasure working with them.

Behind every man, from student to professor, there stands a great woman. Two, to be precise in case for Eric and Jaap. Of course, I am referring to Joke Hellemons and Eva Verschoor. Joke and Eva, your help has been essential for me to complete the thesis while I was living in New York. I believe that we all lost the count of the numerous pages that have been typed, emailed, printed, corrected, scanned, and sent back. Thank you for making this process as smooth as possible. I thank Leen Jacobs for his support for all financial matters.

Moreover, I would like to thank the other members of my thesis committee for their preparedness to read the thesis and to assess it to the best of their knowledge:

Prof. dr.Emiel Krahmer, Prof. dr. Hai Xiang Lin, Prof. dr. John-Jules Meyer, Prof. dr. Aske Plaat, and Dr. David Aha.

I thank Ferenc Huszár and Laurens van der Maaten for their contributions to the description of the Stochastic Outlier Selection algorithm in Chapter 4. I am grateful to David Tax for creating the open source MATLAB® Data Description Toolbox and for our many discussions on one-class classification and meta-learning, which formed the basis of Chapters 5 and 6.

It has been a pleasure to have Giel van Lankveld as my roommate, first at Maastricht University (together with Ruud van Stiphout), and later at Tilburg University. Our discussions about life, the University, and everything have been most enjoyable. Furthermore, I would like to thank Giel and Mattias Kik for agreeing to be my paranymphs during my thesis defence.

Then I wish to thank my colleagues at TiCC not already mentioned above for all the interesting conversations, presentations, and social activities: Afra Alishai, Alain Hong, Antal van den Bosch, Bart Joosten, Carel van Wijk, Doug Mastin, Fons Maes, Gerard van Oortmerssen, Hans Westerbeek, Harm Buisman, Harry Bunt, Herman Stehouwer, Igor Berezhnoy, Ildiko Flesch, Jacintha Buysse, Jan van Zanten, Kalliopi Zervanou, Ko van der Sloot, Laurens van der Maaten, Lisanne van Weelden, Lisette Mol, Maarten van Gompel, Marc Swerts, Maria Mos, Marieke Hoetjes, Marieke van Erp, Marie Postma, Martijn Balster, Martijn Goudbeek, Martin Reynaert, Matje van de Camp, Menno van Zaanen, Paul Vogt, Per van der Wijst, Peter Berck, Pieter Spronck, Rein Cozijn, Ruud Koolen, Ruud Mattheij, Ruud van Stiphout, Sander Bakkes, Sander Wubben, Seza Dogruöz, and Suleman Shahid.

In 2010 I was given the privilege to spend four months at the Computational and Biological Learning Lab at the University of Cambridge. I would like to offer my special thanks to prof.dr. Zoubin Ghahramani and prof.dr. Daniel Wolpert for their generous hospitality. Although the many interesting things I learned (especially concerning Bayesian statistics) did not find a place in my thesis, they will surely continue to play an important role in my further machine learning career. I wish to extend my thanks to the following people, who have kindly given me a glimpse of the mesmerising bubble that is Cambridge: Andrew Wilson, Carl Rasmussen, David Franklin, David Knowles, Diane Unwin, Edward Turnham, Ferenc Huszár, Hugo Vincent, Ian Howard, Jurgen van Gael, Máté Lengyel, Peter Orbanz, Ryan Turner, Sae Franklin, Shamir Mohammed, Sinead Williamson, Yue Wu, and Yunus Saatçi.

Because my research was part of the Poseidon project, I had to opportunity to work with many interesting people from various other Dutch universities, Thales Nederland BV in Hengelo, and Noldus Information Technology in Wageningen. I thank the Embedded Systems Institute for managing the project, and Agentschap NL for sponsoring the project under the BSIK 03021 program. The Poseidon project showed me that there is more than just (my own) scientific research. Collaborating with industry, diving into the maritime domain and intellectual property, are just a few examples of the many things I have learned during the Poseidon project. In particular, I wish to acknowledge Alberto González Sánchez, Arjan Mooij, Bob Eskes, Daniel Trivellato, Davide Ceolin, Eric Piel, Fabrizio Maggi, Frans Beenker, Frans Reckers, Fred Spiessens, Gerben de Vries, Hans-Gerhard Groß, Hans Hiemstra, Huub de Waard, Jacek Skowronek, Jack van Wijk, Jan Tretmans, Jimmy Troost, Joris van den Aker, Lucas Noldus, Maarten van Someren, Maurice Glandrup, Michael Borth, Niels Willems, Piërre van de Laar, Sandro Etalle, Veronique Malaisé, Wil van Dommelen, and Willem Robert van Hage.

Since June 2012 I am working as a Data Scientist at Visual Revenue in New York City. (In March 2013, Visual Revenue got acquired by Outbrain, where it continues to be a business unit.) Here, I am applying the research skills and machine-learning knowledge I have gained during my Ph.D. study in a very exciting environment. I am grateful to Dennis Mortensen, Alex Poon, and Charlie Holbech for giving me this opportunity while I was still completing my thesis.

Finally, I wish to thank the people that are closest to me. I very much appreciate the occasional distraction, laughter, and perspective that my close friends provided. I am greatly indebted to my parents Manuel and Rina, and to Michiel, Yasmin, Tom, Anneke, Karen, and Jurrian for their love, support, and encouragement. Last, but certainly not least, I would like to express special thanks to my wife Esther. Her support, patience, and love have been paramount in overcoming the various anomalies encountered during this adventure.

Jeroen Janssens January 2013, New York, NY

SIKS dissertation series

2009

1 Rasa Jurgelenaite (RUN) *Symmetric Causal Independence Models*

2 Willem Robert van Hage (VU) *Evaluating Ontology-Alignment Techniques*

3 Hans Stol (UvT) *A Framework for Evidence-Based Policy Making Using IT*

4 Josephine Nabukenya (RUN) *Improving the Quality of Organisational Policy Making Using Collaboration Engineering*

5 Sietse Overbeek (RUN) *Bridging Supply and Demand for Knowledge Intensive Tasks – Based on Knowledge, Cognition, and Quality*

6 Muhammad Subianto (UU) *Understanding Classification*

7 Ronald Poppe (UT) *Discriminative Vision-Based Recovery and Recognition of Human Motion*

8 Volker Nannen (VU) *Evolutionary Agent-Based Policy Analysis in Dynamic Environments*

9 Benjamin Kanagwa (RUN) *Design, Discovery and Construction of Service-Oriented Systems*

10 Jan Wielemaker (UvA) *Logic Programming for Knowledge-Intensive Interactive Applications*

11 Alexander Boer (UvA) *Legal Theory, Sources of Law & the Semantic Web*

12 Peter Massuthe (TU/e, Humboldt-Universität zu Berlin) *Operating Guidelines for Services*

13 Steven de Jong (UM) *Fairness in Multi-Agent Systems*

14 Maksym Korotkiy (VU) *From Ontology-Enabled Services to Service-Enabled Ontologies (Making Ontologies Work in E-Science With ONTO-SOA)*

15 Rinke Hoekstra (UvA) *Ontology Representation – Design Patterns and Ontologies That Make Sense*

16 Fritz Reul (UvT) *New Architectures in Computer Chess*

17 Laurens van der Maaten (UvT) *Feature Extraction From Visual Data*

18 Fabian Groffen (CWI) *Armada, an Evolving Database System*

19 Valentin Robu (CWI) *Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets*

20 Bob van der Vecht (UU) *Adjustable Autonomy – Controling Influences on Decision Making*

21 Stijn Vanderlooy (UM) *Ranking and Reliable Classification*

22 Pavel Serdyukov (UT) *Search for Expertise – Going Beyond Direct Evidence*

23 Peter Hofgesang (VU) *Modelling Web Usage in a Changing Environment*

24 Annerieke Heuvelink (VU) *Cognitive Models for Training Simulations*

25 Alex van Ballegooij (CWI) *"RAM – Array Database Management Through Relational Mapping"*

26 Fernando Koch (UU) *An Agent-Based Model for the Development of Intelligent Mobile Services*

27 Christian Glahn (OU) *Contextual Support of Social Engagement and Reflection on the Web*

28 Sander Evers (UT) *Sensor Data Management With Probabilistic Models*

Abbreviations. SIKS – Dutch Research School for Information and Knowledge Systems; CWI – Centrum voor Wiskunde en Informatica, Amsterdam; DROP – Delft Research institute for Operations Programming; EUR – Erasmus Universiteit, Rotterdam; KUB – Katholieke Universiteit Brabant, Tilburg; KUN – Katholieke Universiteit Nijmegen; OU – Open Universiteit Nederland; RUG – Rijksuniversiteit Groningen; RUL – Rijksuniversiteit Leiden; RUN – Radboud Universiteit Nijmegen; TiU – Tilburg University; TUD – Technische Universiteit Delft; TU/e – Technische Universiteit Eindhoven; UL – Universiteit Leiden; UM – Universiteit Maastricht; UT – Universiteit Twente; UU – Universiteit Utrecht; UvA – Universiteit van Amsterdam; UvT – Universiteit van Tilburg; VU – Vrije Universiteit, Amsterdam.

29 Stanislav Pokraev (UT) *Model-Driven Semantic Integration of Service-Oriented Applications*

30 Marcin Zukowski (CWI) *Balancing Vectorized Query Execution With Bandwidth-Optimized Storage*

31 Sofiya Katrenko (UvA) *A Closer Look at Learning Relations From Text*

32 Rik Farenhorst and Remco de Boer (VU) *Architectural Knowledge Management – Supporting Architects and Auditors*

33 Khiet Truong (UT) *How Does Real Affect Affect Affect Recognition in Speech?*

34 Inge van de Weerd (UU) *Advancing in Software Product Management – An Incremental Method Engineering Approach*

35 Wouter Koelewijn (UL) *Privacy en Politiegegevens – Over Geautomatiseerde Normatieve Informatie-Uitwisseling*

36 Marco Kalz (OU) *Placement Support for Learners in Learning Networks*

37 Hendrik Drachsler (OU) *Navigation Support for Learners in Informal Learning Networks*

38 Riina Vuorikari (OU) *Tags and Self-Organisation – A Metadata Ecology for Learning Resources in a Multilingual Context*

39 Christian Stahl (TU/e, Humboldt-Universität zu Berlin) *Service Substitution – A Behavioral Approach Based on Petri Nets*

40 Stephan Raaijmakers (UvT) *Multinomial Language Learning – Investigations Into the Geometry of Language*

41 Igor Berezhnyy (UvT) *Digital Analysis of Paintings*

42 Toine Bogers (UvT) *Recommender Systems for Social Bookmarking*

43 Virginia Nunes Leal Franqueira (UT) *Finding Multi-Step Attacks in Computer Networks Using Heuristic Search and Mobile Ambients*

44 Roberto Santana Tapia (UT) *Assessing Business-IT Alignment in Networked Organizations*

45 Jilles Vreeken (UU) *Making Pattern Mining Useful*

46 Loredana Afanasiev (UvA) *Querying XML – Benchmarks and Recursion*

2010

1 Matthijs van Leeuwen (UU) *Patterns That Matter*

2 Ingo Wassink (UT) *Work Flows in Life Science*

3 Joost Geurts (CWI) *A Document Engineering Model and Processing Framework for Multimedia Documents*

4 Olga Kulyk (UT) *Do You Know What I Know? Situational Awareness of Co-Located Teams in Multidisplay Environments*

5 Claudia Hauff (UT) *Predicting the Effectiveness of Queries and Retrieval Systems*

6 Sander Bakkes (TiU) *Rapid Adaptation of Video Game AI*

7 Wim Fikkert (UT) *Gesture Interaction at a Distance*

8 Krzysztof Siewicz (UL) *Towards an Improved Regulatory Framework of Free Software – Protecting User Freedoms in a World of Software Communities and eGovernments*

9 Hugo Kielman (UL) *A Politiele Gegevensverwerking en Privacy, naar een Effectieve Waarborging*

10 Rebecca Ong (UL) *Mobile Communication and Protection of Children*

11 Adriaan Ter Mors (TUD) *The World According to MARP – Multi-Agent Route Planning*

12 Susan van den Braak (UU) *Sensemaking Software for Crime Analysis*

13 Gianluigi Folino (RUN) *High Performance Data Mining Using Bio-Inspired Techniques*

14 Sander van Splunter (VU) *Automated Web Service Reconfiguration*

15 Lianne Bodenstaff (UT) *Managing Dependency Relations in Inter-Organizational Models*

16 Sicco Verwer (TUD) *Efficient Identification of Timed Automata, Theory and Practice*

17 Spyros Kotoulas (VU) *Scalable Discovery of Networked Resources – Algorithms, Infrastructure, Applications*

18 Charlotte Gerritsen (VU) *Caught in the Act – Investigating Crime by Agent-Based Simulation*

19 Henriette Cramer (UvA) *People's Responses to Autonomous and Adaptive Systems*

20 Ivo Swartjes (UT) *Whose Story Is It Anyway? How Improv Informs Agency and Authorship of Emergent Narrative*

21 Harold van Heerde (UT) *Privacy-Aware Data Management by Means of Data Degradation*

22 Michiel Hildebrand (CWI) *End-User Support for Access to Heterogeneous Linked Data*

23 Bas Steunebrink (UU) *The Logical Structure of Emotions*

24 Dmytro Tykhonov *Designing Generic and Efficient Negotiation Strategies*

25 Zulfiqar Ali Memon (VU) *Modelling Human-Awareness for Ambient Agents – A Human Mindreading Perspective*

26 Ying Zhang (CWI) *XRPC – Efficient Distributed Query Processing on Heterogeneous XQuery Engines*

27 Marten Voulon (UL) *Automatisch Contracteren*

28 Arne Koopman (UU) *Characteristic Relational Patterns*

29 Stratos Idreos (CWI) *Database Cracking – Towards Auto-Tuning Database Kernels*

30 Marieke van Erp (TiU) *Accessing Natural History – Discoveries in Data Cleaning, Structuring, and Retrieval*

31 Victor de Boer (UvA) *Ontology Enrichment From Heterogeneous Sources on the Web*

32 Marcel Hiel (TiU) *An Adaptive Service Oriented Architecture – Automatically Solving Interoperability Problems*

33 Robin Aly (UT) *Modeling Representation Uncertainty in Concept-Based Multimedia Retrieval*

34 Teduh Dirgahayu (UT) *Interaction Design in Service Compositions*

35 Dolf Trieschnigg (UT) *Proof of Concept – Concept-Based Biomedical Information Retrieval*

36 Jose Janssen (OU) *Paving the Way for Lifelong Learning – Facilitating Competence Development Through a Learning Path Specification*

37 Niels Lohmann (TU/e) *Correctness of Services and Their Composition*

38 Dirk Fahland (TU/e) *From Scenarios to Components*

39 Ghazanfar Farooq Siddiqui (VU) *Integrative Modeling of Emotions in Virtual Agents*

40 Mark van Assem (VU) *Converting and Integrating Vocabularies for the Semantic Web*

41 Guillaume Chaslot (UM) *Monte-Carlo Tree Search*

42 Sybren de Kinderen (VU) *Needs-Driven Service Bundling in a Multi-Supplier Setting – The Computational E3-Service Approach*

43 Peter van Kranenburg (UU) *A Computational Approach to Content-Based Retrieval of Folk Song Melodies*

44 Pieter Bellekens (TU/e) *An Approach Towards Context-Sensitive and User-Adapted Access to Heterogeneous Data Sources, Illustrated in the Television Domain*

45 Vasilios Andrikopoulos (TiU) *A Theory and Model for the Evolution of Software Services*

46 Vincent Pijpers (VU) *E3alignment – Exploring Inter-Organizational Business-ICT Alignment*

47 Chen Li (UT) *Mining Process Model Variants – Challenges, Techniques, Examples*

48 Withdrawn

49 Jahn-Takeshi Saito (UM) *Solving Difficult Game Positions*

50 Bouke Huurnink (UvA) *Search in Audiovisual Broadcast Archives*

51 Alia Khairia Amin (CWI) *Understanding and Supporting Information Seeking Tasks in Multiple Sources*

52 Peter-Paul van Maanen (VU) *Adaptive Support for Human-Computer Teams – Exploring the Use of Cognitive Models of Trust and Attention*

53 Edgar Meij (UvA) *Combining Concepts and Language Models for Information Access*

2011

1 Botond Cseke (RUN) *Variational Algorithms for Bayesian Inference in Latent Gaussian Models*

2 Nick Tinnemeier (UU) *Organizing Agent Organizations – Syntax and Operational Semantics of an Organization-Oriented Programming Language*

3 Jan Martijn van der Werf (TU/e) *Compositional Design and Verification of Component-Based Information Systems*

4 Hado van Hasselt (UU) *Insights in Reinforcement Learning – Formal Analysis and Empirical Evaluation of Temporal-Difference*

5 Base van der Raadt (VU) *Enterprise Architecture Coming of Age – Increasing the Performance of an Emerging Discipline*

6 Yiwen Wang (TU/e) *Semantically-Enhanced Recommendations in Cultural Heritage*

7 Yujia Cao (UT) *Multimodal Information Presentation for High Load Human Computer Interaction*

8 Nieske Vergunst (UU) *BDI-based Generation of Robust Task-Oriented Dialogues*

9 Tim de Jong (OU) *Contextualised Mobile Media for Learning*

10 Bart Bogaert (TiU) *Cloud Content Contention*

11 Dhaval Vyas (UT) *Designing for Awareness – An Experience-Focused HCI Perspective*

12 Carmen Bratosin (TU/e) *Grid Architecture for Distributed Process Mining*

13 Xiaoyu Mao (TiU) *Airport Under Control – Multiagent Scheduling for Airport Ground Handling*

14 Milan Lovric (EUR) *Behavioral Finance and Agent-Based Artificial Markets*

15 Marijn Koolen (UvA) *The Meaning of Structure – The Value of Link Evidence for Information Retrieval*

16 Maarten Schadd (UM) *Selective Search in Games of Different Complexity*

17 Jiyin He (UvA) *Exploring Topic Structure – Coherence, Diversity and Relatedness*

18 Mark Ponsen (UM) *Strategic Decision-Making in Complex Games*

19 Ellen Rusman (OU) *The Mind's Eye on Personal Profiles*

20 Qing Gu (VU) *Guiding Service-Oriented Software Engineering – A View-Based Approach*

21 Linda Terlouw (TUD) *Modularization and Specification of Service-Oriented Systems*

22 Junte Zhang (UvA) *System Evaluation of Archival Description and Access*

23 Wouter Weerkamp (UvA) *Finding People and Their Utterances in Social Media*

24 Herwin van Welbergen (UT) *Behavior Generation for Interpersonal Coordination With Virtual Humans on Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior*

25 Syed Waqar ul Qounain Jaffry (VU) *Analysis and Validation of Models for Trust Dynamics*

26 Matthijs Aart Pontier (VU) *Virtual Agents for Human Communication – Emotion Regulation and Involvement-Distance Trade-Offs in Embodied Conversational Agents and Robots*

27 Aniel Bhulai (VU) *Dynamic Website Optimization Through Autonomous Management of Design Patterns*

28 Rianne Kaptein (UvA) *Effective Focused Retrieval by Exploiting Query Context and Document Structure*

29 Faisal Kamiran (TU/e) *Discrimination-Aware Classification*

30 Egon van den Broek (UT) *Affective Signal Processing (ASP) – Unraveling the Mystery of Emotions*

31 Ludo Waltman (EUR) *Computational and Game-Theoretic Approaches for Modeling Bounded Rationality*

32 Nees-Jan van Eck (EUR) *Methodological Advances in Bibliometric Mapping of Science*

33 Tom van der Weide (UU) *Arguing to Motivate Decisions*

34 Paolo Turrini (UU) *Strategic Reasoning in Interdependence – Logical and Game-Theoretical Investigations*

35 Maaike Harbers (UU) *Explaining Agent Behavior in Virtual Training*

36 Erik van der Spek (UU) *Experiments in Serious Game Design – A Cognitive Approach*

37 Adriana Burlutiu (RUN) *MacHine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference*

38 Nyree Lemmens (UM) *Bee-Inspired Distributed Optimization*

39 Joost Westra (UU) *Organizing Adaptation Using Agents in Serious Games*

40 Viktor Clerc (VU) *Architectural Knowledge Management in Global Software Development*

41 Luan Ibraimi (UT) *Cryptographically Enforced Distributed Data Access Control*

42 Michal Sindlar (UU) *Explaining Behavior Through Mental State Attribution*

43 Henk van der Schuur (UU) *Process Improvement Through Software Operation Knowledge*

44 Boris Reuderink (UT) *Robust Brain-Computer Interfaces*

45 Herman Stehouwer (TiU) *Statistical Language Models for Alternative Sequence Selection*

46 Beibei Hu (TUD) *Towards Contextualized Information Delivery – A Rule-Based Architecture for the Domain of Mobile Police Work*

47 Azizi Bin Ab Aziz (VU) *Exploring Computational Models for Intelligent Support of Persons With Depression*

48 Mark ter Maat (UT) *Response Selection and Turn-Taking for a Sensitive Artificial Listening Agent*

49 Andreea Niculescu (UT) *Conversational Interfaces for Task-Oriented Spoken Dialogues – Design Aspects Influencing Interaction Quality*

2012

1 Terry Kakeeto (TiU) *Relationship Marketing for SMEs in Uganda*

2 Muhammad Umair (VU) *Adaptivity, Emotion, and Rationality in Human and Ambient Agent Models*

3 Adam Vanya (VU) *Supporting Architecture Evolution by Mining Software Repositories*

4 Jurriaan Souer (UU) *Development of Content Management System-Based Web Applications*

5 Marijn Plomp (UU) *Maturing Interorganisational Information Systems*

6 Wolfgang Reinhardt (OU) *Awareness Support for Knowledge Workers in Research Networks*

7 Rianne van Lambalgen (VU) *When the Going Gets Tough – Exploring Agent-Based Models of Human Performance Under Demanding Conditions*

8 Gerben de Vries (UvA) *Kernel Methods for Vessel Trajectories*

9 Ricardo Neisse (UT) *Trust and Privacy Management Support for Context-Aware Service Platforms*

10 David Smits (TU/e) *Towards a Generic Distributed Adaptive Hypermedia Environment*

11 J.C.B. Rantham Prabhakara (TU/e) *Process Mining in the Large – Preprocessing, Discovery, and Diagnostics*

12 Kees van der Sluijs (TU/e) *Model Driven Design and Data Integration in Semantic Web Information Systems*

13 Suleman Shahid (TiU) *Fun and Face – Exploring Non-Verbal Expressions of Emotion During Playful Interactions*

14 Evgeny Knutov (TU/e) *Generic Adaptation Framework for Unifying Adaptive Web-Based Systems*

15 Natalie van der Wal (VU) *Social Agents – Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes*

16 Fiemke Both (VU) *Helping People by Understanding Them – Ambient Agents Supporting Task Execution and Depression Treatment*

17 Amal Elgammal (TiU) *Towards a Comprehensive Framework for Business Process Compliance*

18 Eltjo Poort (VU) *Improving Solution Architecting Practices*

19 Helen Schonenberg (TU/e) *What's Next? Operational Support for Business Process Execution*

20 Ali Bahramisharif (RUN) *Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing*

21 Roberto Cornacchia (TUD) *Querying Sparse Matrices for Information Retrieval*

22 Thijs Vis (TiU) *Intelligence, Politie en Veiligheidsdienst – Verenigbare Grootheden?*

23 Christian Muehl (UT) *Toward Affective Brain-Computer Interfaces – Exploring the Neurophysiology of Affect During Human Media Interaction*

24 Laurens van der Werff (UT) *Evaluation of Noisy Transcripts for Spoken Document Retrieval*

25 Silja Eckartz (UT) *Managing the Business Case Development in Inter-Organizational IT Projects – A Methodology and Its Application*

26 Emile de Maat (UvA) *Making Sense of Legal Text*

27 Hayrettin Gurkok (UT) *Mind the Sheep! User Experience Evaluation & Brain-Computer Interface Games*

28 Nancy Pascall (TiU) *Engendering Technology Empowering Women*

29 Almer Tigelaar (UT) *Peer-To-Peer Information Retrieval*

30 Alina Pommeranz (TUD) *Designing Human-Centered Systems for Reflective Decision Making*

31 Emily Bagarukayo (RUN) *A Learning by Construction Approach for Higher Order Cognitive Skills Improvement, Building Capacity and Infrastructure*

32 Wietske Visser (TUD) *Qualitative Multi-Criteria Preference Representation and Reasoning*

33 Rory Sie (OU) *Coalitions in Cooperation Networks (COCOON)*

34 Pavol Jancura (RUN) *Evolutionary Analysis in PPI Networks and Applications*

35 Evert Haasdijk (VU) *Never Too Old to Learn – On-Line Evolution of Controllers in Swarm- and Modular Robotics*

36 Denis Ssebugwawo (RUN) *Analysis and Evaluation of Collaborative Modeling Processes*

37 Agnes Nakakawa (RUN) *A Collaboration Process for Enterprise Architecture Creation*

38 Selmar Smit (VU) *Parameter Tuning and Scientific Testing in Evolutionary Algorithms*

39 Hassan Fatemi (UT) *Risk-Aware Design of Value and Coordination Networks*

40 Agus Gunawan (TiU) *Information Access for SMEs in Indonesia*

41 Sebastian Kelle (OU) *Game Design Patterns for Learning*

42 Dominique Verpoorten (OU) *Reflection Amplifiers in Self-Regulated Learning*

43 Withdrawn

44 Anna Tordai (VU) *On Combining Alignment Techniques*

45 Benedikt Kratz (TiU) *A Model and Language for Business-Aware Transactions*

46 Simon Carter (UvA) *Exploration and Exploitation of Multilingual Data for Statistical MacHine Translation*

47 Manos Tsagkias (UvA) *Mining Social Media – Tracking Content and Predicting Behavior*

48 Jorn Bakker (TU/e) *Handling Abrupt Changes in Evolving Time-Series Data*

49 Michael Kaisers (UM) *Learning Against Learning – Evolutionary Dynamics of Reinforcement Learning Algorithms in Strategic Interactions*

50 Steven van Kervel (TUD) *Ontologogy Driven Enterprise Information Systems Engineering*

51 Jeroen de Jong (TUD) *Heuristics in Dynamic Sceduling – A Practical Framework With a Case Study in Elevator Dispatching*

2013

1 Viorel Milea (EUR) *News Analytics for Financial Decision Support*

2 Erietta Liarou (CWI) *MonetDB/DataCell – Leveraging the Column-Store Database Technology for Efficient and Scalable Stream Processing*

3 Szymon Klarman (VU) *Reasoning With Contexts in Description Logics*

4 Chetan Yadati (TUD) *Coordinating Autonomous Planning and Scheduling*

5 Dulce Pumareja (UT) *Groupware Requirements Evolutions Patterns*

6 Romulo Goncalves (CWI) *The Data Cyclotron – Juggling Data and Queries for a Data Warehouse Audience*

7 Giel van Lankveld (UT) *Quantifying Individual Player Differences*

8 Robbert-Jan Merk (VU) *Making Enemies – Cognitive Modeling for Opponent Agents in Fighter Pilot Simulators*

9 Fabio Gori (RUN) *Metagenomic Data Analysis – Computational Methods and Applications*

10 Jeewanie Jayasinghe Arachchige (TiU) *A Unified Modeling Framework for Service Design*

11 Evangelos Pournaras (TUD) *Multi-Level Reconfigurable Self-organization in Overlay Services*

12 Maryam Razavian (VU) *Knowledge-driven Migration to Services*

13 Mohammad Zafiri (UT) *Service Tailoring – User-Centric Creation of Integrated IT-Based Homecare Services to Support Independent Living of Elderly*

14 Jafar Tanha (UVA) *Ensemble Approaches to Semi-Supervised Learning Learning*

15 Daniel Hennes (UM) *Multiagent Learning – Dynamic Games and Applications*

16 Eric Kok (UU) *Exploring the Practical Benefits of Argumentation in Multi-Agent Deliberation*

17 Koen Kok (VU) *The PowerMatcher: Smart Coordination for the Smart Electricity Grid*

18 Jeroen Janssens (TiU) *Outlier Selection and One-Class Classification*

TiCC dissertation series

1 Pashiera Barkhuysen. *Audiovisual Prosody in Interaction*. Promotores: M.G.J. Swerts, E.J. Krahmer. Tilburg, 3 October 2008.

2 Ben Torben-Nielsen. *Dendritic Morphology: Function Shapes Structure*. Promotores: H.J. van den Herik, E.O. Postma. Co-promotor: K.P. Tuyls. Tilburg, 3 December 2008.

3 Hans Stol. *A Framework for Evidence-based Policy Making Using IT*. Promotor: H.J. van den Herik. Tilburg, 21 January 2009.

4 Jeroen Geertzen. *Dialogue Act Recognition and Prediction*. Promotor: H. Bunt. Co-promotor: J.M.B. Terken. Tilburg, 11 February 2009.

5 Sander Canisius. *Structured Prediction for Natural Language Processing*. Promotores: A.P.J. van den Bosch, W. Daelemans. Tilburg, 13 February 2009.

6 Fritz Reul. *New Architectures in Computer Chess*. Promotor: H.J. van den Herik. Co-promotor: J.W.H.M. Uiterwijk. Tilburg, 17 June 2009.

7 Laurens van der Maaten. *Feature Extraction from Visual Data*. Promotores: E.O. Postma, H.J. van den Herik. Co-promotor: A.G. Lange. Tilburg, 23 June 2009.

8 Stephan Raaijmakers. *Multinomial Language Learning*. Promotores: W. Daelemans, A.P.J. van den Bosch. Tilburg, 1 December 2009.

9 Igor Berezhnoy. *Digital Analysis of Paintings*. Promotores: E.O. Postma, H.J. van den Herik. Tilburg, 7 December 2009.

10 Toine Bogers. *Recommender Systems for Social Bookmarking*. Promotor: A.P.J. van den Bosch. Tilburg, 8 December 2009.

11 Sander Bakkes. *Rapid Adaptation of Video Game AI*. Promotor: H.J. van den Herik. Co-promotor: P. Spronck. Tilburg, 3 March 2010.

12 Maria Mos. *Complex Lexical Items*. Promotor: A.P.J. van den Bosch. Co-promotores: A. Vermeer, A. Backus. Tilburg, 12 May 2010 (in collaboration with the Department of Language and Culture Studies).

13 Marieke van Erp. *Accessing Natural History – Discoveries in data cleaning, structuring, and retrieval*. Promotor: A.P.J. van den Bosch. Co-promotor: P.K. Lendvai. Tilburg, 30 June 2010.

14 Edwin Commandeur. *Implicit Causality and Implicit Consequentiality in Language Comprehension*. Promotores: L.G.M. Noordman, W. Vonk. Co-promotor: R. Cozijn. Tilburg, 30 June 2010.

15 Bart Bogaert. *Cloud Content Contention*. Promotores: H.J. van den Herik, E.O. Postma. Tilburg, 30 March 2011.

16 Xiaoyu Mao. *Airport under Control*. Promotores: H.J. van den Herik, E.O. Postma. Co-promotores: N. Roos, A. Salden. Tilburg, 25 May 2011.

17 Olga Petukhova. *Multidimensional Dialogue Modelling*. Promotor: H. Bunt. Tilburg, 1 September 2011.

18 Lisette Mol. *Language in the Hands*. Promotores: E.J. Krahmer, F. Maes, M.G.J. Swerts. Tilburg, 7 November 2011.

19 Herman Stehouwer. *Statistical Language Models for Alternative Sequence Selection*. Promotores: A.P.J. van den Bosch, H.J.

van den Herik. Co-promotor: M.M. van Zaanen. Tilburg, 7 December 2011.

20 Terry Kakeeto-Aelen. *Relationship Marketing for SMEs in Uganda*. Promotores: J. Chr. van Dalen, H.J. van den Herik. Co-promotor: B.A. Van de Walle. Tilburg, 1 February 2012.

21 Suleman Shahid. *Fun & Face: Exploring non-verbal expressions of emotion during playful interactions*. Promotores: E.J. Krahmer, M.G.J. Swerts. Tilburg, 25 May 2012.

22 Thijs Vis. *Intelligence, Politie en Veiligheidsdienst: Verenigbare Grootheden?*. Promotores: T.A. de Roos, H.J. van den Herik, A.C.M. Spapens. Tilburg, 6 June 2012 (in collaboration with the Tilburg School of Law).

23 Nancy Pascall. *Engendering Technology Empowering Women*. Promotores: H.J. van den Herik, M. Diocaretz. Tilburg, 19 November, 2012.

24 Agus Gunawan. *Information Access for SMEs in Indonesia*. Promotor: H.J. van den Herik. Co-promotores: M. Wahdan, B.A. Van de Walle. Tilburg, 19 December 2012.

25 Giel van Lankveld. *Quantifying Individual Player Differences*. Promotores: H.J. van den Herik, A.R. Arntz. Co-promotor: P. Spronck. Tilburg, 27 February 2013.

26 Sander Wubben. *Text-to-text Generation Using Monolingual Machine Translation*. Promotores: E.J. Krahmer, A.P.J. van den Bosch, H. Bunt. Tilburg, 5 June 2013.

27 Jeroen Janssens. *Outlier Selection and One-Class Classification*. Promotores: E.O. Postma, H.J. van den Herik. Tilburg, 11 June 2013.