A Study of Unsupervised Outlier Detection for One-Class Classification

by

Lorne Swersky

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

One-class classification is a problem that arises in situations where we have data available that describes objects belonging to a particular class but very little or no data describing objects that do not belong to this class, where we must then be able to classify new data objects according to whether or not they belong to this class. Outlier detection is a similar problem where we are presented with an unlabelled collection of data and must determine whether the data objects are outliers or inliers according to some definition of an outlier. In this thesis we explore the relationship between one-class classification and outlier detection by comparing methods used for each problem in a common framework, investigate some unique issues in applying one-class classification in a realistic setting, as well as consider methods to combine one-class classifiers.

In comparing one-class classification and outlier detection, we note that they are similar problems in that both are looking to classify data objects as either inlier or outlier. We extend previous comparison studies by studying a number of one-class classification and unsupervised outlier detection methods in a rigorous experimental setup, comparing them on a large number of datasets with different characteristics using the commonly used area under the receiver operating characteristic curve (AUC) measure, as well as the adjusted precision-at-n measure used in unsupervised outlier detection. An additional contribution is the adaptation of the unsupervised outlier detection method, GLOSH, to the one-class classification setting.

The lack of outlier data objects available for training means that we cannot use the standard procedure of using a validation set in order to estimate the generalization performance of a model for one-class classification, and so selecting good parameters for a method can be difficult. We investigate this problem by comparing the performance of methods at different parameter values to determine how stable their performance is with respect to their parameters, and whether certain parameter settings are likely to do well across multiple datasets.

In combining one-class classifiers, we apply rank-based combination strategies to the outlier rankings produced by multiple one-class classifiers and compare different strategies. We find that simple combinations of ranks can produce robust classifiers which outperform individual classifiers.

Preface

Some of the research conducted for this thesis forms part of a research collaboration led by my supervisor Professor Jörg Sander at the University of Alberta. Chapter 4 was published as part of L. Swersky, H. O. Marques, J. Sander, R. J. Campello, and A. Zimek, "On the Evaluation of Outlier Detection and One-Class Classification Methods," 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Montreal, QC, 2016, pp. 1-10, with sections from Chapters 1, 2, and 3 also appearing in the paper. I was the main contributor in developing the main ideas and the experimental setup for the comparison study performed, as well as the adaptation of GLOSH which additionally led to the methodological framework for adapting unsupervised outlier detection methods to one-class classification, as well as being involved in the writing of the manuscript. H. O. Marques was responsible for conducting the experiments as well as participating in analyzing the results and in the writing of the manuscript. R. J. Campello and J. Sander were involved in developing the ideas as well as analyzing the results and writing the manuscript. A. Zimek was involved in discussing the ideas and editing.

The good news about computers is that they do what you tell them to do. The bad news is that they do what you tell them to do.

– Ted Nelson

Acknowledgements

I would like to thank my supervisor, Jörg Sander, for providing guidance, and especially patience. He always had a suggestion when I had no idea where to look, and without him this thesis would not be possible.

I would also like to thank my additional collaborators: Ricardo Campello, for providing insight and discussion; Henrique Oliveira Marques, for saving me when I proved out of my depth; and Arthur Zimek for his advice.

I also extend my thanks to Davood Rafei for agreeing to be on my examining committee.

Lastly, I would like to thank my family, my brothers: Darren, Nathan, and Kevin, and my sister, Ashley, who have all provided me with support and inspiration, as well as my parents, who I could always rely on for their love.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Outlier detection is one of the central tasks of data mining. The goal of this task is to identify those observations which deviate substantially from the remaining data. Many definitions of outlier exist in the literature. One of the most used is Hawkins' definition [24], which refers to an outlier as "an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism". Detecting such patterns is important because they might represent extraordinary behaviors that deserve some special attention, such as traffic accidents and network intrusion attacks.

Outlier detection algorithms can be categorized in supervised, semi-supervised, and unsupervised techniques [22]. Supervised techniques can be seen as a special case of binary classification, where there are enough observations labeled as inliers and outliers available to train a classifier using approaches for imbalanced classification [1]. In semi-supervised outlier detection, due to the rarity of the outliers, there are only very few or even no outlier observations available to sufficiently describe the outlier class. In this scenario, also referred to as "one-class classification", the model is typically obtained using one-class classification techniques [46]. When no labeled data are available, unsupervised techniques can be used that do not assume any prior knowledge about which observations are outliers and which are inliers [52].

One-class classification in particular represents an important problem in today's world, as our society becomes more and more data-driven. In situations

such as credit card fraud or aircraft engine testing, we have many instances of normal behavior, but few anomalous observations which do not provide a representative sample with which we could train a binary classifier and achieve good results. Instead we turn to one-class classification techniques which only require instances of normal objects in order to make predictions about new data.

1.1 Thesis Outline

The thesis is organized as follows. In Chapter 2 we provide an overview of the one-class classification and unsupervised outlier detection problems, as well as the methods used in each that we compare in this study. We also detail the evaluation measures commonly used.

In Chapter 3 we propose a methodological framework in which unsupervised outlier detection methods can be adapted for one-class classification while establishing basic principles that should not be violated in order to provide a foundation for a model is not changed by new observations, as well as to improve the runtime performance of unsupervised outlier detection methods for one-class classification. Utilizing this framework, we adapt the GLOSH outlier detection method to one-class classification.

In Chapter 4 we perform a thorough comparison study on a number of unsupervised outlier detection methods and one-class classification methods, performing an expanded study similar to those previously done, as well as in a new scenario that has not been previously studied by reversing the selection of classes for inliers and outliers. Additionally we introduce an evaluation measure used in unsupervised outlier detection (Adjusted-Precision-at-n) to one-class classification to complement those commonly used.

In Chapter 5 we investigate the hyperparameter selection problem in one-class classification, performing an analysis of the datasets used and the relative strengths and weaknesses of some of the better performing methods in order to gain some insight which may help guide the selection of hyperparameters in the absence of real outliers for performing validation.

In Chapter 6 we apply the use of rank-based combination strategies to one-class classifiers, showing a fast and simple way to improve robustness and accuracy by combining one-class classifiers.

Finally, in Chapter 7 we conclude this thesis by providing a summary of this study and providing possible future directions for research.

Chapter 2

Background and Related Work

In this chapter we discuss related work that compares one-class classification methods with unsupervised outlier detection methods, as well as provide descriptions of the outlier detection and one-class classification methods used in this thesis, and how they are evaluated.

Although both unsupervised outlier detection and one-class classification were first studied in field of statistics [4], [40], little has been done in the literature in order to compare the performance of both categories of algorithms. The one-class classification scenario generally is an easier task, because training data for one class is available. While in the one-class classification setting one can estimate the probability density function (p.d.f.) or other models without considering the presence of outliers in the dataset, in the unsupervised outlier detection setting one must deal with possible outliers while estimating the p.d.f. or other models. Given the differences between these two settings, the corresponding methods cannot be compared in a straightforward way.

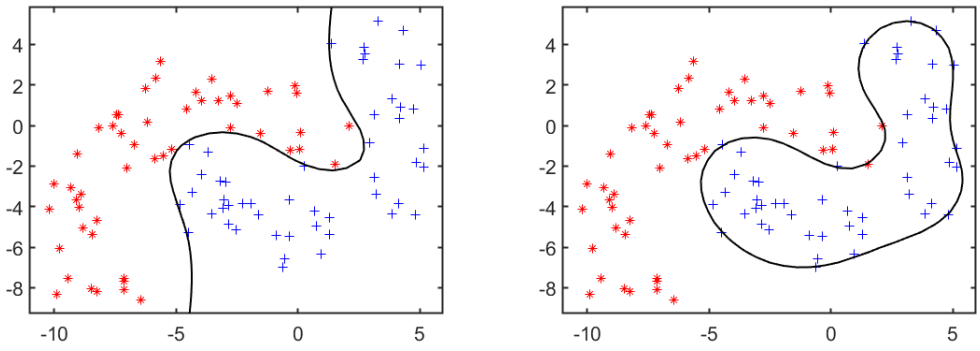
One attempt to compare one-class classification and unsupervised outlier detection methods was done by Hido *et al.* [26]. In that work, the authors compared their proposed outlier detection algorithm against other approaches, including supervised, semi-supervised and unsupervised outlier detection techniques. The comparison, however, was not entirely fair since most compared algorithms had their parameters tuned using cross-validation, while only 3 different values for LOF's (see Section 2.2) parameter were tested.

Janssens *et al.* [29] proposed a methodological framework to make unsu-

pervised outlier detection algorithms work in a one-class classification setup and thus be able to compare them against algorithms specifically designed for this task. The authors, however, only assessed the performance of two unsupervised methods, namely, LOF and LOCI (see Section 2.2), in the one-class classification scenario. In a follow-up work [28], the same framework previously proposed in [29] was once again applied to LOF and LOCI, but this time these methods were also compared against three one-class classification methods. The authors concluded that LOF and SVDD (see Section 2.1) are the top two performers with an identical average performance rank, although each method has particular scenarios where one may outperform the other.

2.1 One-Class Classification

Unlike in the traditional classification problem, in one-class classification [55] we are only provided with observations from one class and our model must then classify new observations as belonging to this class or not. While in binary classification one strategy might be to separate the feature space into two areas representing each class, the analogous strategy for one-class classification would be to define some region of the feature space as belonging to the inlier class, with the rest of the feature space belonging to outliers (Figure 2.1).



(a) Binary (b) One-class

Figure 2.1: Binary and one-class classifiers trained on a banana-shaped dataset.

In order to keep terminology consistent, we will refer to observations be-

longing to the provided class as inliers, and observations not belonging to this class as outliers.

One-class classifiers can be categorized into density methods, boundary methods, and reconstruction methods. Commonly used density methods for one-class classification are the Gaussian density [5], Mixture of Gaussians [5], and Parzen density estimation [44]. In density estimation, the parameters for some p.d.f. can be fit using the training data, and then new observations can be classified using this p.d.f. Since the p.d.f. is estimated using only inliers, there is no risk of outliers affecting the distribution. One drawback, however, is that we require a sufficiently large sample of inliers to produce a good estimation. Depending, *e.g.*, on the dimensionality of the problem, the number of observations required to sufficiently represent the underlying distribution can become very large, and may prove too expensive to obtain in a real-world setting.

Boundary methods avoid the requirement for a large number of samples by instead attempting to define a boundary around the training data, such that new observations that fall within the boundary are classified as inliers, while observations falling outside of the boundary are classified as outliers. Since we are only interested in defining this boundary, it is not necessary to obtain a large number of samples to fully represent the inlier class. Boundary methods include the Support Vector Data Description (SVDD) [54] and the Linear Programming Distance Data Description (LP) [45].

Lastly, reconstruction methods can be used to model the training data by using a generating process. Such a generating process is chosen to provide a compact representation of the data while attempting to preserving most of the information as well as filtering out noise. Once the model has been obtained, new observations can then be described through this model in order to be classified. Reconstruction methods include approaches like auto-encoder networks [30].

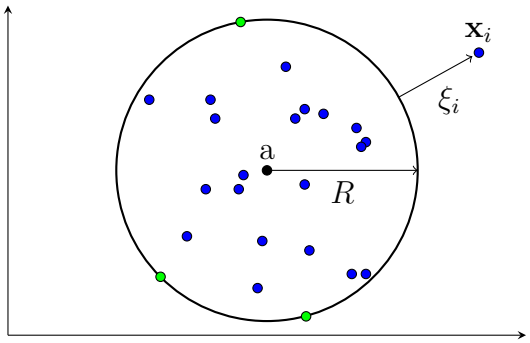


Figure 2.2: An example hypersphere enclosing inlier data with center $\mathbf{a}$ and radius R. The objects on the boundary form support vectors, and the object $\mathbf{x}_i$ is outside the boundary with $\xi_i > 0$ (adapted from [55]).

Support Vector Data Description

Support Vector Data Description (SVDD) [54] is a boundary-based one-class classification method inspired by Support Vector Machines (SVM) [59] used in regular classification problems. The primary difference between SVDD and SVM is that while SVM attempts to separate two or more classes with a maximum margin hyperplane, SVDD instead will enclose the inlier class in a minimum volume hypersphere (see Figure 2.2) by minimizing the following error:

$$\mathcal{E}(R, \mathbf{a}, \boldsymbol{\xi}) = R^2 + C \sum_i \xi_i \tag{2.1}$$

subject to the constraints:

$$\|\mathbf{x}_i - \mathbf{a}\|^2 \leq R^2 + \xi_i, \quad \xi_i \geq 0, \quad \forall i \tag{2.2}$$

where R is the radius of the hypersphere, $\mathbf{a}$ is the center of the hypersphere, $\boldsymbol{\xi}$ are slack variables allowing training observations $\mathbf{x}$ to fall outside the SVDD boundary, and C is a penalty (regularization) parameter. C can be replaced by a parameter *fracrej*, which represents the fraction of training objects that will lie outside the hypersphere boundary, allowing for more intuitive tuning of the regularization parameter.

Like traditional SVMs, the above formulation can also be extended to non-linearly transformed spaces using kernel methods. In our experiments we use

a Gaussian kernel.

Gaussian Data Description

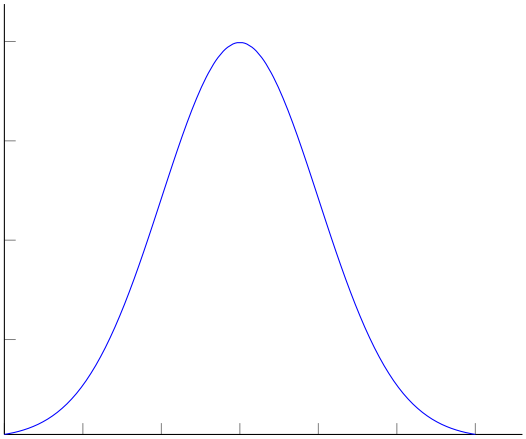


Figure 2.3: A simple Gaussian probability density function.

In the Gaussian Data Description [55], the Gaussian probability density function (Figure 2.3):

$$p_{Gauss}(\mathbf{x}|\boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \mathrm{e}^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})} \tag{2.3}$$

is fit to the inlier data, where $\boldsymbol{\mu}$ is the mean and Σ is the covariance matrix, and d is the dimensionality of the data. A new observation can be classified by computing its probability under the learned distribution. We can also apply a regularization parameter r to the covariance matrix:

$$\Sigma_{new} = (1 - r)\Sigma + rI \tag{2.4}$$

Parzen Window Data Description

Parzen Window Data Description (PW) is based on Parzen Density Estimation [44] which estimates the density of the data using a mixture of kernels centered on each of the N individual training observations (Figure 2.4). In our case, we use Gaussian kernels with diagonal covariance matrices $\Sigma_i = \lambda I$, where λ is a width parameter which can be optimized using the maximum likelihood

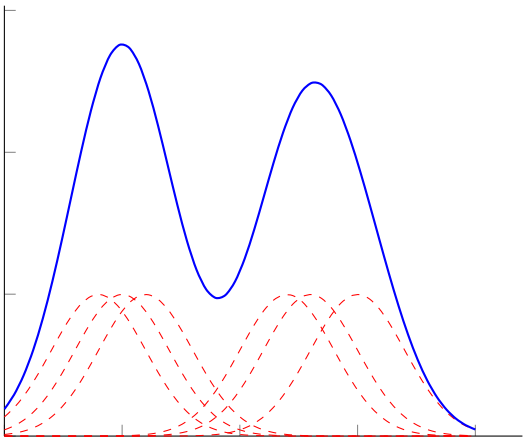


Figure 2.4: An example of a Parzen density estimator.

solution. The probability of an observation being an inlier is then computed as:

$$PW(\mathbf{x}) = \frac{1}{N} \sum_i p_{Gauss}(\mathbf{x}|\mathbf{x}_i, \lambda I) \tag{2.5}$$

Unlike other density methods, PW is non-parametric and since it sums over all data points, classifying new observations can be relatively expensive.

Linear Programming Distance Data Description

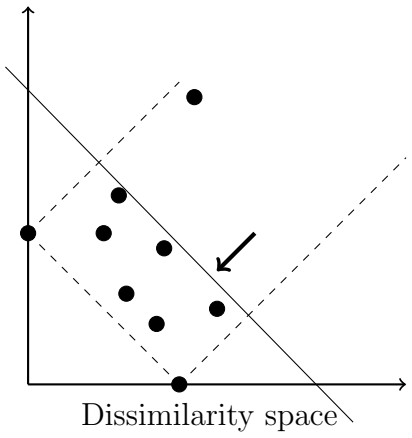


Figure 2.5: The intuition of LP is to attract a hyperplane as close to the origin as possible in the dissimilarity space. The dashed lines indicate the open boundary that objects are contained in (adapted from [45]).

The Linear Programming Distance Data Description (LP) [45] is a bound-

ary method which utilizes a dissimilarity measure to compare new observations to inlier observations in the training set, $\mathcal{D}_{\text{train}}$. The dissimilarity measure used must meet a number of criteria defined by the authors. One such measure they propose and which we use in our experiments is the $L_{0.95}$ dissimilarity. LP constructs a boundary by bringing a hyperplane which bounds the training set from above in the dissimilarity space as close to the origin as possible while still accepting most inliers (Figure 2.5).

Auto-Encoder Data Description

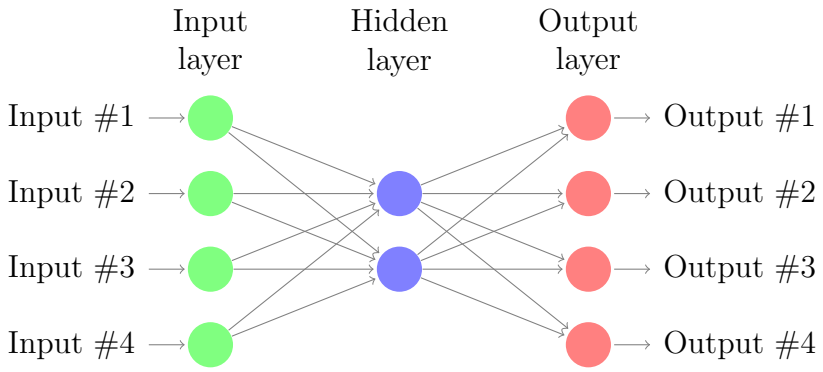


Figure 2.6: An auto-encoder with 2 hidden units.

In the Auto-Encoder Data Description [55], a neural network with hyperbolic tangent sigmoid units, a single hidden layer, and a parameter-defined number of hidden units is trained on the inlier class with the goal of recovering the input data at the output layer (Figure 2.6). In order to classify new observations, each observation to be classified is supplied as input to the network, and the difference between the original input and the network's output in terms of mean squared error is computed.

The Auto-Encoder Data Description falls within the category of reconstruction methods for one-class classification.

k-Nearest Neighbor Data Description

k-Nearest Neighbor Data Description [28], [48], which we call here kNN_{local}, is similar to LOF and LOCI in that it approximates the local density of the training observations, however in a simpler way (see Figure 2.7). An observation

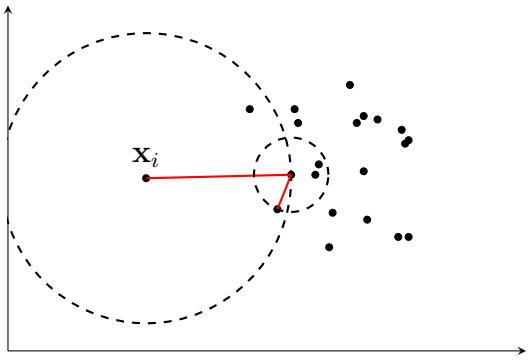


Figure 2.7: k-Nearest Neighbor Data Description with $k = 3$.

is classified under kNN_{local} by computing the ratio between the distance from an observation to its k^{th} nearest neighbor $\text{NN}_{\text{k}}(\mathbf{x}_i)$, and the distance between the k^{th} nearest neighbor and its k^{th} nearest neighbor:

$$\text{kNN}_{\text{local}}(\mathbf{x}_i, k) = \frac{d(\mathbf{x}_i, \text{NN}_{\text{k}}(\mathbf{x}_i))}{d(\text{NN}_{\text{k}}(\mathbf{x}_i), \text{NN}_{\text{k}}(\text{NN}_{\text{k}}(\mathbf{x}_i)))} \tag{2.6}$$

2.2 Unsupervised Outlier Detection

In unsupervised outlier detection, we are provided with a set of unlabeled observations and are tasked with determining whether each observation is an inlier or an outlier. In this situation, an outlier may be defined as an observation that deviates from other observations in some significant way, as determined by the chosen method. There are a variety of approaches, including statistical, density-based, and cluster-based methods.

Statistical outlier detection methods [24] assume that the inliers were generated using some known parametric type of probability density function (p.d.f.). Potential drawbacks of these methods in the unsupervised setting are (1) the data may now contain outliers that will influence the estimated parameters of the assumed density, and (2) we rarely know in advance the true distribution of the data.

Density-based methods assume that outliers will appear in a region of low density, according to some non-parametric measure of density. Density-based approaches generally fall into global and local density methods. With

global density methods, the density around an observation is compared with some density measure for the entire dataset. If the observation's density is sufficiently low, it is considered an outlier. In contrast, local density methods compare the density of an observation to that of its neighbors, rather than the entire dataset. Density-based methods include LOF [8] and LOCI [43].

Cluster-based methods use a clustering of the data in order to detect outliers. The intuition behind these methods is that observations that do not fit well into clusters can be considered outliers. A recent example is GLOSH, which is based on the HDBSCAN* clustering hierarchy [9].

k-Nearest Neighbor Outlier Detection

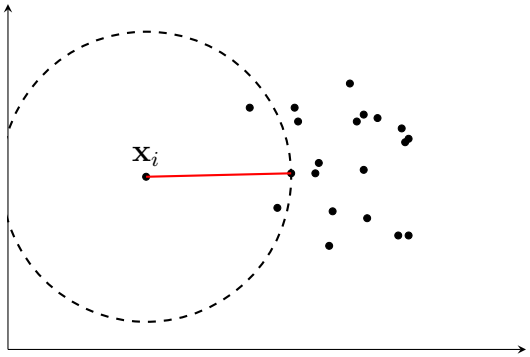


Figure 2.8: k-Nearest Neighbor Outlier Detection with $k = 3$.

The k-nearest neighbor outlier detection method, which we call kNN_{global}, has been originally introduced as an unsupervised distance-based outlier detection method [47]. Its score is the numerator of Equation (2.6):

$$\text{kNN}_{\text{global}}(\mathbf{x}_i, k) = d(\mathbf{x}_i, \text{NN}_{\text{k}}(\mathbf{x}_i)) \tag{2.7}$$

This makes the score global rather than local (see Figure 2.8).

Local Outlier Factor

Local Outlier Factor (LOF) [8] is an unsupervised outlier detection method which functions similarly to kNN_{local} by comparing the local density of an

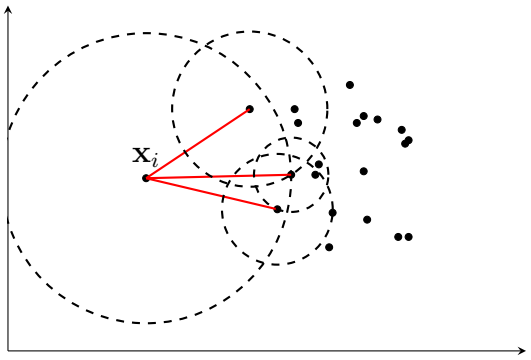


Figure 2.9: The intuition behind Local Outlier Factor is that it compares the local density of an object with the densities of its neighbors.

observation to that of its neighbors (Figure 2.9). The distances between observations are replaced by reachability distances, defined as:

$$\text{reach-dist}_k(\mathbf{x}_i \leftarrow \mathbf{x}_j) = \max\{d(\mathbf{x}_j, \text{NN}_k(\mathbf{x}_j)), d(\mathbf{x}_i, \mathbf{x}_j)\} \tag{2.8}$$

The local reachability density of an observation $\mathbf{x}_i$ is then defined as the inverse average reachability distance from the set of $\mathbf{x}_i$'s neighbors, $k\text{NN}(\mathbf{x}_i)$[1], that are within the k nearest neighbor distance around $\mathbf{x}_i$:

$$\text{lrd}_\text{k}(\mathbf{x}_i) = \frac{|\, k\text{NN}(\mathbf{x}_i)|}{\sum_{\mathbf{x}_j \in k\text{NN}(\mathbf{x}_i)} \text{reach-dist}_k(\mathbf{x}_i \leftarrow \mathbf{x}_j)} \tag{2.9}$$

Finally, the LOF score of an observation is computed by comparing the lrd of the observation with that of its neighbors:

$$\text{LOF}_k(\mathbf{x}_i) = \frac{\sum_{\mathbf{x}_j \in k\text{NN}(\mathbf{x}_i)} \frac{\text{lrd}_k(\mathbf{x}_j)}{\text{lrd}_k(\mathbf{x}_i)}}{|\, k\text{NN}(\mathbf{x}_i)|} \tag{2.10}$$

Local Correlation Integral

Local Correlation Integral (LOCI) [43] is an unsupervised outlier detection method which analyzes the density of an observation at multiple neighborhood radii αr of a given maximum radius r, where $\alpha \in (0, 1]$. Figure 2.10 shows this analysis at one such radius. For each observation $\mathbf{x}_i$, a (local) r-neighborhood

[1] In most cases $k\text{NN}(\mathbf{x}_i)$ will have k objects, however in rare cases where there is a "tie," it can be greater than k.

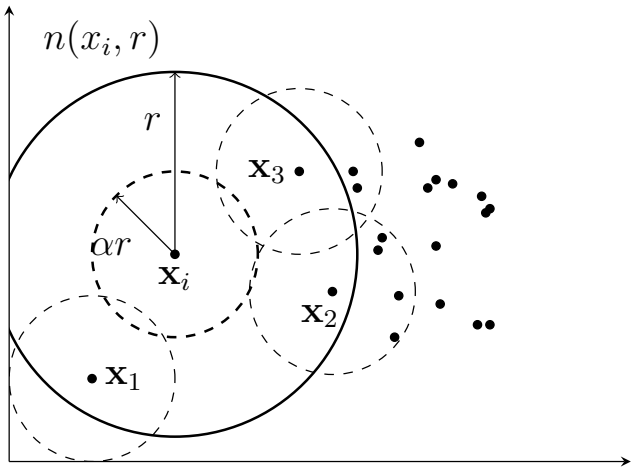


Figure 2.10: Local Correlation Integral also compares an object's local density with its neighbors, but at multiple levels of granularity (adapted from [43]).

$\mathcal{N}(\mathbf{x}_i, r) = \{\mathbf{x} | d(\mathbf{x}_i, \mathbf{x}) \leq r\}$ and a (local) r-density $n(\mathbf{x}_i, r) = |\mathcal{N}(\mathbf{x}_i, r)|$ are defined.

The average αr-density inside an r-neighborhood around an observation $\mathbf{x}_i$ is then defined as:

$$\hat{n}(\mathbf{x}_i, r, \alpha) = \frac{\sum_{\mathbf{x}_j \in \mathcal{N}(\mathbf{x}_i, r)} n(\mathbf{x}_j, \alpha r)}{n(\mathbf{x}_i, r)} \tag{2.11}$$

and the multi-granularity deviation factor (MDEF) is given by:

$$\text{MDEF}(\mathbf{x}_i, r, \alpha) = 1 - \frac{n(\mathbf{x}_i, \alpha r)}{\hat{n}(\mathbf{x}_i, r, \alpha)} \tag{2.12}$$

An observation $\mathbf{x}_i$ is classified using the following score:

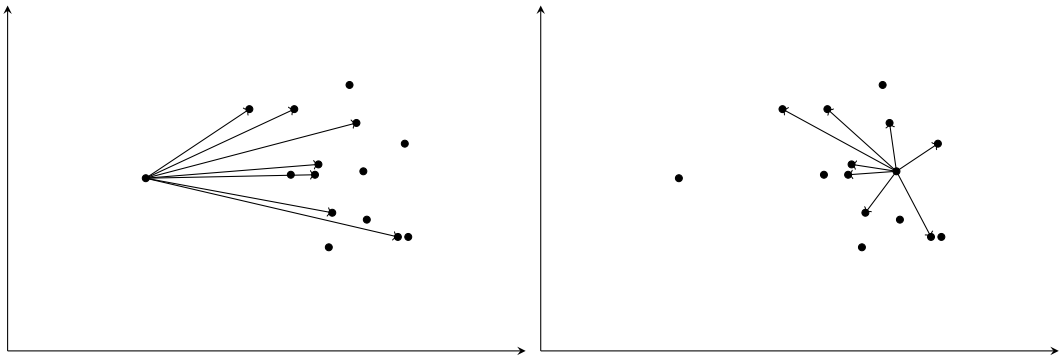
$$\sigma\,\text{MDEF}(\mathbf{x}_i, r, \alpha) = \frac{\sigma_{\hat{n}}(\mathbf{x}_i, r, \alpha)}{\hat{n}(\mathbf{x}_i, r, \alpha)}, \tag{2.13}$$

which is the normalized standard deviation $\sigma_{\hat{n}}(\mathbf{x}_i, r, \alpha)$ of $n(\mathbf{x}_i, \alpha r)$ for $\mathbf{x}_i \in \mathcal{N}(\mathbf{x}_i, r)$. With these quantities, the LOCI score is computed as follows:

$$\text{LOCI}(\mathbf{x}_i, \alpha) = \max_{r \in \mathcal{R}} \left\{ \frac{\text{MDEF}(\mathbf{x}_i, r, \alpha)}{\sigma\,\text{MDEF}(\mathbf{x}_i, r, \alpha)} \right\} \tag{2.14}$$

Angle-Based Outlier Detection

The intuition behind Angle-Based Outlier Detection (ABOD) [36] is that by measuring the variance in the angles between an observation and pairs of other



(a) Outlier (b) Inlier

Figure 2.11: The intuition behind Angle-Based Outlier Detection.

observations, we can determine whether or not an observation is an outlier. If the variance is high, it suggests that the observation is surrounded by other observations (in a cluster), while a low variance suggests that the observation is far away from other observations (an outlier). This is illustrated in Figure 2.11. The Angle-Based Outlier Factor (ABOF) is defined as follows:

$$\text{ABOF}(\mathbf{x}_i) = \text{VAR}_{\mathbf{x}_j, \mathbf{x}_k \in \mathcal{D}_{\text{train}}} \left(\frac{\langle \mathbf{x}_i - \mathbf{x}_j, \mathbf{x}_i - \mathbf{x}_k \rangle}{\|\mathbf{x}_i - \mathbf{x}_j\|^2 \cdot \|\mathbf{x}_i - \mathbf{x}_k\|^2} \right) \tag{2.15}$$

2.3 Evaluating One-Class Classification

In order to evaluate an algorithm's performance in one-class classification and compare it to other algorithms, we must first determine a way of measuring the performance of an algorithm.

When predicting an object's class with a one-class classifier, we either get a classification label {inlier, outlier} or a real-valued score, which can take the form of a probability or some metric by which the classifier quantitatively expresses a degree of "inlierness." We can then apply a threshold to this output in order to classify the object, where values above this threshold result in a classification of "inlier", while values below will result in "outlier".

When evaluating a classifier's performance on some dataset where we are given labels, we can compare the predicted labels provided by the classifier with the true labels in the dataset. In doing this, there are four possible

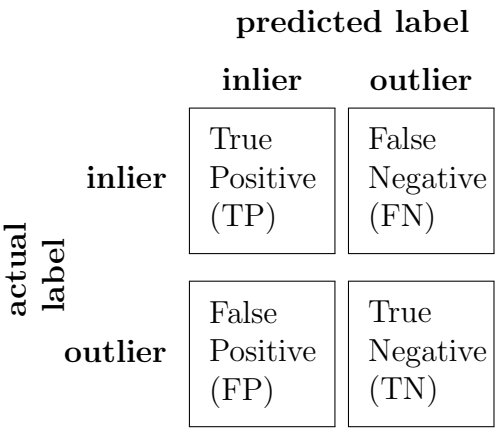


Figure 2.12: A confusion matrix representing the four possible outcomes of a one-class classifier's predictions compared to the true labels.

outcomes: true positive, false negative, false positive, and true negative. These possibilities can be depicted with a confusion matrix (Figure 2.12).

Using these four outcomes we define two measures which will be used to evaluate the performance of one-class classifiers. The true positive rate (TPR) is the fraction of true inliers which the classifier correctly predicts as inliers:

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{2.16}$$

where a score of 1 means all inlier objects are correctly identified by the classifier, and a score of 0 means all inlier objects were incorrectly labeled as outliers by the classifier. The false positive rate (FPR) is the fraction of true outliers which the classifier incorrectly predicts as inliers:

$$\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP} + \mathrm{TN}} \tag{2.17}$$

By varying the threshold applied to a classifier's output for a labeled dataset, there is a trade-off between the true positive rate and false positive rate obtained. At one extreme, the classifier can label every object as an inlier, achieving a perfect true positive rate of 1, but a false positive rate of 1 as well. At the other extreme, the classifier can label every object as an outlier, achieving a 0 for both rates. This trade-off can be seen in a Receiver Operating Characteristic (ROC) curve (Figure 2.13).

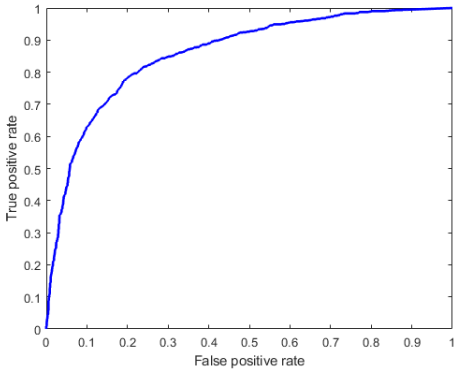


Figure 2.13: Receiver Operating Characteristic curve for a one-class classifier on an example dataset.

If we integrate the ROC curve over all threshold values, we obtain the area under the ROC curve (AUC) [7], which quantifies the trade-off between the true positive rate and false positive rate, and allows us to evaluate a classifier's performance without specifying a threshold. The AUC represents the probability that a random inlier object is correctly given a higher score than a random outlier object. Intuitively, this means that when ranking predictions made by a classifier, true inlier objects with lower scores than any true outlier will decrease the AUC. A perfect AUC of 1 is achieved when every true inlier object is given a higher score than every true outlier object.

In the datasets used in this thesis, many are originally multi-class datasets with two or more classes, and are converted into two or more one-class datasets using the procedure outlined in Chapter 4. When we report a classifier's performance on these datasets, the results from these one-class datasets are aggregated using the weighted AUC (WAUC) measure [18]:

$$\text{WAUC} = \sum_{i=1}^{m} p(i)\,\text{AUC}_i \tag{2.18}$$

where m is the number of classes in a dataset used as inliers, AUC(i) is the classifier's AUC score for the dataset with inlier class i, and $p(i)$ represents the fraction of objects belonging to class i among all the objects in the original dataset that belong to classes used as inliers.

Chapter 3

GLOSH for One-Class Classification

3.1 Adapting Unsupervised Outlier Detection to One-Class Classification

Common to unsupervised outlier detection methods is that they compute a certain outlier score for each observation. To use an unsupervised outlier detection method in one-class classification, the general strategy is as follows: First run the unsupervised method on the (one-class) training data, pre-computing the scores for each inlier. Then compute the score for a new observation to be classified, possibly using other pre-computed quantities (*e.g.*, densities or distances to nearest neighbors) related to observations in the training data. Then, in order to classify the new observation, compare its score with the pre-computed scores for the inliers.

There are two important aspects related to the use of pre-computed quantities that involve training data when computing the outlier score for a new observation to be classified. First, when classifying multiple observations, there is no need to recompute these quantities over and over again for each new observation, since they relate solely to the training data (inlier class model) and can thus be pre-computed, which makes computations faster.

Second and more importantly, using pre-computed quantities regarding the observations in the training data assures that the model is by no means affected by new observations to be classified. This is a basic principle of one-class

classification, i.e., unlabeled observations should not affect the pre-computed inlier class model, since they may be outliers. For example, suppose that a certain algorithm operates by comparing the density of a new observation to be classified against the densities of its nearest neighbors among the known inliers (training data). In this case, the densities of the inliers should be pre-computed, not to be affected by the presence of the unlabeled observation being currently assessed; otherwise, each unlabeled observation would affect the model in a different way, which means that different observations would be classified by different models/criteria.

When classifying multiple observations, it is also recommended that the classification procedure described above be performed independently for each observation. This way, different unlabeled observations will not affect each other's assessment. The reason we only classify one observation at a time instead of multiple observations at once is because we make no assumptions about the nature of each observation in relation to the combined dataset as a whole. It is possible that observations to be classified, while outliers in the sense that they do not belong to the inlier class, may be grouped together in such a way that an unsupervised method would not detect them as ouliers.

3.2 HDBSCAN* and GLOSH

3.2.1 HDBSCAN*

Hierarchical DBSCAN* (HDBSCAN*) [9] is a hierarchical clustering method that follows Hartigan's model of density contour clusters or trees and can be seen as an improvement over OPTICS [2], an earlier hierarchical clustering method. It provides as a result a complete clustering hierarchy composed of all possible density-based clusters following the non-parametric model adopted for an infinite range of density thresholds, from which a simplified cluster tree can be easily extracted by using Hartigan's concept of rigid clusters [23].

DBSCAN* is a density-based clustering algorithm similar to DBSCAN [16], where an object, $\mathbf{x}_i$, is called a core object with respect to the user-defined parameters, ε, and m_{pts}, if it contains at least m_{pts} objects in a minimum

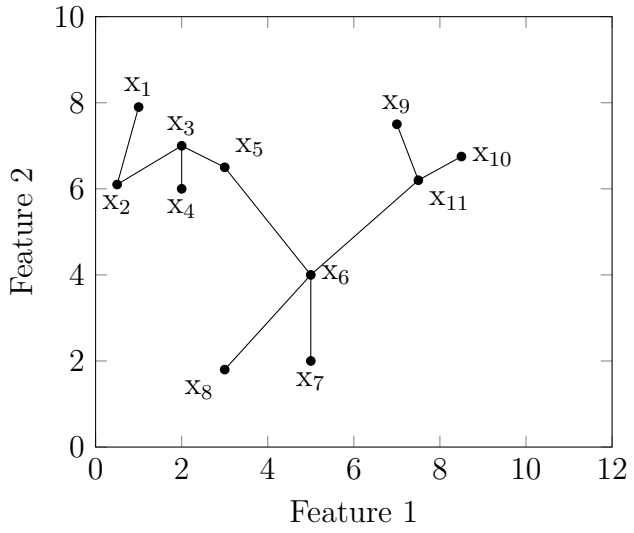


Figure 3.1: Objects and edges of an MST computed using mutual reachability distances with $m_{pts} = 3$.

radius, ε. Otherwise the object is called noise. Based on this definition the core objects are clustered with respect to ε and m_{pts} in non-empty maximal subsets, such that every pair of objects in a cluster is density-connected, meaning they are directly or transitively reachable by the radius, ε (ε-reachable). The original definitions of DBSCAN also include the concept of border objects which is not used by DBSCAN*.

HDBSCAN* has as its input parameters a value for m_{pts}, as well as one for m_{clSize} which controls the minimum cluster size. For a proper formulation of the density-based hierarchy with respect to a value of m_{pts}, it employs notions related to the core and reachability distances introduced for OPTICS. While the notion of core distance, $d_{core}(\mathbf{x}_i)$, is the same as for OPTICS, the minimum radius, ε, such that $\mathbf{x}_i$ becomes a core object with respect to m_{pts}, the reachability distance follows the symmetric definition of reachability distance (mutual reachability distance), defined as: $d_{mreach}(\mathbf{x}_i, \mathbf{x}_j) = \max\{d_{core}(\mathbf{x}_i), d_{core}(\mathbf{x}_j), d(\mathbf{x}_i, \mathbf{x}_j)\}$ [37]. Using those notions, a mutual reachability graph is built in which the objects of the dataset are vertices and the weight of each edge is the mutual reachability distance (with respect to $mpts$) between the respective pair of objects.

Figure 3.1 shows a Minimum Spanning Tree (MST) computed using mutual

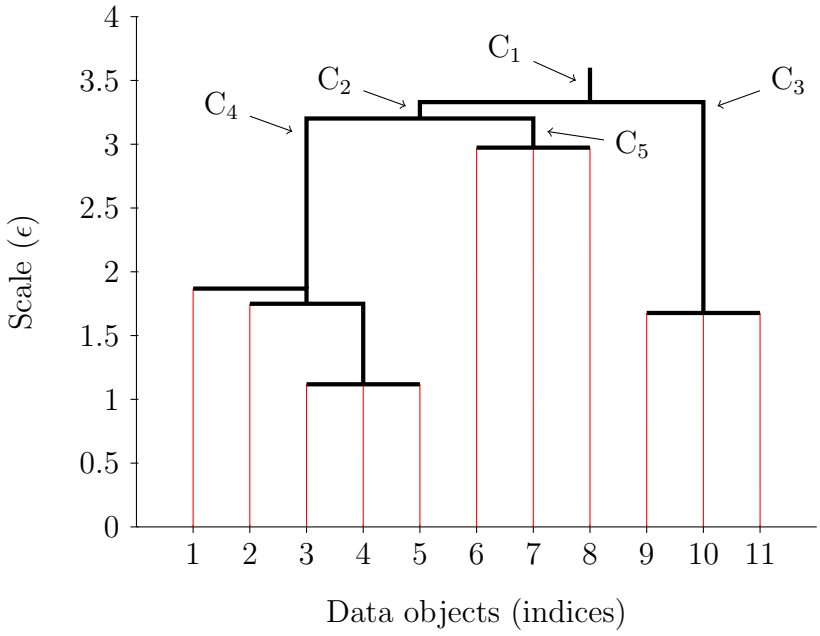


Figure 3.2: Dendrogram representing a hierarchical clustering produced by applying HDBSCAN* to the dataset in Figure 3.1 with $m_{pts} = 3$ and $m_{clSize} = 3$. The thinner red lines correspond to noise.

reachability distances. Removing edges in decreasing order of weight from the mutual reachability graph can produce a hierarchy of all DBSCAN* clusterings for any $\varepsilon \in [0, \infty)$, where clusters according to DBSCAN* with respect to m_{pts} and ε are then the connected components of core objects and the remaining objects are noise. A hierarchical clustering can be represented using a dendrogram as seen in Figure 3.2. By removing the edges with weights greater than some decreasing threshold from a complete proximity graph and then checking for the remaining connected subcomponents of the graph is essentially the graph-based definition of the hierarchical Single-Linkage algorithm [32], which formalizes the conceptual relationship between DBSCAN* and Single-Linkage in the transformed space of mutual reachability distances.

3.2.2 GLOSH

The Global-Local Outlier Score from Hierarchies (GLOSH) [9] is an unsupervised outlier detection algorithm based on the hierarchical density estimates provided by HDBSCAN* where for each query object the scope of a reference

set of objects is chosen dynamically and based on the closest structure within the density-based hierarchy. This allows it to classify outliers on a local or global scale depending on the hierarchy structure and the object's cluster assignment in the hierarchy. In all cases, the most meaningful reference structure for an object is the "nearest" cluster in the hierarchical cluster hierarchy, as opposed to all other objects in some region around the object. This avoids comparing outliers to other outliers or unrelated clusters as could happen for a local method; it also enables the user to find both global and local outliers.

In order to classify an object $\mathbf{x}_i$ with GLOSH, we must determine the nearest cluster (with respect to m_{pts} and m_{clSize}) to the object, using the densest object in the cluster as a referential density for the cluster with which to compare our object. Given the density of $\mathbf{x}_i$, $\lambda(\mathbf{x}_i)$, the closest cluster from a density-based perspective is the cluster $C_{\mathbf{x}_i}$ which $\mathbf{x}_i$ is assigned to in the hierarchy at $\lambda(\mathbf{x}_i)$. The cluster $C_{\mathbf{x}_i}$, is simply the first cluster $\mathbf{x}_i$ is assigned as it switches from noise to core when we increase the density threshold bottom-up through the hierarchy. We use the densest object in $C_{\mathbf{x}_i}$, $\mathbf{x}_l$, as the referential object to compare to the density of $\mathbf{x}_i$ as it can be considered the most "inlierly" object in the cluster and can serve as a standard for other objects with $C_{\mathbf{x}_i}$ as their closest cluster. We can then use the density of $\mathbf{x}_l$ as the referential density for $C_{\mathbf{x}_i}$ for every object $\mathbf{x}_i$ that has $C_{\mathbf{x}_i}$ as its closest density-based cluster. We can now define the GLOSH score of an object:

$$\text{GLOSH}(\mathbf{x}_i) = \frac{\lambda_{\max}(C_{\mathbf{x}_i}) - \lambda(\mathbf{x}_i)}{\lambda_{\max}(C_{\mathbf{x}_i})} \tag{3.1}$$

where $\lambda(\mathbf{x}_i)$ is the density of $\mathbf{x}_i$ and $\lambda_{\max}(C_{\mathbf{x}_i})$ is the density of the highest density object $\mathbf{x}_l \in C_{\mathbf{x}_i}$, where densities are defined by the density threshold ε, the density level where an object is labeled noise in the HDBSCAN* hierarchy. The closest cluster $C_{\mathbf{x}_i}$ is the one that $\mathbf{x}_i$ belongs to at the density level of $\mathbf{x}_i$.

The output of GLOSH are scores in the range $[0, 1)$, where a score of 0 means the density of an object $\mathbf{x}_i$ is equivalent to that of the densest object in $C_{\mathbf{x}_i}$, while a score near 1 means that $\mathbf{x}_i$ has a much lower density than $C_{\mathbf{x}_i}$.

3.3 Adapting GLOSH to One-Class Classification

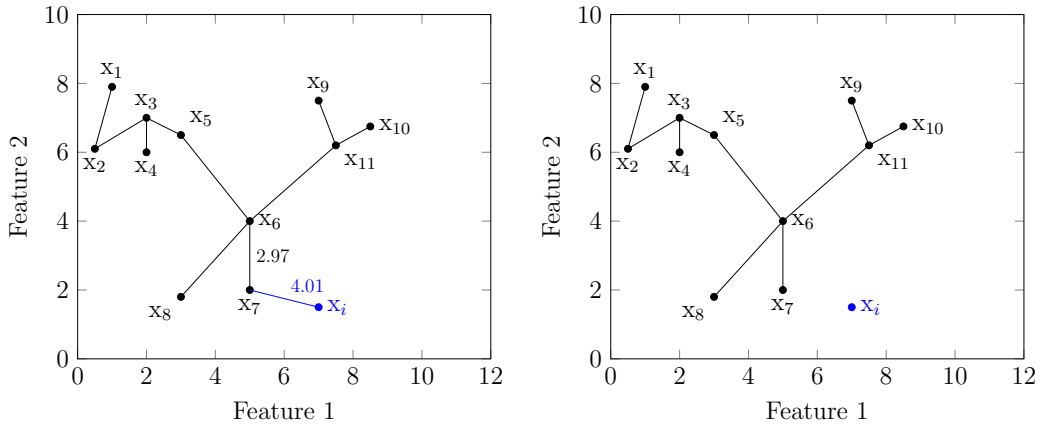
One of the contributions of this thesis is the adaptation of GLOSH to the one-class classification scenario. The motivation for doing so is to continue the investigation of unsupervised outlier detection methods for one-class classification, given the success of other unsupervised methods in the past. By taking advantage of potential hierarchical structures in the data, we may be able to find semantically meaningful outliers that other methods would have difficulty detecting. Additionally, this would expand the utility of HDBSCAN* for use in one-class classification with little added computational cost.

We can use the procedure described in Section 3.1 to apply GLOSH to one-class classification. Given the one-class training data, we construct a density-based hierarchy using HDBSCAN*, supplying the parameters m_{pts} and m_{clSize}, and store relevant information such as each cluster assignment given to each training object as we vary the density threshold, as well as the referential density for each cluster, $\lambda_{\max}(C)$.

In order to classify an incoming object, we must determine which cluster it is last assigned to before becoming noise as the density threshold is increased. Since adding a new object to the HDBSCAN* clustering hierarchy can potentially change the clustering results, we instead fix the clustering hierarchy as our inlier class model, and estimate the assigned cluster by some approximation.

While there are potentially many ways to approximate an object's cluster assignment and GLOSH score, in this thesis we use the following approach: we conceptually consider the effect of connecting an object $\mathbf{x}_i$ to its nearest neighbor $\mathbf{x}_j$ (according to mutual reachability distance) with an edge equal to $d_{mreach}(\mathbf{x}_i, \mathbf{x}_j)$, and what cluster assignment results from this connection.

If we track this connection as we remove edges in decreasing order of weight from the mutual reachability graph, we note that there are two situations that may occur. The first situation is that the edge connecting $\mathbf{x}_i$ and $\mathbf{x}_j$ is removed before $\mathbf{x}_j$ is separated from the original mutual reachability graph. In this



(a) $\epsilon > 4.01$ (b) $\epsilon < 4.01$

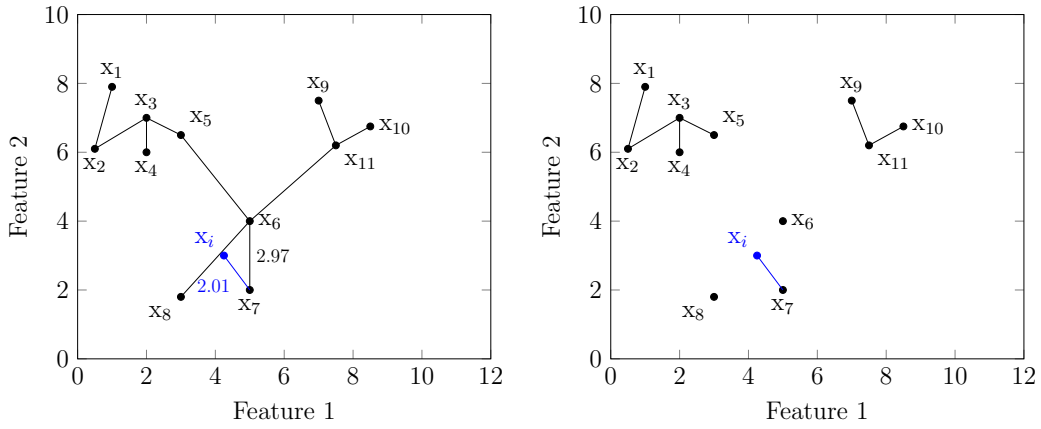
Figure 3.3: MST based on mutual reachability distances with $m_{pts} = 3$ using the dataset in Figure 3.1 with the addition of a new observation $\mathbf{x}_i$ with edges removed just above and below the density threshold ϵ corresponding to the edge connecting object $\mathbf{x}_i$ and $\mathbf{x}_7$. Since $d_{mreach}(\mathbf{x}_i, \mathbf{x}_7) > d_{mreach}(\mathbf{x}_6, \mathbf{x}_7)$, the edge between $\mathbf{x}_i$ and $\mathbf{x}_7$ is removed before $\mathbf{x}_7$ becomes considered noise and so the closest cluster to $\mathbf{x}_i$ is the one $\mathbf{x}_7$ is assigned at this density level. In this case it is the cluster C_1 from Figure 3.2.

case, the closest cluster to $\mathbf{x}_i$ will be the cluster $\mathbf{x}_j$ belongs to at this density threshold, and we can use this cluster to compute the GLOSH score for $\mathbf{x}_i$. An example of this situation is seen in Figure 3.3.

The second situation is that $\mathbf{x}_j$ is separated from the mutual reachability graph before the edge between $\mathbf{x}_i$ and $\mathbf{x}_j$ is removed. In this case, the closest cluster for $\mathbf{x}_i$ is the same as the closest cluster for $\mathbf{x}_j$. This is demonstrated in Figure 3.4.

In order to quickly and easily classify an object in the one-class classification setting with this adaptation of GLOSH, we can perform the following procedure:

After performing HDBSCAN* on our inlier training data $\mathcal{X}_{tr}$ with supplied values for m_{pts} and m_{clSize}, we store the densities $\lambda(\mathbf{x}_j)$ for all $\mathbf{x}_j \in \mathcal{X}_{tr}$, the cluster assignments for each $\mathbf{x}_j$ that they are assigned to in the density-based hierarchy at all density levels, the referential densities for each cluster, the GLOSH scores for $\mathcal{X}_{tr}$ if they will be used to compare to new objects, as well as the original training data. When classifying a new object $\mathbf{x}_i$, we



(a) $\epsilon > 2.01$ (b) $\epsilon < 2.01$

Figure 3.4: MST based on mutual reachability distances with $m_{pts} = 3$ using the dataset in Figure 3.1 with the addition of a new observation $\mathbf{x}_i$ with edges removed just above and below the density threshold ϵ corresponding to the edge connecting object $\mathbf{x}_i$ and $\mathbf{x}_7$. Since $d_{mreach}(\mathbf{x}_i, \mathbf{x}_7) < d_{mreach}(\mathbf{x}_6, \mathbf{x}_7)$, the edge between $\mathbf{x}_6$ and $\mathbf{x}_7$ is removed before $\mathbf{x}_i$ becomes considered noise and so the closest cluster to $\mathbf{x}_i$ is the closest cluster for $\mathbf{x}_7$. In this case it is the cluster C_5 from Figure 3.2.

compute $d_{core}(\mathbf{x}_i)$ with m_{pts} relative to $\mathcal{X}_{tr}$ and use this to compute the mutual reachability distances, $d_{mreach}(\mathbf{x}_i, \mathbf{x}_j)$ for all $\mathbf{x}_j \in \mathcal{X}_{tr}$. Note that the core distances for the objects in $\mathcal{X}_{tr}$ are not changed by $\mathbf{x}_i$. We then find the nearest neighbor of $\mathbf{x}_i$, $\text{NN}_{mreach}(\mathbf{x}_i)$ based on mutual reachability distance. If $d_{mreach}(\mathbf{x}_i, \text{NN}_{mreach}(\mathbf{x}_i))$ is greater than the density threshold ϵ associated with $\lambda(\text{NN}_{mreach}(\mathbf{x}_i)))$, then $\mathbf{x}_i$ will become noise before $\text{NN}_{mreach}(\mathbf{x}_i)$ and we must find the cluster assignment for $\text{NN}_{mreach}(\mathbf{x}_i)$ as in Figure 3.3, which we have saved. Otherwise, if $d_{mreach}(\mathbf{x}_i, \text{NN}_{mreach}(\mathbf{x}_i))$ is less than the density threshold ϵ associated with $\lambda(\text{NN}_{mreach}(\mathbf{x}_i)))$, then the closest cluster for $\mathbf{x}_i$ is the same as for $\text{NN}_{mreach}(\mathbf{x}_i)$, as in Figure 3.4.

Once we have identified the closest cluster for $\mathbf{x}_i$, we can compute its GLOSH score using Equation 3.1.

Using this approach we satisfy the principles established in Section 3.1. We only need to perform the HDBSCAN* clustering once on our training data, and store all the necessary computed quantities to perform classification. Furthermore, we ensure that the model does not change with new observations.

Chapter 4

On the Evaluation of Outlier Detection and One-Class Classification Methods

4.1 Introduction

In this chapter[1], we focus on the comparison of one-class classification methods with unsupervised outlier detection methods adapted to the problem of one-class classification. Following the approach proposed in [29], unsupervised outlier detection methods can be extended to use inlier class information to be applicable also in the semi-supervised setting.

Janssens *et al.* [28] performed a comparative study between 3 methods proposed for one-class classification (kNN Data Description [48], Parzen Windows [44] and SVDD [54]) and 2 methods originally proposed for unsupervised outlier detection i.e. LOF [8] and LOCI [43] extended to the one-class classification scenario. The authors concluded that LOF and SVDD are the top two performers and that they are not distinguishable from a statistical significance test perspective.

In this chapter, we perform a more comprehensive investigation, using a more rigorous experimental setup, which leads to conclusions that do not fully agree with Janssens *et al.* [28]. In particular, we make the following

[1]The work in this chapter has been published as L. Swersky, H. O. Marques, J. Sander, R. J. Campello, and A. Zimek, "On the Evaluation of Outlier Detection and One-Class Classification Methods," 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Montreal, QC, 2016, pp. 1-10, with small changes for the thesis format.

contributions:

- When reproducing the experiments in [28], which reports averages over 5 repetitions of 5-fold cross-validation, we noticed a large variability in the results. In order to increase results confidence, we perform 30 repetitions using 10-fold cross-validation instead.

- We increase the number of datasets used in the evaluation from 24 to 433 (33 base datasets plus 400 dataset variants from an image collection), and the number of compared methods from 5 to 11.

- In addition to the well-known Area Under the ROC curve (ROC AUC), used in Janssens *et al.* [28], we also use Adjusted Precision-at-n (AdjustedPrec@n), as defined in [10], to measure performance. These measures complement each other and together give a more complete picture of the performance characteristic of a method [10].

- In addition to the type of experiment performed by Janssens *et al.* [28], where one class is labeled as inlier and the other classes are relabeled outliers, we also conduct a second type of experiment where one class is labeled as outlier and the other classes are re-labeled inliers. Both types represent possible real application scenarios. We show that some of the conclusions change depending on the type of experiment.

- We also include an adaptation of a recent outlier detection method — called GLOSH [9] — to the one-class classification problem.

- Lastly, we discuss basic principles of one-class classification that should not be violated when adapting unsupervised outlier detection methods to this task. We carefully examined and adjusted all the codes used in our experiments, making sure that these principles were not violated when producing the reported results.

4.2 Experimental Setup

We compare and evaluate the 11 algorithms described in Chapter 2: ABOD, Auto-Encoder, Gaussian Density, GLOSH, kNN_{global}, kNN_{local}, LOCI, LOF, Linear Programming, Parzen Windows and SVDD. We use code from the repository available at `http://prlab.tudelft.nl/users/david-tax/` [53] for most compared algorithms, except for LOF, LOCI, kNN_{local} and GLOSH. In the case of LOF and LOCI, their implementations were modified to ensure that new observations to be classified do not affect the pre-computed model for the inlier class, following the guidelines previously discussed in Section 3.1. As KNN_{local} was not available in that repository, we used our own implementation for this algorithm. GLOSH was adapted based on the implementation of HDBSCAN* made available at `http://lapad-web.icmc.usp.br/`.

We use 31 real-world datasets from the UCI Machine Learning Repository [38] as pre-processed for one-class classification and made available at `http://prlab.tudelft.nl/users/david-tax/`: Abalone, Arrhythmia, Balance-scale, Ball-bearing, Biomed, Breast, Cancer, Colon, Delft1x3, Delft2x2, Delft3x2, Delft5x1, Delft5x3, Diabetes, Ecoli, Glass, Heart, Hepatitis, Housing, Imports, Ionosphere, Iris, Liver, Satellite, Sonar, Spectf, Survival, Vehicle, Vowels, Waveform and Wine.

In addition, we use CellCycle-237 and YeastGalactose, made public by Yeung *et al.* [63], [64], as well as a collection of 400 datasets based on the Amsterdam Library of Object Images (ALOI) [19], created as described in [27]. Specifically, this collection has been created by randomly selecting 2, 3, 4 or 5 ALOI image categories as class labels and then sampling 25 images from each of the selected categories, thus resulting in datasets containing 2, 3, 4, or 5 classes and 50, 75, 100, or 125 images (observations). Following [27], these images are described by six descriptors: color moments (144 attributes), texture statistics extracted from the gray-level co-occurrence matrix (88 attributes), Sobel edge histogram (128 attributes), first-order statistics from the gray level histogram (5 attributes), gray-level run-length matrix features (44 attributes), and gray-level histogram (256 attributes). PCA is then applied to each set of attribute

vectors separately and the first principal component resulting from each set is extracted. The extracted first components are then combined in such a way that each image is thus described by a vector with six attributes.

In total, we have 433 *real-world* multi-class datasets. We average the results for the ALOI and Delft datasets, for they are variants obtained from the same source. Finally, due the inability of some algorithms to deal with replicated observations, duplicates are removed from the datasets where they are present.

In order to evaluate a method's performance on a one-class dataset, we perform the following procedure: First, we split the dataset into 2 subsets, one containing 20% and the other containing 80% of the data. In the subset with 80% of the data we apply a 10-fold cross-validation procedure to optimize the parameters of the methods with respect to ROC AUC.

The hyperparameters of the methods were optimized in the following ranges: $k = 1, 2, \cdots, 50$ for LOF, kNN_{global} and kNN_{local}; $m_{clSize} = m_{pts} = 1, 2, \cdots, 50$ for GLOSH; No. hidden units = 2, 5, 7, 10, 12, 15, 17, 20, 22, 25 for Auto-Encoder[2], $h = 0.001$ to 50 (discretized logarithmically in 25 different values) for Gaussian Density and for the Gaussian kernel used in SVDD, $\alpha = 0.1, 0.2, \cdots, 1.0$ for LOCI, LP, Parzen Windows and SVDD.

After hyperparameter optimization, the subset containing 20% of the data (test set) is used to measure the performances of the methods (trained with the optimal parameter values from the 10-fold cross-validation). In order to get more reliable results, this procedure is repeated 30 times, and the resulting ROC AUC values are aggregated and reported.

For the sake of comparison with the results reported by Janssens *et al.*, we also compute the Weighted ROC AUC measure used in their work [28], which gives more weight to results obtained from experiments involving larger inlier classes. It is worth remarking, however, that this approach may be questionable, since it is well known that ROC curves already inherently adjust for the imbalance of class sizes.

[2]Due to the high computational demand of Auto-Encoder, a time limit of 1000s was imposed to the network convergence, which gives a maximum amount of time equal to 30 repetitions $\times$ 10 folds $\times$ 10 parameters $\times$ 1000s $\approx$ 35 days per experiment involving a single dataset.

In addition to ROC AUC values, we also report the adjusted precision-at-n measure (AjustedPrec@n) [10] for the classification of results obtained on the test sets. While ROC AUC takes the entire test set into account, precision-at-n (fraction of true outliers among the top n outlier scores in the test set) evaluates only the top n observations, where n is the number of observations known to be outliers in the test set. To compare precision-at-n values across datasets with different numbers of outliers, one has to adjust precision-at-n by chance, as discussed in [10], which gives rise to the AjustedPrec@n measure used in our experiments.

A high ROC AUC score indicates only that, in the overall outlier ranking, outliers are more likely to be ranked ahead of inliers; it does not necessarily mean that the top positions in the ranking are dominated by outliers. Therefore, following the extensive study in [10], we argue that one cannot rely solely on ROC AUC scores in judging the quality of an outlier method; rather, ROC AUC and AdjustedPrec@n complement each other, as they reveal different aspects of an outlier ranking, both of which are relevant in practice.

We perform two major types of experiments. In the **first type** of experiment (Type I), we follow the only approach taken by Janssens et al. [28], where multi-class datasets are transformed into one-class datasets by re-labeling one class as inliers, and the remaining classes as outliers. Except for datasets where only a single inlier class has been pre-defined as such in the data repository (`http://prlab.tudelft.nl/users/david-tax/`) — *e.g.*, Ecoli — we repeat the procedure for every class as inliers in a dataset, and average the results.

For the **second type** of experiment (Type II), we reverse the inlier and outlier classes obtained in the first type of experiment for datasets that have more than 2 possible inlier classes defined. This type of experiment is important since it models situations with a possibly multi-modal inlier class. Note that Type II experiments are only different from Type I — and are therefore only reported — for datasets with 3 or more classes. For this reason, results for Type II experiments are only available for a subset of the datasets considered in Type I experiments.

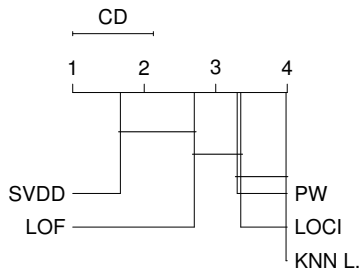


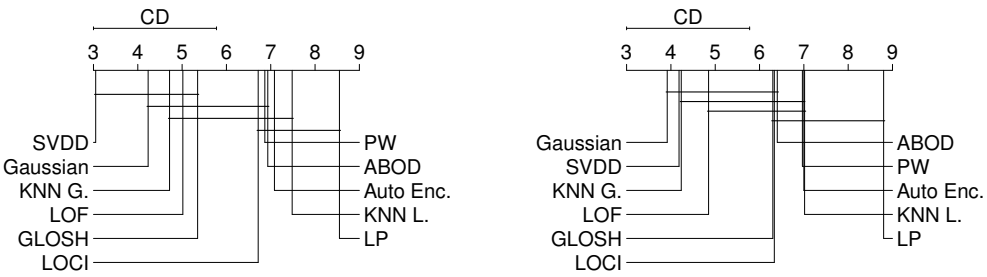
Figure 4.1: Average ranking of the methods compared in [28] over all Type I experiments w.r.t. weighted ROC AUC.

4.3 Results and Discussion

Figure 4.1 shows the average ranking of the 5 methods compared in [28] over all Type I experiments w.r.t. weighted ROC AUC. This figure summarizes our attempt at reproducing Janssens *et al.*'s results [28], which were restricted to this particular setup only. The width of the upper bar (CD) indicates the critical distance of the well-known Friedman/Nemenyi statistical test at significance level $\alpha = 0.05$. The horizontal bars within the plots indicate subsets of methods for which there is no significant difference according to the test. The analogous figure in [28] shows two subsets of methods: (1) the top performers SVDD, LOF, and KNN_{local} (with SVDD and LOF having exactly the same average rank), and (2) a group consisting of PW and LOCI, clearly separated from (1), with much lower performance. Our result does not agree in several respects: first, LOF and SVDD are not tied, second, there are no longer two clearly separated groups, and third, KNN_{local} is the worst performer rather than one of the best.

In the following, we describe the results according to our experimental setup described above. Detailed numbers for all experiments are given in tables 4.1, 4.2, 4.3, and 4.4. The overall ranking results are summarized in Figures 4.2 and 4.3.

Figure 4.2 shows the average rankings of the methods over all Type I experiments with respect to ROC AUC and AdjustedPrec@n. When looking at ROC AUC, one can see SVDD, Gaussian, and kNN_{global}, in this order, at the top. The average ROC AUC for SVDD, Gaussian, and kNN_{global} were

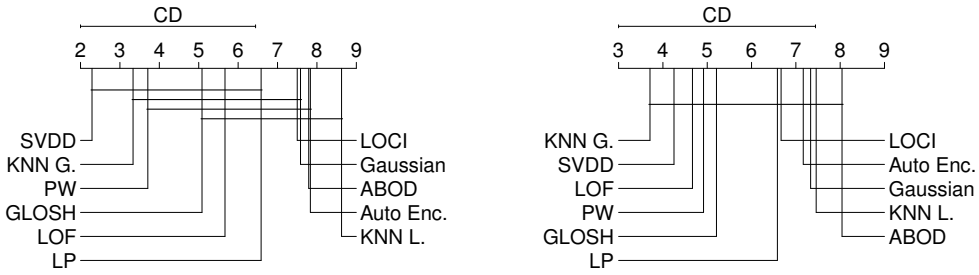


(a) ROC AUC (b) AdjustedPrec@n

Figure 4.2: Average rankings of the methods over all Type I experiments.

0.8, 0.8, and 0.78, respectively. When looking at AdjustedPrec@n the overall picture is similar, but AdjustedPrec@n tells us that SVDD is no longer the top performer as it is now outperformed by Gaussian. This suggests that in the outlier scoring produced by Gaussian there are on average more true outliers with top scores than in SVDD, whereas for SVDD the scores of the true outliers tend to be higher than those of inliers, overall.

Figure 4.3 shows the average rankings of the methods over all Type II experiments with respect to ROC AUC and AdjustedPrec@n. When comparing ROC AUC with AdjustedPrec@n, we can notice again some inversions in the ranks, *e.g.*, between LOF and PW/GLOSH. In particular, the relative performance of SVDD drops once more for AdjustedPrec@n, as it also does in Type I experiments, but now SVDD is outperformed by KNN_{global}, not by Gaussian. In fact, when comparing Type I experiments in Figure 4.2 and Type II experiments in Figure 4.3, a noticeable difference can be observed with respect to Gaussian: while it was among the top 3 performers in Type I experiments, its relative performance drops sharply in Type II. The absolute performance of Gaussian indeed drops from Type I to Type II experiments, from 0.8 to 0.77 for ROC AUC and from 0.48 to 0.38 for AdjustedPrec@n, which is expected as Gaussian presumes a unimodal inlier class model that fits best the data as arranged in Type I experiments. But this alone does not fully explain the drop of Gaussian in terms of relative performance. What also explains it is that other methods, particularly local density-based methods like GLOSH and LOF, perform better in Type II experiments (see the tables in the Appendix



(a) ROC AUC

(b) AdjustedPrec@n

Figure 4.3: Average rankings of the methods over all Type II experiments.

for detailed values), which correspond to application scenarios with possibly multi-modal target classes.

4.4 Conclusion

Overall, based on both types of experiments, we conclude that: (i) in agreement with [28], SVDD is a top performer, especially with respect to ROC AUC; (ii) kNN_{global}, however, might be a preferable choice, since it is consistently a top performer, yet it is much simpler than SVDD; this method was not included in the study in [28]; and (iii) in contrast to [28], LOF does not perform as strongly as SVDD, and kNN_{local} is not among the top performers, but consistently among the worst.

In this chapter we provided a comprehensive comparison of one-class classification algorithms and unsupervised outlier detection methods extended to the one-class classification scenario. These methods were evaluated over a number of datasets, measuring performance with respect to ROC AUC and AdjustedPrec@n in different experimental settings. The most important conclusion is that SVDD and kNN_{global} are the top choices for one-class classification, while we do not recommend kNN_{local}. This is in contrast to the previous comparison study by Janssens *et al.* [28], which did not include kNN_{global} and reported kNN_{local} as a top performer. In addition, we could not confirm the top performance of LOF reported in [28], but only that of SVDD.

As additional contributions, we proposed and described an adaptation of a

Table 4.1: First type of experiments — ROC AUC

Dataset	GLOSH	KNN L.	LP	ABOD	Auto Enc.	Gaussian	KNN G.	LOCI	LOF	PW	SVDD
Abalone	0.66	0.66	0.71	0.71	0.72	0.74	0.73	0.76	0.66	0.74	**0.77**
Aloi	0.98	0.97	0.95	**0.99**	0.98	**0.99**	**0.99**	0.98	0.98	0.98	0.98
Arrhythmia	0.63	0.58	0.5	0.51	0.53	0.55	0.52	0.53	0.6	0.5	0.64
Balance-Scale	0.88	0.86	0.88	0.86	0.91	**0.93**	0.87	0.87	0.92	0.87	0.91
Ball-Bearing	0.98	0.97	0.5	0.93	**1**	**1**	0.98	0.96	0.99	0.53	0.98
Biomed	**0.84**	0.81	0.51	0.65	0.72	0.8	**0.84**	0.79	0.71	0.69	0.7
Breast	0.96	0.93	0.81	0.93	0.91	**0.98**	0.96	**0.98**	0.96	0.8	**0.98**
Cancer	0.52	0.52	0.5	0.53	0.54	**0.59**	0.54	0.53	0.53	0.51	0.53
CellCycle237	0.81	0.72	0.74	0.81	0.76	0.82	**0.84**	0.72	0.81	0.74	0.83
Colon	0.67	0.63	0.5	0.64	0.58	0.67	0.66	0.59	**0.68**	0.5	**0.68**
Delft	0.95	**0.96**	0.93	0.68	0.83	0.95	0.93	0.89	**0.96**	0.93	**0.96**
Diabetes	0.65	0.63	0.51	0.61	0.62	0.64	0.62	0.63	**0.66**	0.59	0.65
Ecoli	**0.94**	0.93	0.92	0.93	0.89	**0.94**	**0.94**	**0.94**	**0.94**	**0.94**	**0.94**
Glass	0.78	0.78	0.81	0.79	0.76	0.78	0.81	0.67	0.81	0.81	**0.82**
Heart	0.6	0.57	0.5	0.59	0.67	**0.73**	0.58	0.58	0.59	0.55	0.61
Hepatitis	0.56	0.53	0.5	0.56	0.73	**0.74**	0.56	0.55	0.54	0.56	0.57
Housing	0.65	0.65	0.58	0.68	0.76	**0.78**	0.69	0.66	0.65	0.7	0.7
Imports	0.7	0.68	0.81	0.66	0.69	0.65	0.71	0.73	0.78	**0.83**	0.74
Ionosphere	**0.74**	0.66	0.66	0.64	0.62	0.64	0.67	0.6	0.64	0.64	**0.74**
Iris	0.97	0.95	**0.98**	0.97	0.96	**0.98**	0.97	0.97	0.97	**0.98**	**0.98**
Liver	0.54	0.55	0.53	0.55	0.54	0.54	0.55	**0.58**	0.55	0.53	0.56
Satellite	0.95	0.92	0.5	0.95	0.9	0.94	**0.96**	0.94	0.93	0.92	**0.96**
Sonar	0.7	0.73	0.76	0.64	0.67	0.66	0.76	0.65	**0.77**	0.76	0.75
Spectf	**0.66**	0.61	0.5	0.56	0.57	0.55	0.52	0.61	0.63	0.64	**0.66**
Survival	0.61	0.58	0.56	0.56	0.57	0.58	0.62	0.62	0.61	0.55	**0.65**
Vehicle	0.75	0.77	0.5	0.76	0.83	**0.9**	0.79	0.76	0.77	0.79	0.82
Vowels	0.99	0.98	**1**	0.99	0.63	0.99	**1**	0.91	0.99	**1**	0.99
Waveform	0.89	0.85	0.87	0.9	0.86	**0.91**	0.89	0.89	0.88	0.88	**0.91**
Wine	0.86	0.85	0.57	0.88	0.85	**0.96**	0.86	0.86	0.86	0.83	0.87
YeastGalactose	**0.99**	**0.99**	0.98	**0.99**	0.97	0.98	**0.99**	0.75	**0.99**	**0.99**	**0.99**
Average Rank	5.35	7.48	8.55	6.93	7.08	4.23	4.72	6.72	5.02	6.87	3.05

Table 4.2: Second type of experiments — ROC AUC

Dataset	GLOSH	KNN L.	LP	ABOD	Auto Enc.	Gaussian	KNN G.	LOCI	LOF	PW	SVDD
Abalone	0.62	0.62	0.7	0.67	0.65	0.7	0.68	0.67	0.63	0.71	**0.73**
Aloi	**0.96**	0.94	0.95	0.92	0.94	0.92	**0.96**	0.92	**0.96**	**0.96**	**0.96**
Balance-Scale	0.82	0.8	0.79	0.79	0.77	0.78	0.83	0.81	**0.84**	0.82	0.81
CellCycle237	**0.81**	0.69	0.73	0.75	0.76	0.76	0.79	0.71	0.75	0.74	0.78
Glass	0.75	0.71	0.76	0.69	0.69	0.64	0.76	0.7	**0.77**	0.76	**0.77**
Iris	0.95	0.93	**0.97**	0.96	0.94	0.82	0.96	0.95	0.93	**0.97**	**0.97**
Satellite	**0.85**	0.79	0.5	0.74	0.77	0.74	0.84	0.77	0.82	0.82	**0.85**
Vehicle	0.68	0.63	0.5	0.61	0.66	0.71	0.74	0.7	0.67	0.72	**0.75**
Vowels	0.94	0.94	**0.98**	0.75	0.7	0.64	**0.98**	0.9	0.95	**0.98**	**0.98**
Waveform	0.81	0.69	0.75	0.75	0.62	0.76	0.81	0.83	0.75	0.77	**0.86**
Wine	0.74	0.73	0.58	0.76	0.84	**0.85**	0.75	0.71	0.74	0.76	0.77
YeastGalactose	0.95	0.91	**0.97**	0.93	0.93	0.91	**0.97**	0.93	0.95	**0.97**	**0.97**
Average Rank	5.08	8.63	6.58	7.79	7.83	7.58	3.33	7.25	5.67	3.71	2.29

Table 4.3: First type of experiments — AdjustedPrec@n

Dataset	GLOSH	KNN L.	LP	ABOD	Auto Enc.	Gaussian	KNN G.	LOCI	LOF	PW	SVDD
Abalone	0.21	0.17	0.3	0.32	0.29	0.32	0.33	0.37	0.2	0.31	**0.4**
Aloi	0.92	0.87	0.85	0.93	0.92	**0.95**	0.94	0.91	0.91	0.92	0.92
Arrhythmia	-0.19	0.14	-0.77	0.03	0.08	0.09	0.05	0.05	**0.17**	-0.77	-0.19
Balance-Scale	0.44	0.5	0.53	0.51	0.61	0.59	0.44	0.55	0.61	0.56	**0.64**
Ball-Bearing	0.84	0.78	-0.28	0.7	0.97	**0.98**	0.85	0.76	0.88	-0.2	0.84
Biomed	**0.57**	0.52	-0.54	0.31	0.4	0.54	**0.57**	0.45	0.37	0.06	0.08
Breast	0.84	0.75	0.43	0.75	0.72	0.87	0.84	**0.88**	0.83	0.38	0.87
Cancer	0.05	0.06	-0.32	0.06	0.03	0.08	**0.11**	-0.01	0.08	-0.32	-0.04
CellCycle237	0.41	0.32	0.4	0.45	0.42	0.51	0.5	0.34	0.44	0.45	**0.52**
Colon	0.2	0.16	-0.62	0.18	0.13	0.21	0.21	-0.17	0.2	-0.62	**0.27**
Delft	0.73	0.73	0.67	0.29	0.48	0.71	0.67	0.63	**0.77**	0.67	0.73
Diabetes	0.22	0.19	-0.52	0.18	0.16	0.22	0.18	0.21	**0.24**	0.11	0.23
Ecoli	**0.75**	0.7	0.66	0.72	0.59	0.73	0.75	0.72	0.74	0.74	0.74
Glass	0.4	0.38	0.47	0.45	0.37	0.41	0.48	0.24	0.46	**0.49**	**0.49**
Heart	0.16	0.11	-0.86	0.13	0.27	**0.34**	0.1	0.13	0.13	-0.23	0.17
Hepatitis	0.01	0.03	-0.28	0.04	0.24	**0.25**	0.01	0.02	0	-0.26	-0.02
Housing	0.11	0.13	0	0.14	0.2	**0.22**	0.18	0.16	0.13	0.14	0.16
Imports	0.35	0.28	0.45	0.21	0.31	0.22	0.36	0.38	0.41	**0.53**	0.32
Ionosphere	0.12	0.3	0.28	0.29	0.25	0.32	**0.39**	0.14	0.32	0.31	0.17
Iris	0.81	0.78	**0.88**	0.85	0.8	**0.88**	0.82	0.83	0.84	0.86	0.86
Liver	0.01	0.08	-0.62	0.06	0.05	0.06	0.05	**0.13**	0.07	-0.32	0.03
Satellite	0.73	0.59	-0.21	0.73	0.55	0.71	0.75	0.71	0.67	0.71	**0.77**
Sonar	0.26	0.34	0.38	0.18	0.23	0.21	0.39	0.23	**0.41**	0.39	0.3
Spectf	0.04	0.08	-0.26	0.06	0.05	0.08	0.07	**0.12**	0.11	0.04	0.05
Survival	0.15	0.12	0.04	0.1	0.11	0.13	0.19	0.17	0.17	0.06	**0.25**
Vehicle	0.35	0.37	-0.33	0.35	0.46	**0.62**	0.44	0.35	0.38	0.43	0.47
Vowels	0.87	0.83	**0.94**	0.83	0.84	0.82	**0.94**	0.82	0.86	**0.94**	**0.94**
Waveform	0.56	0.47	0.51	0.61	0.5	0.61	0.56	0.61	0.53	0.53	**0.62**
Wine	0.5	0.52	-0.28	0.56	0.49	**0.8**	0.51	0.52	0.51	0.47	0.53
YeastGalactose	0.96	0.9	0.88	0.94	0.88	0.9	**0.96**	0.44	0.91	0.92	0.95
Average Rank	6.3	7.02	8.8	6.4	7	3.92	4.23	6.33	4.85	6.97	4.18

Table 4.4: Second type of experiments — AdjustedPrec@n

Dataset	GLOSH	KNN L.	LP	ABOD	Auto Enc.	Gaussian	KNN G.	LOCI	LOF	PW	SVDD
Abalone	0.12	0.2	0.22	0.21	0.18	**0.3**	0.23	0.26	0.21	**0.3**	0.16
Aloi	**0.81**	0.74	0.67	0.68	0.77	0.74	**0.81**	0.69	0.8	0.78	**0.81**
Balance-Scale	0.43	0.48	0.54	0.43	0.45	0.44	0.43	0.52	**0.59**	0.54	0.56
CellCycle237	**0.31**	0.18	0.04	0.22	0.24	0.23	0.28	0.17	0.3	0.21	0.27
Glass	0.29	0.28	**0.34**	0.2	0.22	0.13	0.32	0.21	**0.34**	**0.34**	0.17
Iris	0.8	0.75	**0.86**	0.83	0.78	0.58	0.82	0.79	0.76	0.85	**0.86**
Satellite	**0.47**	0.44	-0.21	0.21	0.35	0.31	0.46	0.29	0.45	0.32	0.44
Vehicle	0.25	0.17	-0.33	0.14	0.2	0.25	**0.28**	0.23	0.23	0.16	**0.28**
Vowels	0.58	0.58	**0.77**	0.14	0.19	0.08	**0.77**	0.3	0.61	**0.77**	0.76
Waveform	0.45	0.29	0.38	0.35	0.14	0.34	0.45	0.48	0.36	0.4	**0.53**
Wine	0.33	0.38	-0.51	0.37	0.54	**0.56**	0.37	0.37	0.38	0.26	0.31
YeastGalactose	0.81	0.64	0.82	0.74	0.69	0.64	**0.85**	0.73	0.73	0.84	**0.85**
Average Rank	5.21	7.46	6.58	8.04	7.17	7.33	3.71	6.67	4.67	4.92	4.25

recent outlier detection method — called GLOSH [9] — to the one-class classification problem. We also discussed basic principles of one-class classification that should not be violated when adapting unsupervised outlier detection methods to this task, and how to assure that such principles are not violated.

Chapter 5

Hyperparameter Selection in One-Class Classification

5.1 Introduction

In chapter 4, we evaluated a number of different algorithms for one-class classification in a rigorous experimental setup. While our experimental setup was consistent with previous literature involving one-class classification, there is a major concern when performing one-class classification in real-world situations: performing model selection.

In traditional classification, it is possible to perform model selection and choose hyperparameters using a separate validation set from training and test data or performing cross-validation using training data, in order to produce an estimate of the generalization performance of a model and guide the optimization of hyperparameters. In one-class classification however, we do not have outlier objects with which to evaluate different models. If we were to try to use cross-validation using only inlier objects, we could end up with a model that simply classifies everything as an inlier, achieving perfect accuracy on training data, but likely to do poorly at detecting outliers when deployed.

There has been some work done in exploring hyperparameter tuning in the absence of outliers, these methods fall into the categories of heuristic methods which are designed for one algorithm [17], [20], [33], [57], [60]–[62], or relying on the generation of artificial outliers to use for validation [3], [13], [14], [56], which quickly becomes computationally expensive with increasing dimensionality.

In order to better understand the effects that hyperparameter values can have on an algorithm's accuracy for one-class classification, we can evaluate an algorithm's model on test data after training it on a range of hyperparameter values on the training data. In this way we can analyze the sensitivity of an algorithm to its parameters, which will determine its performance if a way to reliably find "good" parameters is not available. We should be careful to note that a comparison between different one-class classification algorithms is not valid here, as we are intentionally evaluating each algorithm's generalization performance across a range of hyperparameter values. Even though one algorithm's top accuracy score may be better than others', there is no guarantee that the associated hyperparameter values will be selected under normal training and validation circumstances.

5.2 Experimental Setup

Our experimental setup is similar to that used in Chapter 4. For these experiments, we evaluate the five top performing one-class classification algorithms from Chapter 4 on 28 datasets, excluding the Aloi and Delft datasets as they are aggregated datasets and as such, the effects of different hyperparameters on individual datasets might not be reflected in the aggregate results. The methods compared are: Gaussian Data Description, k-Nearest Neighbor Outlier Detection, Local Outlier Factor, GLOSH, and SVDD with a Gaussian kernel.

From each dataset we produce multiple new datasets, each labeling a single class as inliers, with all other classes as outliers, as in the Type I experiments in 4, using the classes noted in Appendix A. Each dataset is split into training and testing subsets containing 80% and 20% of the data respectively. Our training phase differs from the experiments in Chapter 4, as we are not performing cross-validation in order to select the best hyperparameters, instead for each hyperparameter value we train each method on the training data (using only inliers), and evaluate the trained model on the testing subset using the Weighted ROC AUC measure. This process is repeated 30 times to improve

the reliability of our results. The full results for each method and dataset are provided in Appendix B.

Two of the methods used, GLOSH and SVDD, have two parameters available for tuning. For the parameters of GLOSH, we adopted the convention of setting m_{clSize} equal to m_{pts}. For SVDD, we fixed the parameter *fracrej* to 0.1, which is commonly used in practice.

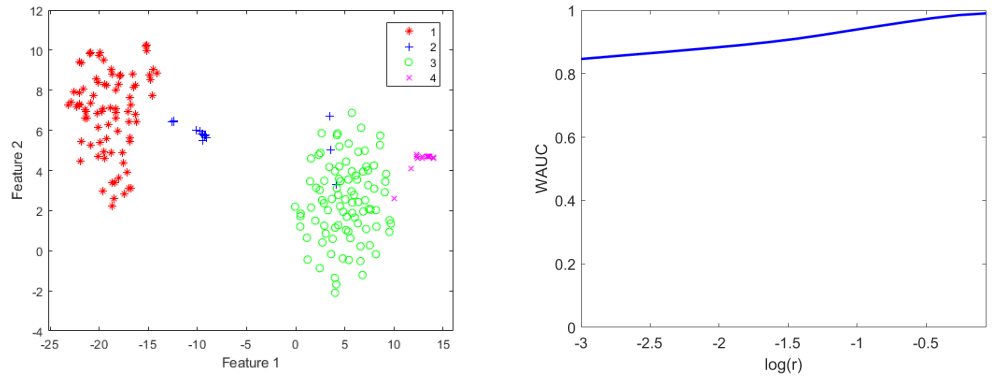
5.3 Results and Discussion

In this section we will analyze the results of the experiments, noting some interesting cases as well as providing insights that can be drawn from them.

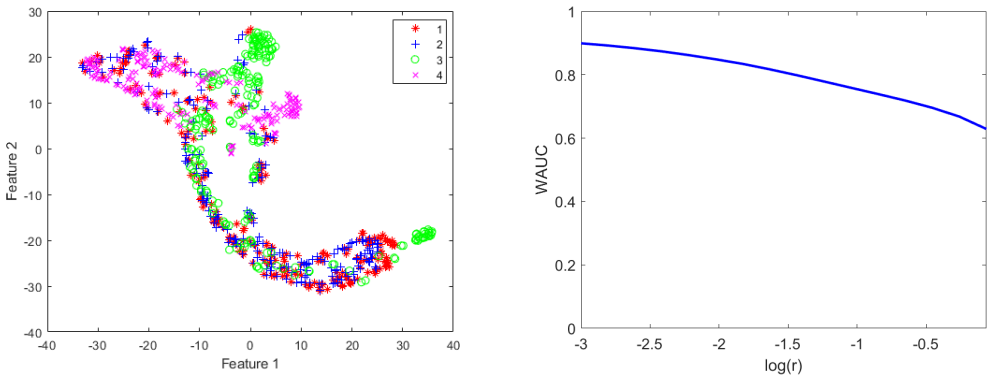
For the Gaussian Data Description, we can see that the results tend to be quite stable, with changes in the regularization parameter, r, not having as large of an effect on its accuracy as with the other methods. In the datasets where we do notice an effect, one of two situations occurs: either the accuracy increases monotonically with larger parameter values up to the maximum value, or decreases monotonically. When $r = 1$, the covariance matrix becomes the identity matrix, and the classifier's output simply depends on an object's distance to the mean of the inlier objects in the training data, regardless of direction. In datasets where the classes are well separated, such as Yeast-Galactose (see Figure 5.1a), this can result in accurate classification results. In datasets where the classes are not well separated, such as Vehicle (Figure 5.1b), retaining the covariances is important to accurately classify objects, and so adding regularization has a negative effect.

Since we only have inlier objects available to us in one-class classification, we cannot predict whether or not the outlier objects will be sufficiently separate from the inlier objects to guide our decision to use covariance regularization, and this can impact the results when we cannot perform cross-validation with outlier objects as we did in Chapter 4.

For the most part, the results for k-Nearest Neighbor Data Description are fairly stable across a range of values for k. Some interesting results include those from the Biomed, Imports, Sonar, and Vowels datasets.



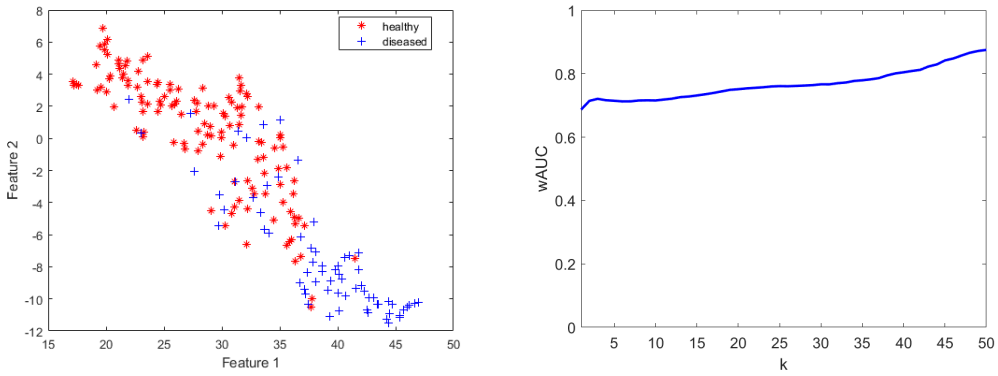
(a) YeastGalactose



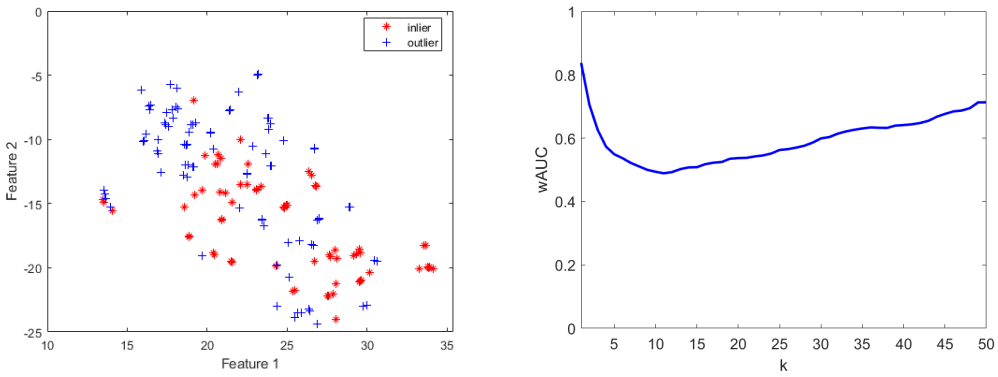
(b) Vehicle

Figure 5.1: t-SNE [39] scatter plots and Gaussian hyperparameter results for the YeastGalactose and Vehicle datasets.

Recall that in k-NN Outlier Detection, an object is classified simply by its distance to its k-th nearest neighbor. In Biomed (Figure 5.2a), it seems that the classes are sufficiently separated such that with larger values of k, it is more likely that for an object of one class, the k-th nearest neighbor in the other class is towards the "back" of the distribution, resulting in a higher distance and better classification. Imports (Figure 5.2b) is an interesting case, as the inlier objects seem to form many small clusters. As a result, when simply looking at the 1-nearest neighbor, we get good results. As we increase k, we lose the benefits of the small clusters of neighbors as the distance to other clusters starts to be considered until k becomes large enough that the more global distances start to improve results similar to Biomed. In Sonar (Figure 5.3a) and Vowels (Figure 5.3b), we see a similar situation to Imports where



(a) Biomed



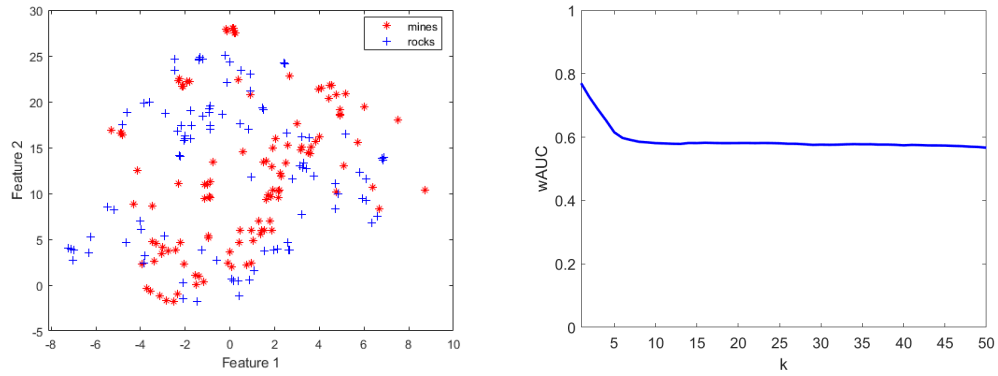
(b) Imports

Figure 5.2: t-SNE scatter plots and kNN hyperparameter results for the Biomed and Imports datasets.

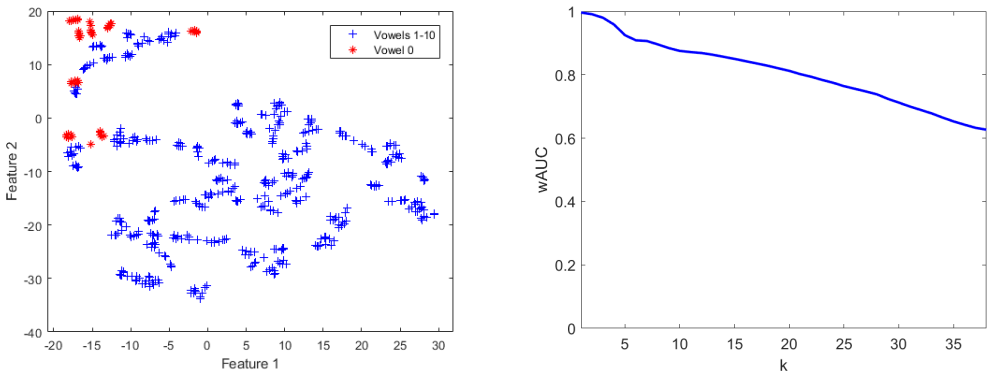
the small clusters lead to good results with very small values of k, however we do not see the same improvement with large values of k as the classes are not sufficiently separate.

In practice we once again cannot predict how close the distribution of outliers will be to the inliers. It would therefore seem that using large values of k for k-NN Outlier Detection would not reliably achieve good results. If, however, we are able to detect small, closely grouped clusters as in Imports, Sonar, and Vowels, then it is possible that using small values of k can achieve good results.

In Local Outlier Factor we see larger variability in the results with changing hyperparameter values compared to Gaussian Data Description and k-NN Outlier Detection. In datasets such as Imports, Sonar, and Vowels (Figure 5.4)



(a) Sonar

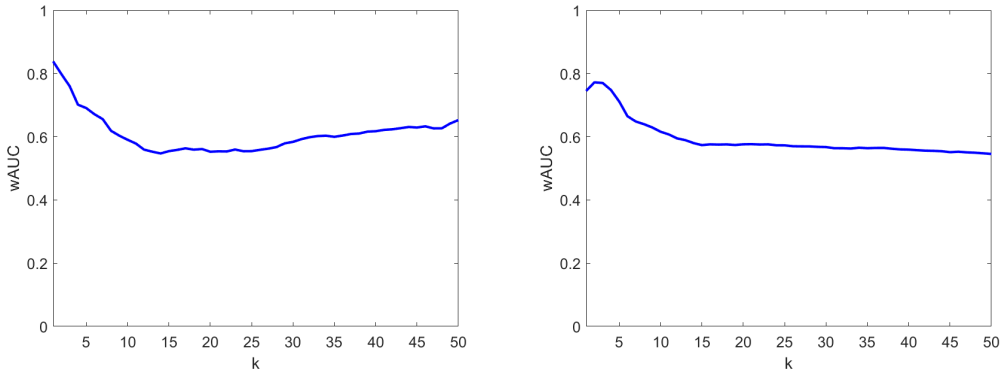


(b) Vowels

Figure 5.3: t-SNE scatter plots and kNN hyperparameter results for the Sonar and Vowels datasets.

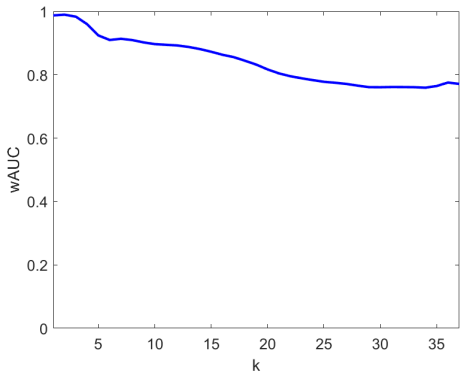
we again see high accuracy with low values of k, reflecting the small clusters from which we can produce good local density estimates, however in other cases it is not so clear that we can derive any information about a good value for k without knowing any information about the outlier distribution.

A pattern emerges with GLOSH in that for most datasets, we see good results for some small value of m_{pts}, though the exact value varies. Once again we see the small clusters present in the Imports, Sonar, and Vowels datasets (Figure 5.5) resulting in good results for small m_{pts} with the advantage quickly disappearing. In other cases, there appears to be a fairly wide range of values for which the performance of GLOSH is stable. The exceptions to this are the Breast (Figure 5.6a) and Spectf (Figure 5.6b) datasets. We also show the performance of GLOSH for each individual class. One possible reason



(a) Imports

(b) Sonar

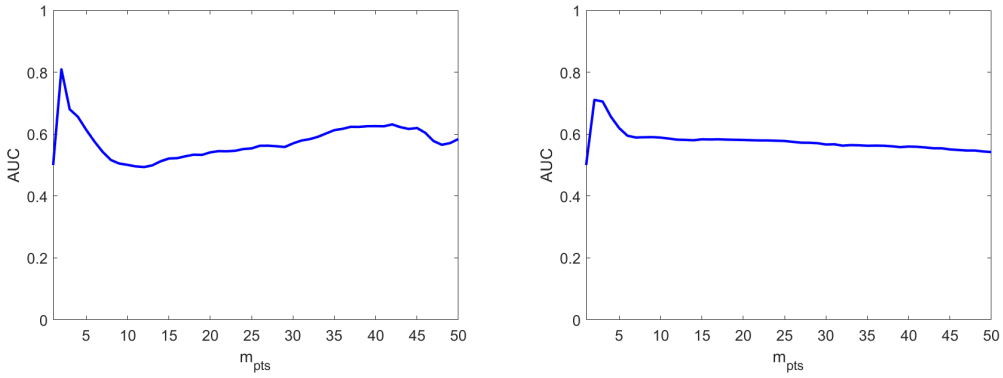


(c) Vowels

Figure 5.4: LOF hyperparameter results for the Imports, Sonar, and Vowels datasets.

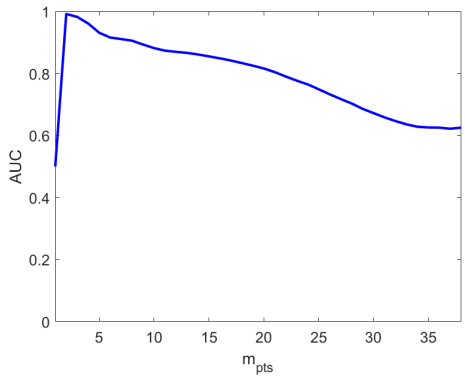
for the poor performance of GLOSH at low values of m_{pts} in Breast is due to the small number of benign objects which overlap the malignant objects. With a low value of m_{pts}, these could be interpreted as a low density cluster by HDBSCAN*, with malignant objects then being assigned to this cluster for GLOSH and having a similar density to the benign objects in the cluster, being classified as inliers. For Spectf, we note this result as a drawback of the Weighted ROC AUC evaluation measure, as the aggregate measure can result in a poor performance for one class masking a good performance with another class, especially in an imbalanced dataset, as we see here. It is possible that identifying values for m_{pts} (and m_{ClSize}) that result in good HDBSCAN* clustering results may also produce good results for GLOSH.

The results for SVDD show that in many cases, the kernel bandwidth



(a) Imports

(b) Sonar



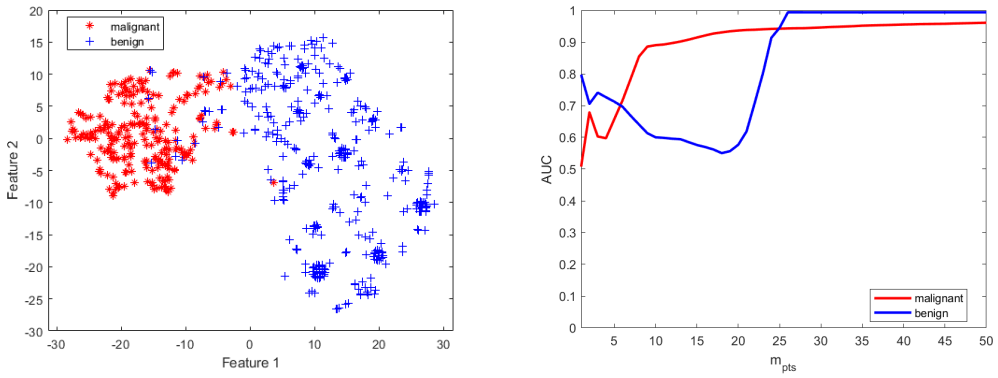
(c) Vowels

Figure 5.5: GLOSH hyperparameter results for the Imports, Sonar, and Vowels datasets.

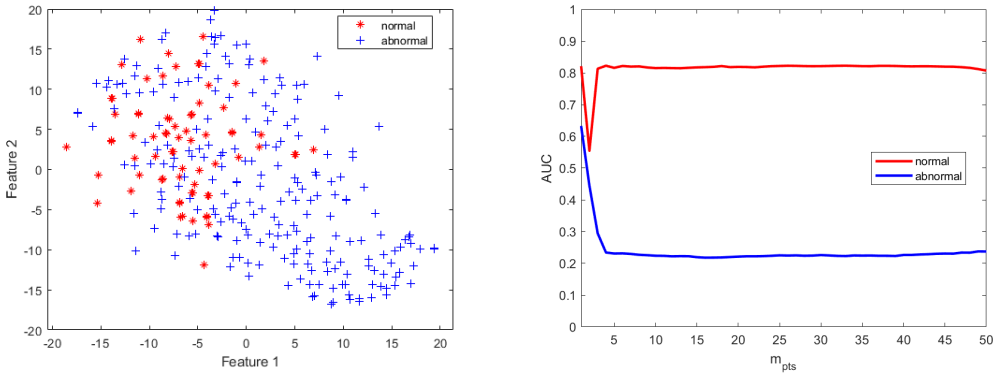
parameter σ is quite sensitive to different values, with a noticeable peak for many datasets (Figure 5.7). The performance of SVDD would then highly depend on the hyperparameter selection method's ability to find these peaks in the absence of outlier data.

5.4 Summary and Conclusion

We analyzed the effect of a range of hyperparameter values on five one-class classification algorithms. From this we found that in the absence of outlier data for training and validation, it can be difficult to reliably select the best hyperparameters. We saw how certain characteristics of the inlier class may be helpful in selecting hyperparameters, such as the presence of clusters which could help the unsupervised methods, though whether or not this is a reliable



(a) Breast

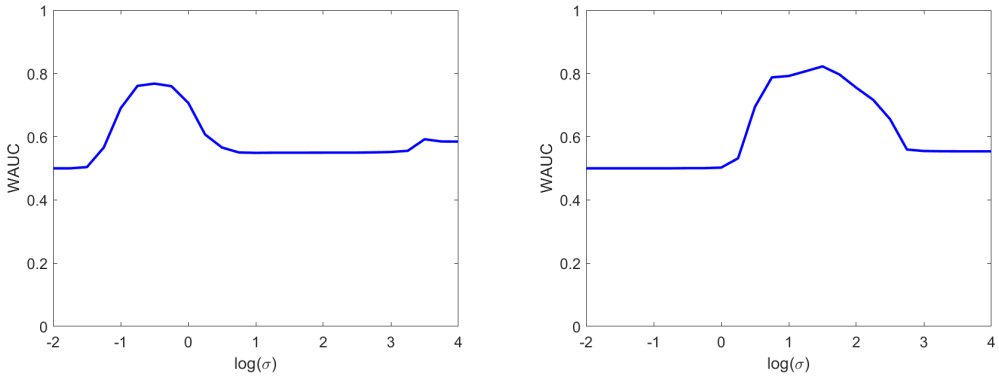


(b) Spectf

Figure 5.6: t-SNE scatter plots and GLOSH hyperparameter results for the Breast and Spectf datasets.

method remains to be seen.

While the SVDD algorithm has a number of hyperparameter selection methods proposed, there is no similar work done for other one-class classification algorithms, nor has there been an extensive comparison of algorithms without including outlier objects for validation. It is possible that internal evaluation methods such as that proposed in [41] could provide a way to select hyperparameters for one-class classification, and for a method such as that proposed in [42] to produce a good clustering result for GLOSH.



(a) Sonar (b) Vehicle

Figure 5.7: SVDD hyperparameter results for the Sonar and Vehicle datasets.

Chapter 6

Combination Methods

6.1 Introduction

Often times when performing one-class classification, it can be beneficial to combine different classifiers in order to improve their robustness and performance [51]. The use of ensemble methods for classification is well studied and many different methods exist to construct an ensemble [49]. Additionally, ensemble methods have been used in outlier detection. [50], [65]. There is also a growing amount of research into ensemble methods for one-class classification. [25], [35], [55].

When looking to build an ensemble of one-class classifiers, some strategies involve combining the classification scores produced by multiple classifiers on some batch set of data. In general, these strategies can be divided into score-based strategies and rank-based strategies. With score-based strategies, the output scores for different classifiers must be combined in some fashion to produce a new output score. One difficulty with this strategy is that the output domains of the different classifiers may vary, for example one might output a probability while another might be a distance measure. In order to combine the classifiers, some normalization procedure must be applied to the scores.

In order to avoid the problem of normalizing output scores, another set of strategies instead relies on combining the rankings of the classifier outputs. Given a list of outputs by a classifier on some data, we can construct a set of rankings such that an object with a rank of j means that it has the j'th small-

est value according to the classifier's outputs, or the object most considered an outlier. In this chapter, we will explore the use of rank-based strategies for building ensembles of one-class classifiers utilizing the implementations provided by [31]. In particular, we utilized the Borda count [6], median [58], Footrule [15], Condorcet [12], Reciprocal Rank Fusion [11], and ULARA [34] methods.

Borda Count

The Borda count [6] is commonly used in elections as a ranking system where voters can rank candidates according to their order of preference. For a given voter's ballot, each candidate is then awarded points according to their ranking in the ballot. The winner is the candidate with the most points summed across all voters. There are multiple versions of the Borda count, however in our experiments we use the method which, given a ranking of n objects from a classifier, awards $n-1$ points to the object ranked 1st, $n-2$ points to the object ranked 2nd, and so on, with the object ranked last receiving 0 points.

Median

With the median combination rule [58], we simply present the median of an object's rankings across all classifiers as its score.

Footrule

The Footrule method [15] seeks to find the optimal solution to the problem of minimizing the sum of distances between the combined ranking and the individual rankings:

$$\text{Footrule}(\mathcal{R}) = \arg\min_{\pi} \left(\sum_{\tau \in \mathcal{R}} d(\tau, \pi) \right), \tag{6.1}$$

where $\mathcal{R}$ is the collection of all the rankings from individual classifiers, τ is the set of rankings from an individual classifier, and π is the set of rankings produced by the combination method. The distance measure we use in our

experiments is the Spearman footrule distance:

$$d(\tau_1, \tau_2) = \sum_{i=1}^{|\tau|} |\tau_1(i) - \tau_2(i)| \tag{6.2}$$

Condorcet

A Condorcet method is any method which satisfies the Condorcet criterion [12] which states that in a ranked voting system, if a candidate would win in all pairwise comparisons with the other candidates that candidate is the Condorcet winner and should be ranked first. We use a method which satisfies the Condorcet criterion by computing a matrix P of pairwise comparisons between each object in a given classifier's ranking, where $P(i, j) = 1$ if the object at index i is ranked higher than the one at index j, and 0 otherwise. We produce one such matrix for each classifier, and sum all the matrices together. We then perform another series of pairwise contests between each object, ranking them according to how many contests they win.

Reciprocal Rank Fusion

Reciprocal Rank Fusion (RRF) [11] is a simple scoring method that is based on the authors' intuition that the effect of lower ranked objects should not vanish as it would were an exponential function used. The score of an object is given by:

$$\text{RRF score}(s) = \sum_{\tau \in \mathcal{R}} \frac{1}{\epsilon + \tau(s)} \tag{6.3}$$

where $\tau(s)$ is the rank of object s from a given classifier and ϵ is a fixed parameter. We use $\epsilon = 60$ as suggested in [11] which mitigates the impact of abnormal classifier rankings.

ULARA

ULARA [34] rewards classifiers that produce different rankings than other classifiers by weighting them according to a measure of inconsistency:

$$\text{Inconsistency}(\tau_{c_i}) = \sum_{j=1}^{|\tau|} (\tau_{c_i}(j) - \mu(j))^2$$

where τ_{c_i} is the set of rankings from a given classifier, μ is the average ranked list among all the classifiers, and $|\tau|$ is the number of ranked objects. We can then assign a weight to each classifier:

$$W(\tau_{c_i}) = \frac{\text{Inconsistency}(\tau_{c_i})}{\sum_{j=1}^{|\tau|} \text{Inconsistency}(\tau_{c_i})}$$

The weights are then used to perform a weighted average of the rankings from each classifier.

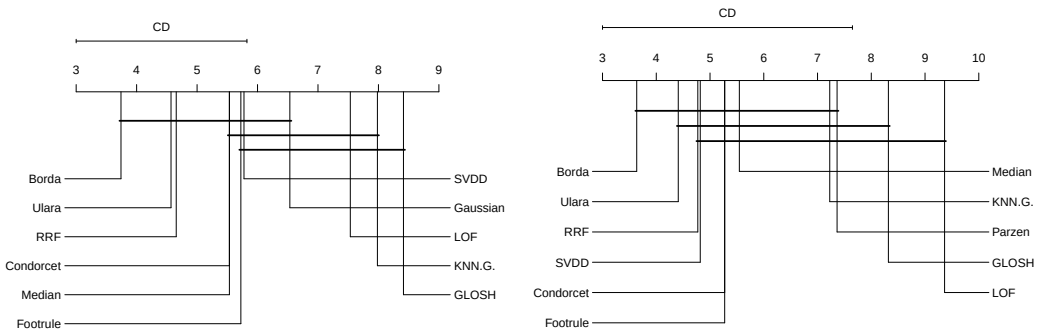
6.2 Experimental Setup

Our experimental setup is as follows: for each experiment type outlined in Chapter 4, we select the five top performing one-class classification algorithms to use for combination (SVDD, Gaussian, kNN_{global}, LOF, and GLOSH for Type I, SVDD, PW, kNN_{global}, LOF, and GLOSH for Type II). The datasets and testing subsets are the same as in Chapter 4, with the exception of the Aloi dataset. From the predictions made by each algorithm on each testing subset of size m, we convert the algorithm's output into a set of rankings from 1 to m, with the object of rank 1 considered the most outlier, and the object of rank m considered the least outlier.

Once we have obtained a set of rankings for each algorithm on a given testing subset, we then apply each combination method using these rankings to obtain the predictions, and evaluate the output using the Weighted ROC AUC measure. This procedure is repeated 30 times as before. Our goal is to investigate the application of rank combination methods to one-class classification, and to determine whether or not they can build a robust classifier.

6.3 Results and Discussion

Tables 6.1 and 6.2 show the weighted ROC AUC scores for the combination methods. Figure 6.1 shows the average rankings of the methods with



(a) Type I (b) Type II

Figure 6.1: Average rankings of the combination methods over Type I and Type II experiments.

width of the the upper bar (CD) indicating the critical distance of the Friedman/Nemenyi statistical test at significance level $\alpha = 0.05$.

For the type I experiments, we can see that all the combination methods do well compared to the individual methods, although the Footrule method performs the worst among the methods, and close to SVDD. The Borda count method frequently performs highest among the combination methods, as well as overall. RRF and Ulara also perform well, though Borda is able to equal or beat their performance on every dataset.

In the type II experiments, we see that SVDD now performs better than the Condorcet, Footrule, and Median methods. The Borda count method once again outperforms all the other methods, achieving the highest performance on most datasets except Abalone, Glass, and Waveform. RRF and Ulara once again are 2nd and 3rd in performance to Borda, though SVDD is not far behind.

From the results, we can see that when most individual methods perform well, the combination methods will also perform well and in many cases exceed any individual method's performance as the diverse strengths of each individual method benefits the combination. In the Spectf dataset, we see the opposite effect in which the diverse weaknesses of each individual method manifests in the combination methods doing very poorly.

Table 6.1: Type I experiments with combination methods — ROC AUC

Dataset	Combination Methods						Single Methods				
	Borda	Condorcet	Footrule	Median	RRF	Ulara	GLOSH	Gaussian	KNN G.	LOF	SVDD
Abalone	0.74	0.74	0.75	0.75	0.73	0.74	0.66	0.74	0.73	0.66	**0.77**
Arrhythmia	0.57	0.57	0.59	0.59	0.57	0.57	0.63	0.55	0.52	0.6	**0.64**
Balance-Scale	**0.96**	**0.96**	0.95	0.95	**0.96**	**0.96**	0.88	0.93	0.87	0.92	0.91
Ball-Bearing	0.99	0.98	0.98	0.98	0.99	0.99	0.98	**1**	0.98	0.99	0.98
Biomed	0.84	0.84	0.84	**0.85**	0.84	0.84	0.84	0.8	0.84	0.71	0.7
Breast	0.97	0.96	0.96	0.96	0.97	0.97	0.96	**0.98**	0.96	0.96	**0.98**
Cancer	0.56	0.56	0.56	0.56	0.56	0.56	0.52	**0.59**	0.54	0.53	0.53
CellCycle237	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**	**0.88**	0.81	0.82	0.84	0.81	0.83
Colon	**0.68**	**0.68**	**0.68**	**0.68**	**0.68**	**0.68**	0.67	0.67	0.66	**0.68**	**0.68**
Delft	**0.98**	0.97	0.97	0.97	0.97	0.97	0.95	0.95	0.93	0.96	0.96
Diabetes	**0.68**	0.67	0.67	0.67	0.67	0.67	0.65	0.64	0.62	0.66	0.65
Ecoli	**0.94**	**0.94**	**0.94**	**0.94**	**0.94**	**0.94**	**0.94**	**0.94**	**0.94**	**0.94**	**0.94**
Glass	0.81	0.8	0.8	0.8	0.8	0.81	0.78	0.78	0.81	0.81	**0.82**
Heart	0.64	0.62	0.62	0.62	0.63	0.63	0.6	**0.73**	0.58	0.59	0.61
Hepatitis	0.61	0.58	0.57	0.57	0.61	0.59	0.56	**0.74**	0.56	0.54	0.57
Housing	0.65	0.63	0.63	0.63	0.64	0.64	0.65	**0.78**	0.69	0.65	0.7
Imports	0.76	0.74	0.73	0.73	0.76	0.75	0.7	0.65	0.71	**0.78**	0.74
Ionosphere	**0.75**	0.74	0.74	0.74	**0.75**	**0.75**	0.74	0.64	0.67	0.64	0.74
Iris	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	0.97	**0.98**	0.97	0.97	**0.98**
Liver	**0.56**	**0.56**	0.55	0.55	**0.56**	**0.56**	0.54	0.54	0.55	0.55	**0.56**
Sat	**0.97**	**0.97**	**0.97**	**0.97**	0.96	**0.97**	0.95	0.94	0.96	0.93	0.96
Sonar	**0.77**	0.76	0.76	0.76	**0.77**	0.76	0.7	0.66	0.76	**0.77**	0.75
Spectf	0.38	0.39	0.39	0.41	0.38	0.38	**0.66**	0.55	0.52	0.63	**0.66**
Survival	**0.65**	**0.65**	**0.65**	**0.65**	**0.65**	**0.65**	0.61	0.58	0.62	0.61	**0.65**
Vehicle	0.85	0.84	0.84	0.84	0.84	0.84	0.75	**0.9**	0.79	0.77	0.82
Vowels	**1**	**1**	**1**	**1**	**1**	**1**	0.99	0.99	**1**	0.99	0.99
Waveform	**0.91**	0.89	0.89	0.89	0.9	0.9	0.89	**0.91**	0.89	0.88	**0.91**
Wine	0.91	0.88	0.88	0.88	0.91	0.89	0.86	**0.96**	0.86	0.86	0.87
YeastGalactose	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	**0.99**	0.98	**0.99**	**0.99**	**0.99**
Avg. rank	**3.78**	5.59	5.91	5.70	4.54	4.56	8.30	6.46	7.85	7.43	5.89

Table 6.2: Type II experiments with combination methods — ROC AUC

Dataset	Combination Methods						Single Methods				
	Borda	Condorcet	Footrule	Median	RRF	Ulara	GLOSH	PW	KNN G.	LOF	SVDD
Abalone	0.7	0.69	0.69	0.69	0.69	0.7	0.62	0.71	0.68	0.63	**0.73**
Balance-Scale	**0.94**	**0.94**	**0.94**	0.93	**0.94**	**0.94**	0.82	0.82	0.83	0.84	0.81
CellCycle237	**0.81**	**0.81**	**0.81**	**0.81**	**0.81**	**0.81**	**0.81**	0.74	0.79	0.75	0.78
Glass	0.74	0.72	0.72	0.72	0.73	0.73	0.75	0.76	0.76	**0.77**	**0.77**
Iris	**0.97**	**0.97**	**0.97**	**0.97**	**0.97**	**0.97**	0.95	**0.97**	0.96	0.93	**0.97**
Satellite	**0.86**	0.85	0.85	0.85	0.84	0.85	0.85	0.82	0.84	0.82	0.85
Vehicle	**0.75**	0.74	0.74	0.74	0.74	**0.75**	0.68	0.72	0.74	0.67	**0.75**
Vowels	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	0.94	**0.98**	**0.98**	0.95	**0.98**
Waveform	0.81	0.81	0.81	0.81	0.82	0.8	0.81	0.77	0.81	0.75	**0.86**
Wine	**0.78**	0.77	0.77	0.77	**0.78**	**0.78**	0.74	0.76	0.75	0.74	0.77
YeastGalactose	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	**0.98**	0.95	0.97	0.97	0.95	0.97
Average Rank	**3.64**	5.27	5.27	5.55	4.77	4.41	8.32	7.36	7.23	9.36	4.82

6.4 Summary and Conclusions

In this chapter we investigated the use of rank-based strategies for combining one-class classifiers. The use of rankings rather than real-valued outputs alleviates the need for normalizing the outputs of different methods.

One potential drawback of rank-based strategies over other strategies is that we discard important information that could be present in scores, for example if two objects have adjacent rankings but a large difference in score, we may want this represented in the combination by a greater amount.

Another drawback is that rank-based strategies do not specifically target the strengths and weaknesses of the individual classifiers and instead rely on the classifiers being sufficiently accurate to achieve good results. In one-class classification, however, because we normally do not have outlier objects with which to validate our models, it is difficult to assess the strengths and weaknesses we could use to build our ensemble.

Nevertheless, we found the Borda method to be a simple and highly effective method for combining multiple one-class classifiers, consistently performing among the best out of all the methods tested.

Chapter 7

Summary and Conclusions

In this thesis, we performed an extensive study on unsupervised outlier detection methods adapted for one-class classification. We proposed a methodological framework for adapting unsupervised outlier detection methods to one-class classification and applied it to GLOSH as an example.

We performed a comparison of one-class classification algorithms and unsupervised outlier detection methods in the one-class classification scenario over a number of datasets in two different experimental settings, as well as introducing a complementary evaluation measure to one-class classification with AdjustedPrec@n. We found that SVDD and kNN$_{global}$ were the top performing methods.

We investigated the hyperparameter selection problem in one-class classification, providing an overview of the performance of hyperparameter settings across a variety of datasets and gaining insight into what characteristics of a dataset can affect the performance of a method. We saw how in the absence of true outliers for performing hyperparameter optimization, it is difficult to determine how well a method can perform in practice even if it does well with good hyperparameters.

Finally, we investigated the use of rank-based combination strategies for combining multiple one-class classifiers and found that they are a simple and fast way to improve the robustness and performance over individual methods.

7.1 Future Work

There are many avenues of research available in the realm of one-class classification and the application of techniques from unsupervised outlier detection and learning. While we performed an extensive comparison of one-class classification algorithms and unsupervised outlier detection methods, there are still more methods which could be compared in a larger study.

In addition, the comparisons performed in this study and in some literature utilize true outliers in order to perform hyperparameter optimization, which violates the principle that in one-class classification we do not have sufficient samples of outliers to perform validation. While there have been methods proposed for optimizing hyperparameters in the absence of outliers, a detailed comparison has not been performed.

The hyperparameter selection problem is one which we believe could see the biggest benefits from research. One possible direction could be the use of generative models (such as Generative Adversarial Networks [21]) for generating artificial outliers which lie close to the boundary of an inlier class for validation.

References

[1] C. C. Aggarwal, *Outlier analysis*. Springer, 2013, ISBN: 978-1-4614-6395-5, 978-1-4614-6396-2. 1

[2] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "OPTICS: ordering points to identify the clustering structure," ACM Press, 1999, pp. 49–60. 19

[3] A. Bánhalmi, A. Kocsor, and R. Busa-Fekete, "Counter-example generation-based one-class classification," in *European Conference on Machine Learning*, Springer, 2007, pp. 543–550. 37

[4] V. Barnett and T. Lewis, *Outliers in statistical data*, 3rd. John Wiley & Sons, 1994. 4

[5] C. M. Bishop, *Neural networks for pattern recognition*. Oxford university press, 1995. 6

[6] J. C. de Borda, "Mémoire sur les élections au scrutin," 1781. 48

[7] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997. 17

[8] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '00, Dallas, Texas, USA: ACM, 2000, pp. 93–104, ISBN: 1-58113-217-4. DOI: 10.1145/342009.335388. 12, 26

[9] R. J. G. B. Campello, D. Moulavi, A. Zimek, and J. Sander, "Hierarchical density estimates for data clustering, visualization, and outlier detection," *ACM Trans. Knowl. Discov. Data*, vol. 10, no. 1, 5:1–5:51, Jul. 2015, ISSN: 1556-4681. DOI: 10.1145/2733381. [Online]. Available: http://doi.acm.org/10.1145/2733381. 12, 19, 21, 27, 36

[10] G. O. Campos, A. Zimek, J. Sander, R. J. G. B. Campello, B. Micenková, E. Schubert, I. Assent, and M. E. Houle, "On the evaluation of unsupervised outlier detection: Measures, datasets, and an empirical study," *Data Mining and Knowledge Discovery*, vol. 30, no. 4, pp. 891–927, 2016, ISSN: 1573-756X. DOI: 10.1007/s10618-015-0444-8. [Online]. Available: http://dx.doi.org/10.1007/s10618-015-0444-8. 27, 30

[11] G. V. Cormack, C. L. Clarke, and S. Buettcher, "Reciprocal rank fusion outperforms condorcet and individual rank learning methods," in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, ACM, 2009, pp. 758–759. 48, 49

[12] N. De Condorcet *et al.*, *Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix*. Cambridge University Press, 2014. 48, 49

[13] H. Deng and R. Xu, "Model selection for anomaly detection in wireless ad hoc networks," in *Computational Intelligence and Data Mining, 2007. CIDM 2007. IEEE Symposium on*, IEEE, 2007, pp. 540–546. 37

[14] C. Désir, S. Bernard, C. Petitjean, and L. Heutte, "One class random forests," *Pattern Recognition*, vol. 46, no. 12, pp. 3490–3506, 2013. 37

[15] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar, "Rank aggregation methods for the web," in *Proceedings of the 10th international conference on World Wide Web*, ACM, 2001, pp. 613–622. 48

[16] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," AAAI Press, 1996, pp. 226–231. 19

[17] P. F. Evangelista, M. J. Embrechts, and B. K. Szymanski, "Some properties of the gaussian kernel for one class learning," in *International Conference on Artificial Neural Networks*, Springer, 2007, pp. 269–278. 37

[18] T. Fawcett, "Roc graphs: Notes and practical considerations for researchers," *Machine learning*, vol. 31, no. 1, pp. 1–38, 2004. 17

[19] J.-M. Geusebroek, G. J. Burghouts, and A. W. Smeulders, "The amsterdam library of object images," *International Journal of Computer Vision*, vol. 61, no. 1, pp. 103–112, 2005, ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000042993.50813.60. [Online]. Available: http://dx.doi.org/10.1023/B:VISI.0000042993.50813.60. 28

[20] Z. Ghafoori, S. Rajasegarar, S. M. Erfani, S. Karunasekera, and C. A. Leckie, "Unsupervised parameter estimation for one-class support vector machines," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, Springer, 2016, pp. 183–195. 37

[21] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680. 55

[22] J. Han, M. Kamber, and J. Pei, *Data mining: Concepts and techniques*, 3rd. Morgan Kaufmann, 2011. 1

[23] J. A. Hartigan, *Clustering algorithms*, 99th. New York, NY, USA: John Wiley & Sons, Inc., 1975, ISBN: 047135645X. 19

[24] D. Hawkins, *Identification of outliers*. Chapman and Hall, 1980. 1, 11

[25] K. Hempstalk, E. Frank, and I. H. Witten, "One-class classification by combining density and class probability estimation," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2008, pp. 505–519. 47

[26] S. Hido, Y. Tsuboi, H. Kashima, M. Sugiyama, and T. Kanamori, "Inlier-based outlier detection via direct density ratio estimation," in *2008 Eighth IEEE International Conference on Data Mining*, Dec. 2008, pp. 223–232. DOI: 10.1109/ICDM.2008.49. 4

[27] D. Horta and R. J. G. B. Campello, "Automatic aspect discrimination in data clustering," *Pattern Recognition*, vol. 45, no. 12, pp. 4370–4388, 2012, ISSN: 0031-3203. DOI: http://dx.doi.org/10.1016/j.patcog.2012.05.011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0031320312002415. 28

[28] J. H. M. Janssens, I. Flesch, and E. O. Postma, "Outlier detection with one-class classifiers from ml and kdd," in *Machine Learning and Applications, 2009. ICMLA '09. International Conference on*, Dec. 2009, pp. 147–153. DOI: 10.1109/ICMLA.2009.16. 5, 10, 26, 27, 29–31, 33

[29] J. H. M. Janssens and E. O. Postma, "One-class classification with LOF and LOCI: An empirical comparison," in *Proceedings of the 18th Annual Belgian-Dutch on Machine Learning*, Tilburg, The Netherlands, May 2009, pp. 56–64. 4, 5, 26

[30] N. Japkowicz, C. Myers, M. Gluck, *et al.*, "A novelty detection approach to classification," in *IJCAI*, 1995, pp. 518–523. 6

[31] P. A. Jaskowiak, D. Moulavi, A. C. Furtado, R. J. Campello, A. Zimek, and J. Sander, "On strategies for building effective ensembles of relative clustering validity criteria," *Knowledge and Information Systems*, vol. 47, no. 2, pp. 329–354, 2016. 48

[32] S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, no. 3, pp. 241–254, 1967, ISSN: 1860-0980. DOI: 10.1007/BF02289588. [Online]. Available: http://dx.doi.org/10.1007/BF02289588. 21

[33] S. Khazai, S. Homayouni, A. Safari, and B. Mojaradi, "Anomaly detection in hyperspectral images based on an adaptive support vector method," *IEEE Geoscience and Remote Sensing Letters*, vol. 8, no. 4, pp. 646–650, 2011. 37

[34] A. Klementiev, D. Roth, and K. Small, "An unsupervised learning algorithm for rank aggregation," in *European Conference on Machine Learning*, Springer, 2007, pp. 616–623. 48, 49

[35] B. Krawczyk and M. Woźniak, "Dynamic classifier selection for one-class classification," *Knowledge-Based Systems*, vol. 107, pp. 43–53, 2016. 47

[36] H.-P. Kriegel, M. Schubert, and A. Zimek, "Angle-based outlier detection in high-dimensional data," in *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '08, Las Vegas, Nevada, USA: ACM, 2008, pp. 444–452, ISBN: 978-1-60558-193-4. DOI: 10.1145/1401890.1401946. 14

[37] L. Lelis and J. Sander, "Semi-supervised density-based clustering," in *2009 Ninth IEEE International Conference on Data Mining*, Dec. 2009, pp. 842–847. DOI: 10.1109/ICDM.2009.143. 20

[38] M. Lichman, *UCI machine learning repository*, 2013. [Online]. Available: http://archive.ics.uci.edu/ml. 28

[39] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008. 40

[40] M. Markou and S. Singh, "Novelty detection: A review–part 1: Statistical approaches," *Signal Processing*, vol. 83, no. 12, pp. 2481–2497, 2003, ISSN: 0165-1684. DOI: http://dx.doi.org/10.1016/j.sigpro.2003.07.018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0165168403002020. 4

[41] H. O. Marques, R. J. Campello, A. Zimek, and J. Sander, "On the internal evaluation of unsupervised outlier detection," in *Proceedings of the 27th International Conference on Scientific and Statistical Database Management*, ACM, 2015, p. 7. 45

[42] D. Moulavi, P. A. Jaskowiak, R. J. Campello, A. Zimek, and J. Sander, "Density-based clustering validation," in *Proceedings of the 2014 SIAM International Conference on Data Mining*, SIAM, 2014, pp. 839–847. 45

[43] S. Papadimitriou, H. Kitagawa, P. Gibbons, and C. Faloutsos, "LOCI: fast outlier detection using the local correlation integral," Bangalore, India, 2003, pp. 315–326. 12–14, 26

[44] E. Parzen, "On estimation of a probability density function and mode," *The annals of mathematical statistics*, vol. 33, no. 3, pp. 1065–1076, 1962. 6, 8, 26

[45] E. Pekalska, D. M. J. Tax, and R. P. W. Duin, "One-class LP classifier for dissimilarity representations," in *Advances in Neural Information Processing Systems*, S. Becker, S. Thrun, and K. Obermayer, Eds., vol. 15, MIT Press: Cambridge, MA, 2003. 6, 9

[46] M. A. F. Pimentel, D. A. Clifton, L. Clifton, and L. Tarassenko, "A review of novelty detection," *Signal Processing*, vol. 99, pp. 215–249, 2014, ISSN: 0165-1684. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S016516841300515X. 1

[47] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," in *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '00, Dallas, Texas, USA: ACM, 2000, pp. 427–438, ISBN: 1-58113-217-4. DOI: 10.1145/342009.335437. 12

[48] D. de Ridder, D. M. Tax, and R. P. W. Duin, "An experimental comparison of one-class classification methods," in *Proc. 4th Annual Conference of the Advanced School for Computing and Imaging (ASCI'98)*, B. Ter Haar Romeny, D. Epema, J. Tonino, and A. Wolters, Eds., ASCI, Delft, The Netherlands: ASCI, 1998, pp. 213–218. 10, 26

[49] L. Rokach, "Ensemble-based classifiers," *Artificial Intelligence Review*, vol. 33, no. 1-2, pp. 1–39, 2010. 47

[50] E. Schubert, R. Wojdanowski, A. Zimek, and H.-P. Kriegel, "On evaluation of outlier rankings and outlier scores," in *Proceedings of the 2012 SIAM International Conference on Data Mining*, SIAM, 2012, pp. 1047–1058. 47

[51] A. J. Sharkey and N. E. Sharkey, *How to improve the reliability of artificial neural networks*. Citeseer, 1995. 47

[52] P.-N. Tan, M. Steinbach, and V. Kumar, *Introduction to data mining*. Addison Wesley, 2006. 1

[53] D. M. J. Tax, *DDtools, the data description toolbox for Matlab*, version 2.1.2, Jun. 2015. 28

[54] D. M. J. Tax and R. P. W. Duin, "Support vector data description," *Machine learning*, vol. 54, no. 1, pp. 45–66, 2004. 6, 7, 26

[55] D. M. J. Tax, *One-class classification*. TU Delft, Delft University of Technology, 2001. 5, 7, 8, 10, 47

[56] D. M. Tax and R. P. Duin, "Uniform object generation for optimizing one-class classifiers," *Journal of machine learning research*, vol. 2, no. Dec, pp. 155–173, 2001. 37

[57] D. M. Tax and K.-R. Muller, "A consistency-based model selection for one-class classification," in *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*, IEEE, vol. 3, 2004, pp. 363–366. 37

[58] L. Vendramin, P. A. Jaskowiak, and R. J. Campello, "On the combination of relative clustering validity criteria," in *Proceedings of the 25th International Conference on Scientific and Statistical Database Management*, ACM, 2013, p. 4. 48

[59] V. N. Vladimir and V. Vapnik, *The nature of statistical learning theory*, 1995. 7

[60] S. Wang, J. Yu, E. Lapira, and J. Lee, "A modified support vector data description based novelty detection approach for machinery components," *Applied Soft Computing*, vol. 13, no. 2, pp. 1193–1205, 2013. 37

[61] Y. Xiao, H. Wang, and W. Xu, "Parameter selection of gaussian kernel for one-class svm," *IEEE transactions on cybernetics*, vol. 45, no. 5, pp. 941–953, 2015. 37

[62] Y. Xiao, H. Wang, L. Zhang, and W. Xu, "Two methods of selecting gaussian kernel parameters for one-class svm and their application to fault detection," *Knowledge-Based Systems*, vol. 59, pp. 75–84, 2014. 37

[63] K. Y. Yeung, C. Fraley, A. Murua, A. E. Raftery, and W. L. Ruzzo, "Model-based clustering and data transformations for gene expression data," *Bioinformatics*, vol. 17, no. 10, pp. 977–987, 2001. 28

[64] K. Y. Yeung, M. Medvedovic, and R. E. Bumgarner, "Clustering gene-expression data with repeated measurements," *Genome Biology*, vol. 4, no. 5, pp. 1–17, 2003, ISSN: 1474-760X. DOI: 10.1186/gb-2003-4-5-r34. [Online]. Available: http://dx.doi.org/10.1186/gb-2003-4-5-r34. 28

[65] A. Zimek, R. J. Campello, and J. Sander, "Ensembles for unsupervised outlier detection: Challenges and research questions a position paper," *ACM Sigkdd Explorations Newsletter*, vol. 15, no. 1, pp. 11–22, 2014. 47

Appendix A

Datasets

Table A.1 contains a list of all the Datasets used in this thesis, including the number of objects in the dataset, dimensionality, number of classes, number of objects in each class, as well as which classes were used as inliers to produce one-class datasets.

Table A.1: Datasets Used

Dataset	# of objects	# of features	# of classes	Class sizes	Classes used
Abalone	4177	10	3	1407, 1323, 1447	all
Aloi[a]	50 - 125 X 400	6	2-5	25 X 2-5	all
Arrhythmia	420	278	2	237, 183	all
Balance-Scale	625	4	3	288, 49, 288	all
Ball-Bearing	4150	32	2	913, 3237	1
Biomed	194	5	2	127, 67	all
Breast	699	9	2	241, 458	all
Cancer	198	33	2	151, 47	all
CellCycle237	237	17	4	49, 31, 18, 139	all
Colon	62	1908	2	22, 40	all
Delft[b]	3300	64	2	817, 2483	1
Diabetes	768	8	2	500, 268	all
Ecoli	336	7	2	52, 284	1
Glass	214	9	6	70, 76, 17, 13, 9, 29	1,2,3,4,6
Heart	297	13	2	137, 160	all
Hepatitis	155	19	2	123, 32	1
Housing	506	13	2	458, 48	all
Imports	159	25	2	71, 88	1
Ionosphere	351	32	2	225, 126	all
Iris	150	4	3	50, 50, 50	all
Liver	345	6	2	145, 200	all
Satellite	4435	36	6	1072, 479, 961, 415, 470, 1038	all
Sonar	208	60	2	111, 97	all
Spectf	349	44	2	95, 254	all
Survival	306	3	2	225, 81	all
Vehicle	846	18	4	212, 217, 218, 199	all
Vowels	528	10	11	48 X 11	all
Waveform	5000	21	3	1657, 1647, 1696	all
Wine	178	13	3	59, 71, 48	all
YeastGalactose	205	80	4	83, 15, 93, 14	all

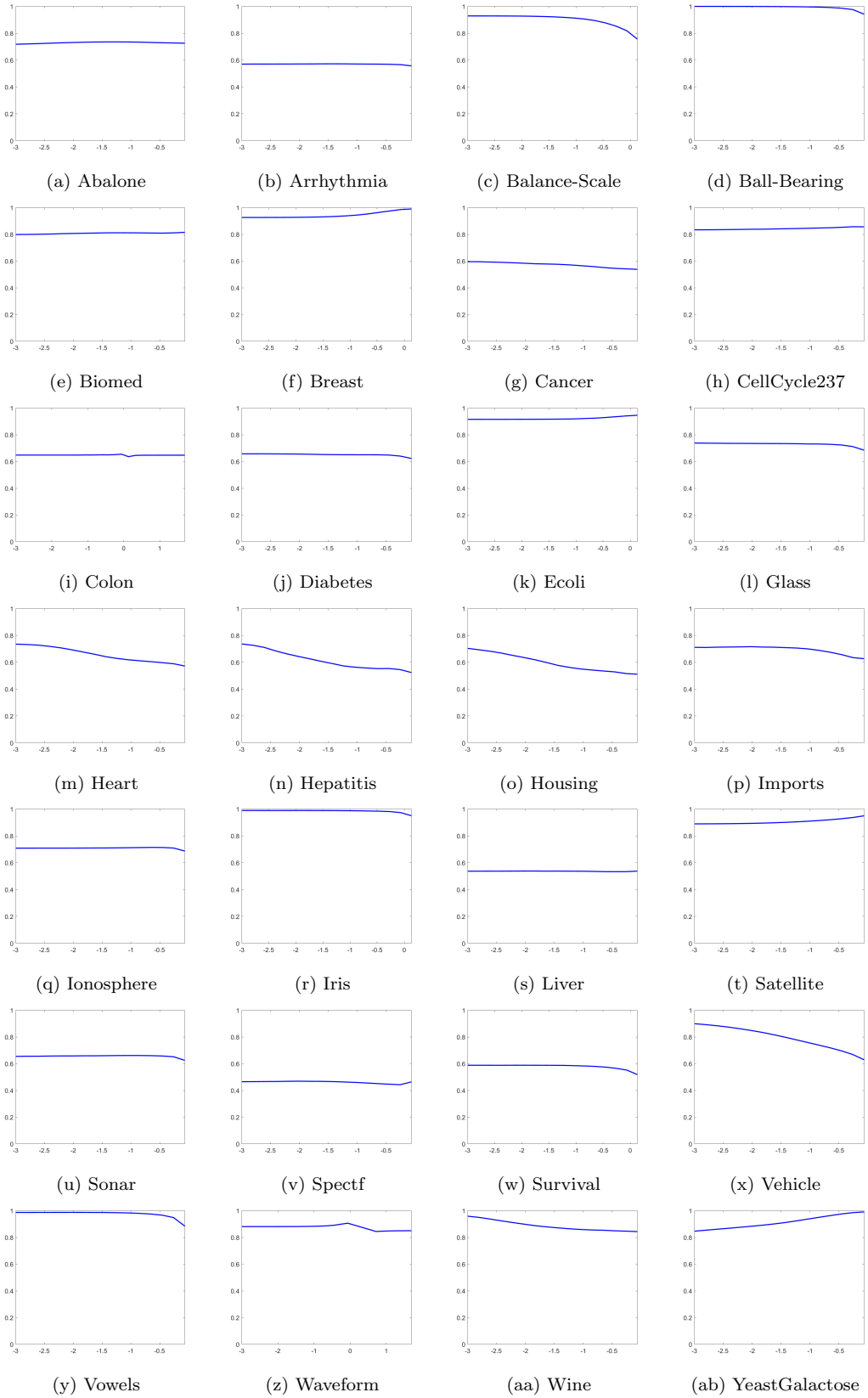
[a]The Aloi dataset is an aggregation of of 400 individual datasets, as detailed in Section 4.2.

[b]The Delft dataset is an aggregation of 5 individual datasets: Delft5x3, Delft5x1, Delft3x2, Delft2x2, and Delft1x3.

Appendix B

Hyperparameter Selection

Figures B.1, B.2, B.3, B.4, and B.5 show the weighted ROC AUC (y-axis) obtained on testing data by varying hyperparameters (x-axis).



(a) Abalone (b) Arrhythmia (c) Balance-Scale (d) Ball-Bearing

(e) Biomed (f) Breast (g) Cancer (h) CellCycle237

(i) Colon (j) Diabetes (k) Ecoli (l) Glass

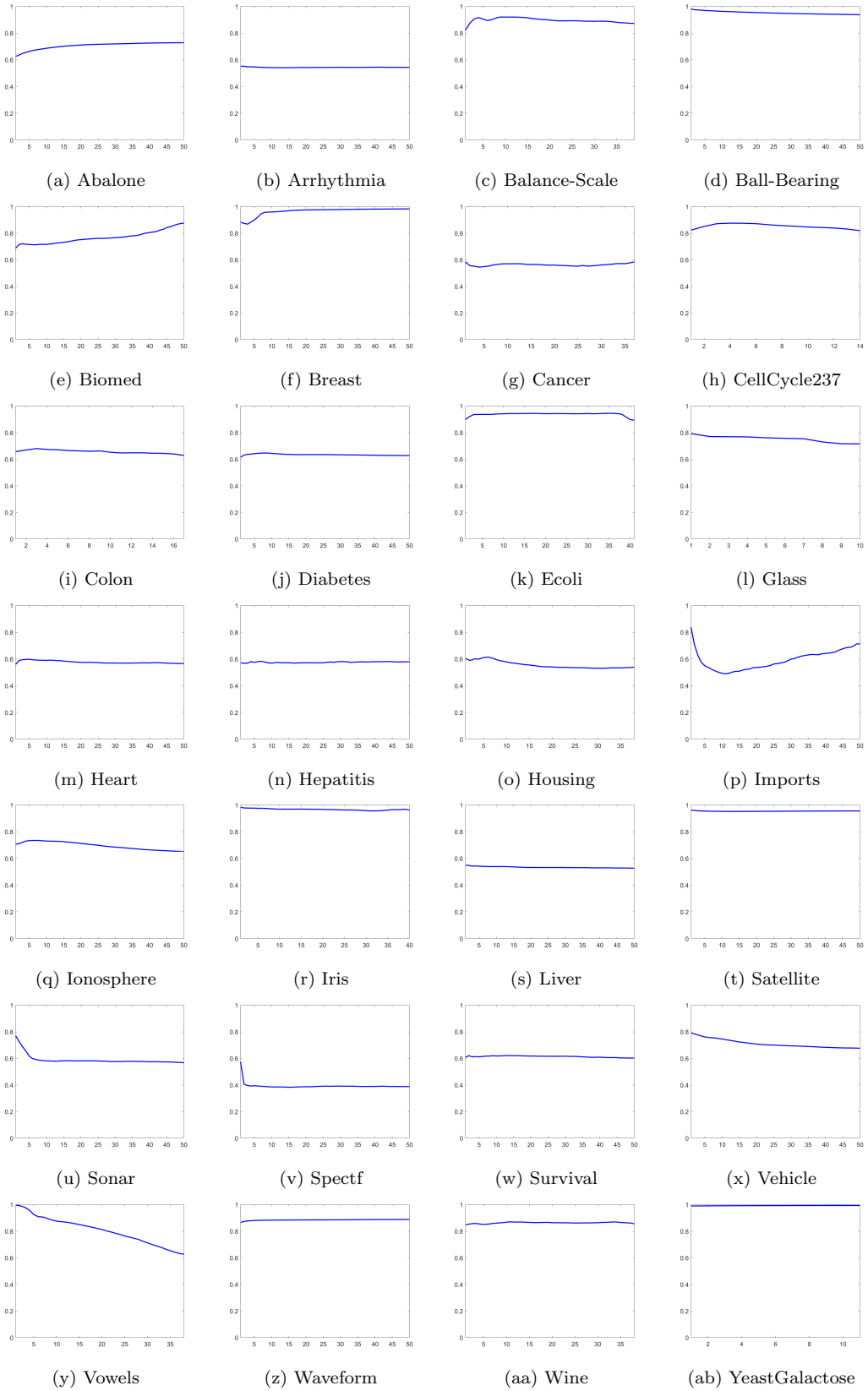
(m) Heart (n) Hepatitis (o) Housing (p) Imports

(q) Ionosphere (r) Iris (s) Liver (t) Satellite

(u) Sonar (v) Spectf (w) Survival (x) Vehicle

(y) Vowels (z) Waveform (aa) Wine (ab) YeastGalactose

Figure B.1: Gaussian Data Description ROC AUC, parameter 10^r



(a) Abalone (b) Arrhythmia (c) Balance-Scale (d) Ball-Bearing

(e) Biomed (f) Breast (g) Cancer (h) CellCycle237

(i) Colon (j) Diabetes (k) Ecoli (l) Glass

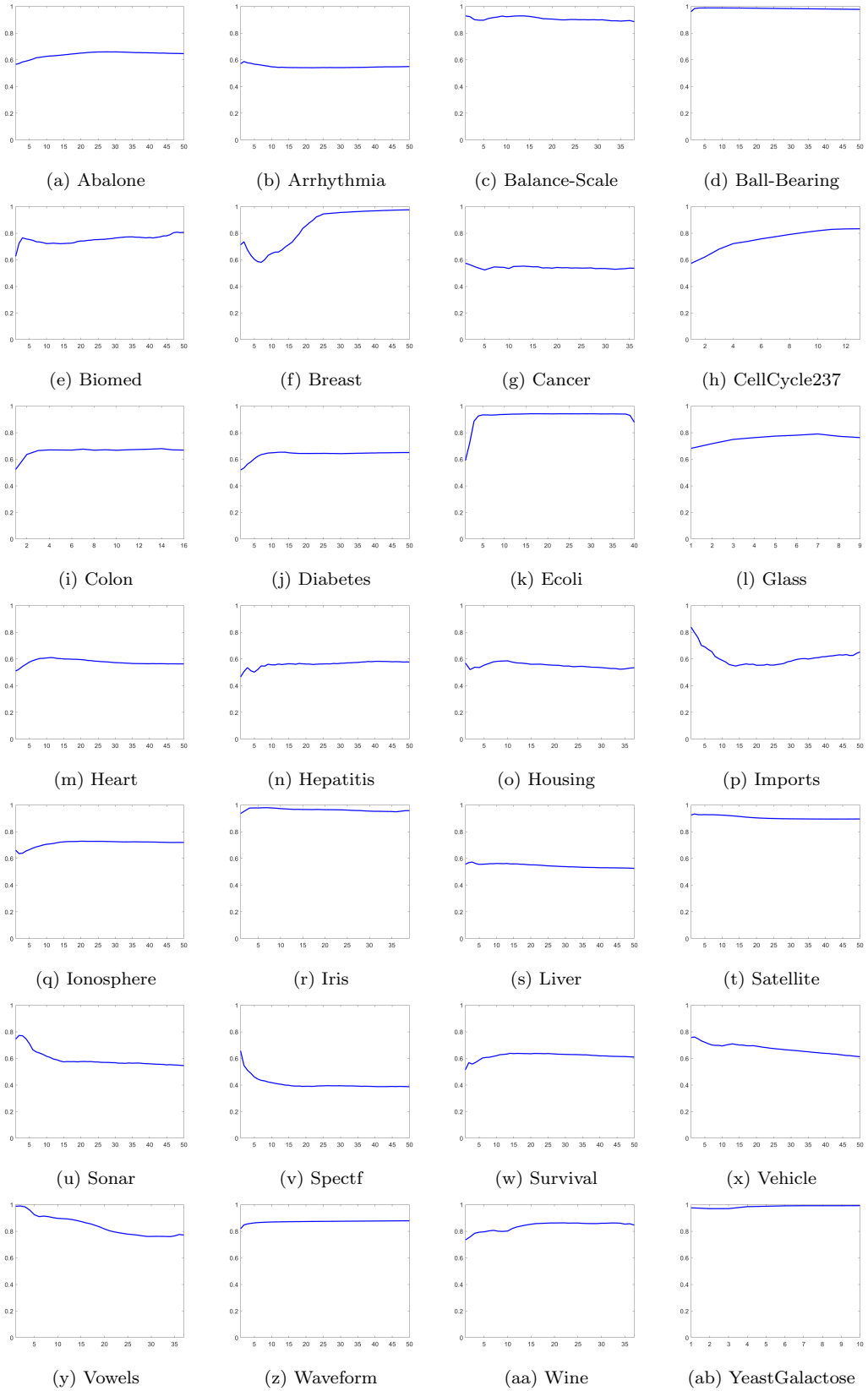
(m) Heart (n) Hepatitis (o) Housing (p) Imports

(q) Ionosphere (r) Iris (s) Liver (t) Satellite

(u) Sonar (v) Spectf (w) Survival (x) Vehicle

(y) Vowels (z) Waveform (aa) Wine (ab) YeastGalactose

Figure B.2: k-Nearest Neighbor Outlier Detection ROC AUC, parameter k



(a) Abalone (b) Arrhythmia (c) Balance-Scale (d) Ball-Bearing

(e) Biomed (f) Breast (g) Cancer (h) CellCycle237

(i) Colon (j) Diabetes (k) Ecoli (l) Glass

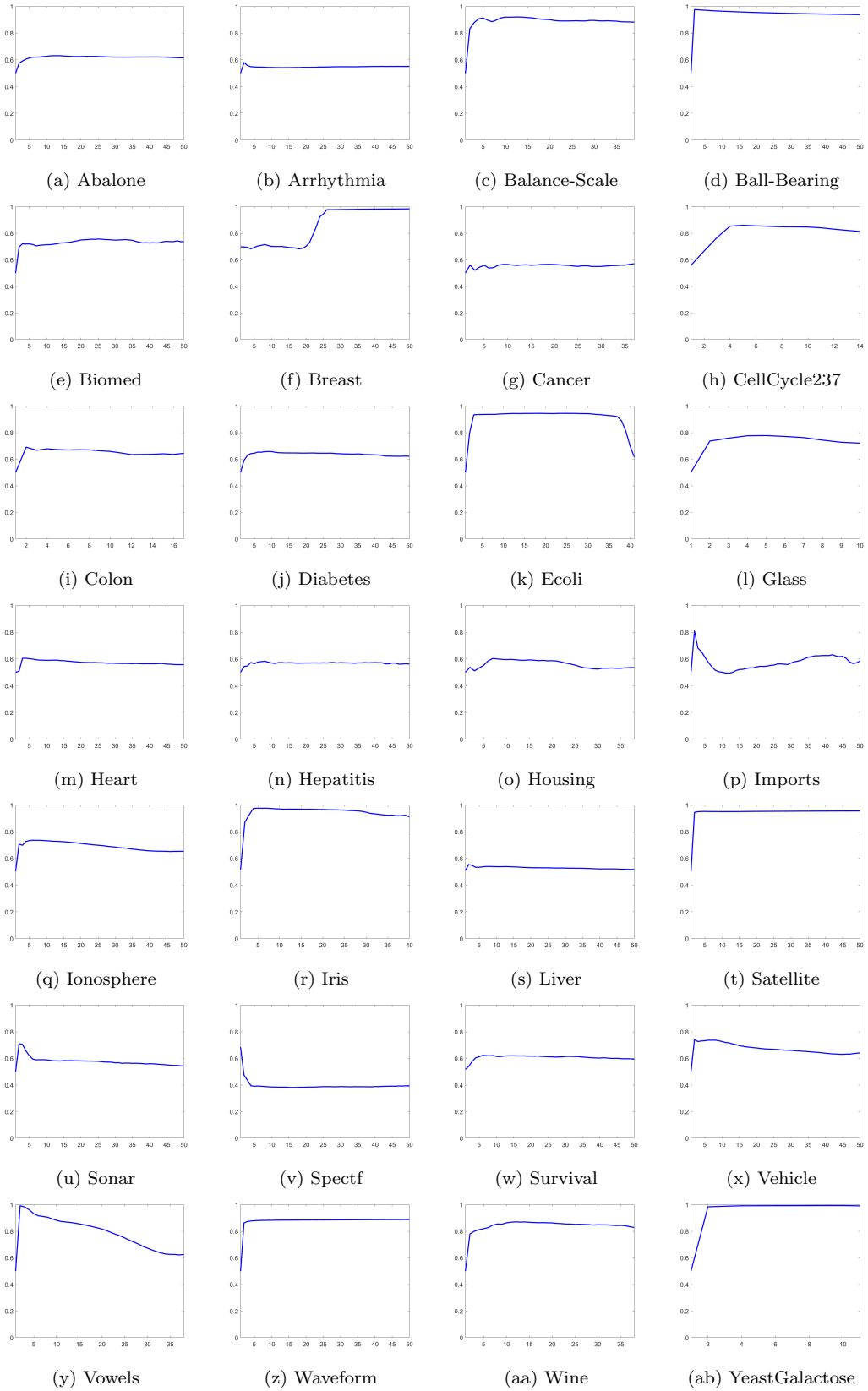
(m) Heart (n) Hepatitis (o) Housing (p) Imports

(q) Ionosphere (r) Iris (s) Liver (t) Satellite

(u) Sonar (v) Spectf (w) Survival (x) Vehicle

(y) Vowels (z) Waveform (aa) Wine (ab) YeastGalactose

Figure B.3: Local Outlier Factor ROC AUC, parameter k



(a) Abalone (b) Arrhythmia (c) Balance-Scale (d) Ball-Bearing

(e) Biomed (f) Breast (g) Cancer (h) CellCycle237

(i) Colon (j) Diabetes (k) Ecoli (l) Glass

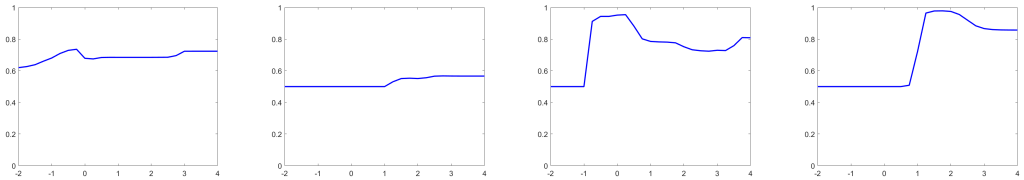
(m) Heart (n) Hepatitis (o) Housing (p) Imports

(q) Ionosphere (r) Iris (s) Liver (t) Satellite

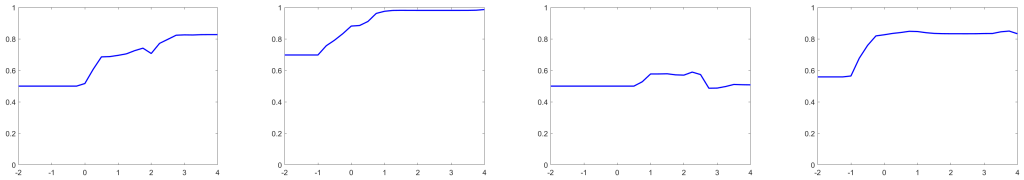
(u) Sonar (v) Spectf (w) Survival (x) Vehicle

(y) Vowels (z) Waveform (aa) Wine (ab) YeastGalactose

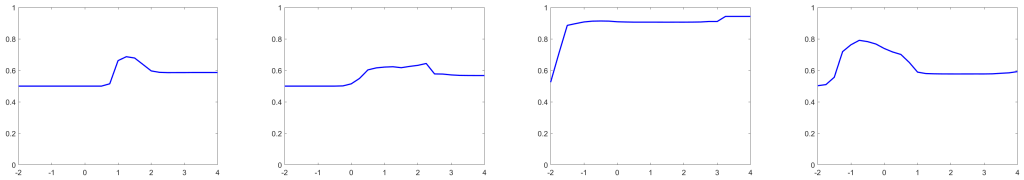
Figure B.4: GLOSH ROC AUC, parameter m_{pts}



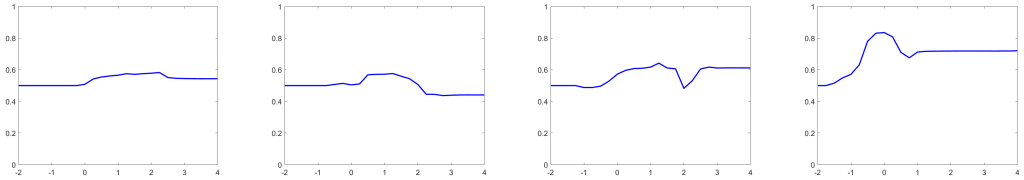
(a) Abalone (b) Arrhythmia (c) Balance-Scale (d) Ball-Bearing



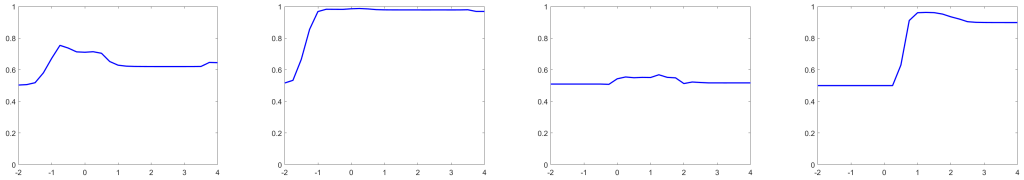
(e) Biomed (f) Breast (g) Cancer (h) CellCycle237



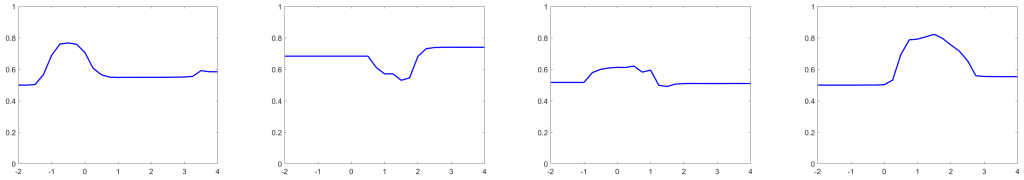
(i) Colon (j) Diabetes (k) Ecoli (l) Glass



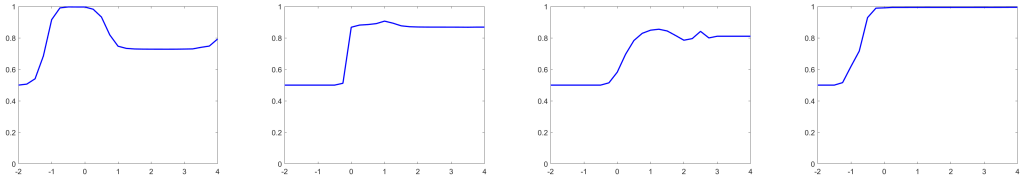
(m) Heart (n) Hepatitis (o) Housing (p) Imports



(q) Ionosphere (r) Iris (s) Liver (t) Satellite



(u) Sonar (v) Spectf (w) Survival (x) Vehicle



(y) Vowels (z) Waveform (aa) Wine (ab) YeastGalactose

Figure B.5: SVDD ROC AUC, $fracrej = 0.1$, parameter 10^{σ}