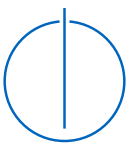
DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Outlier Detection Techniques using Sparse Grids Density Estimation

Tobias Bernecker



DEPARTMENT OF INFORMATICS

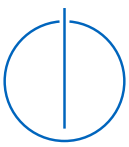
TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Outlier Detection Techniques using Sparse Grids Density Estimation

Outlier Erkennung mit der Sparse Grid Density Estimation

Author:	Tobias Bernecker
Supervisor:	Prof. Dr. rer. nat. habil. Hans-Joachim Bungartz
Advisor:	M.Sc. (hons) Paul-Cristian Sarbu
Submission Date:	September 15, 2018



I confirm that this bachelor's thesis in informatics is my own work and I have documented all sources and material used.

Munich, September 15, 2018 Tobias Bernecker

Abstract

Outlier detection helps to improve results of a clustering process by identifying noisy, anomalous data points in a dataset. However, lots of techniques for outlier detection require a density estimation of the data points, which cannot be computed exactly. To deal with this problem, spatially adaptive sparse grids can be used to learn and approximate the underlying density function of a multi-dimensional dataset. After this learning process, also known as sparse grids density estimation, the obtained approximated function can be evaluated at every data point to receive a corresponding density value. In this thesis, several outlier detection techniques including the Local Outlier Factor and the Local Density Factor are presented. Furthermore, a new density-based approach to obtain a factor for the outlierness of a data point is introduced. The purpose of this thesis is to assess whether outlier detection is a suitable field of application for sparse grids density estimation. To this end, this approach is integrated into an outlier detection framework allowing comparison to other known methods. To validate the results of the presented outlier detection techniques, several artificial datasets with a certain percentage of outliers are tested. Additionally, real datasets are used for further expirements and analysis of the studied detection methods.

Contents

1 Introduction

Clustering is an unsupervised machine learning task that tries to put unlabeled data points with similar properties into the same cluster and data points with different features in distinct clusters. If (single) data points do not belong to any of the clusters, they can be considered outliers. The detection and removal of outliers in a dataset can be very useful because outliers can be seen as noise, e.g. in a dataset containing measurements, which distorts the actual data.

In 1996, a clustering algoritm that separates a dataset into different clusters including a noise cluster, called Density-Based Spatial Clustering of Applications with Noise (DBSCAN), was introduced by Ester, Kriegel, Sander, and X. Xu [7]. Another approach of Rousseeuw [16] focuses on the quality of an already performed clustering. In their method, they define a coefficient, the so-called Silhouette Coefficient, that gives information about how good the assignment of a data point to a cluster is or if the data point should be assigned to another cluster to improve the clustering.

Breunig, Kriegel, Ng, and Sander [4] came up with a new approach which not focusses on deciding whether a data point is an outlier or not. Instead, their method called Local Outlier Factor (LOF) assigns a value representing a degree of outlyingness to every data point. Based on the LOF technique, Latecki, Lazarevic, and Pokrajac [11] came up with the Local Density Factor (LDF) method that is based on a kernel density estimate (KDE). It also uses the approach of LOF to only consider the local neighborhood of a data point to compute its LDF. Similar to LOF, the LDF value gives a measure for the degree of outlyingness of a data point.

In high-dimensional data space, detecting outliers becomes more difficult because of the sparsity of the data [1]. Aggarwal and Yu [1] focus on the projections of a dataset to detect outliers even in high-dimensional data space. Furthermore, Lu, Chen, and Kou [12] propose algorithms that are able to detect sparse outliers accurately compared to other methods by increasing the detection rate and decreasing the false alarm rate, two measures that are defined in Latecki et al. [11].

In their survey, Hodge and Austin [9] compare different outlier detection techniques. The discussed methods are proximity-based techniques and parametric, non-parametric and semi-parametric approaches. Furthermore, Hodge et al. take a look at machine learning and hybrid techniques. They conclude that the choice of an outlier detection technique depends on the dataset and that there is no single universal method for all

datasets.

Pflüger [15] worked on spatially adaptive sparse grids for high-dimensional problems. On this basis, Peherstorfer [14] used techniques of Pflüger to perform multi-dimensional density estimation based on sparse grids (SGDE).

Inspired by the mentioned approaches, the idea of combining the previous work of SGDE and existing outlier detection algorithms was born. This thesis focuses on density-based outlier detection techniques computed with the densities returned by SGDE for a given dataset. Additionally, this thesis presents a new approach for density-based outlier detection in combination with SGDE. Furthermore, these methods are evaluated and compared to non-density-based techniques to evaluate the quality of the results and to conclude whether it is worthwhile to focus on outlier detection based on SGDE in future work.

The rest of this thesis has the following structure. Chapter 2 gives an overview about sparse grids and sparse grids density estimation. Additionally, it provides background information about clustering, it presents a density-based clustering method and it gives an introduction to outlier detection techniques and measures. Chapter 3 describes the workflow of the outlier detection process. Therefore, it explains different techniques and measures that were implemented during the thesis and it presents the new approach for outlier detection. In chapter 4, the introduced outlier detection techniques and the new approach are evaluated with the presented outlier detection measures. Finally, conclusions are drawn in chapter 5.

2 Background

2.1 Sparse Grids

According to Pflüger [15], sparse grids are numerical techniques to deal with high-dimensional problems, e.g. interpolation. Therefore, sparse grids use hierarchical basis functions for a decomposition of the underlying approximation spaces.

In the following, d-linear interpolation for a function $f : \Omega \to \mathbb{R}$, which may only be evaluated at certain points, is explained. The presented interpolation process operates on full grids.

First, for simplicity, Ω is restricted to $\Omega := [0,1]^d$, which is the d-dimensional unit-hypercube, because every d-dimensional rectangular volume can easily be linearly transformed to $[0,1]^d$. Afterwards, Ω is discretized using a regular mesh. This grid consists of equidistant points x_i and mesh width $h_n := 2^{-n}$ for every dimension of f, where n is the refinement level.

Starting with the interpolation of a one-dimensional function f, the standard hat function

$$\varphi(x) = \max\{1 - |x|, 0\}$$

can be extended to the one-dimensional hat basis function,

$$\varphi(x)_{l,i} := \varphi(2^l x - i),$$

where l is a level and i is some index, $0 < i < 2^l$. These one-dimensional hat basis functions are centered at the sampling points of f, $x_{l,i} = 2^{-l} i$, and can be used to interpolate f. As a restriction, f needs to be zero on the boundaries $\partial\Omega$ of the domain Ω. The hat basis functions are a simple form of basis functions and can be replaced with piecewise d-polynomial or B-spline basis functions or others [15] whilst taking into account boundary behaviour of f. The interpolation $u(x)$ of $f(x)$ is part of the space of piecewise linear functions V_n, $u(x) \in V_n$, which is constructed by a sum of hierarchical subspaces W_l,

$$V_n = \bigoplus_{l \leq n} W_l.$$

These hierarchical subspaces are spanned the following way,

$$W_l := \operatorname{span}\{\varphi_{l,i}(x) : i \in I_l\},$$

using the hierarchical index set

$$I_l := \{i \in \mathbb{N} : 1 \leq i \leq 2^l - 1, i \text{ odd}\}.$$

Figure 2.1 shows the hierarchical subspaces W_l up to $l = 3$ and the corresponding spaces of piecewise linear basis functions V_n up to $n = 3$.

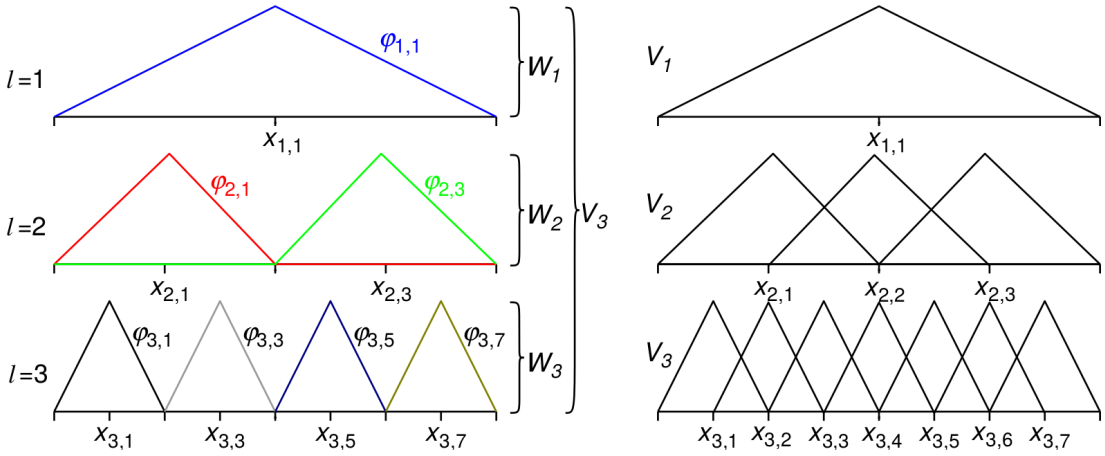


Figure 2.1: One-dimensional hierarchical subspaces W_l with basis functions $\varphi_{l,i}$ up to level $n = 3$ (left) and corresponding spaces V_n in nodal point basis (right). [15]

The interpolation $u(x)$ can now be defined as

$$f(x) \approx u(x) := \sum_{l \leq n, i \in I_l} \alpha_{l,i} \varphi_{l,i}(x),$$

where $\alpha_{l,i}$ are (hierarchical) surplusses for the corresponding basis functions. Figure 2.2 shows an interpolant $u(x)$ constructed with weighted hierarchical basis functions.

This interpolation process can be extended to d dimensions. Therefore, the basis functions are extended using a tensor product,

$$\varphi_{\vec{l},\vec{i}}(\vec{x}) := \prod_{j=1}^{d} \varphi_{l_j,i_j}(x_j),$$

where $\vec{l}$ and $\vec{i}$ are d-dimensional multi-indices. In the d-dimensional case, V_n is the space of piecewise d-linear functions,

$$V_n = \bigoplus_{|\vec{l}|_\infty \leq n} W_{\vec{l}},$$

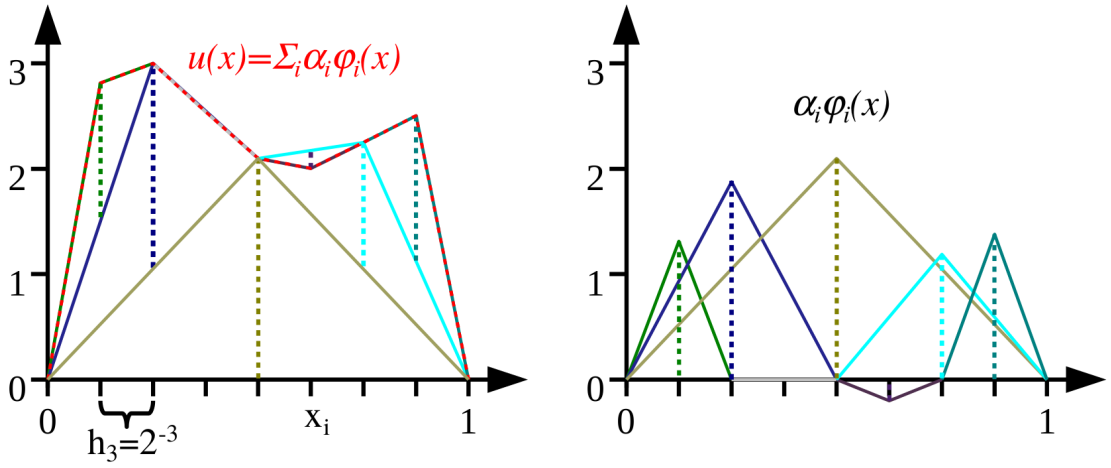


Figure 2.2: One-dimensional piecewise linear interpolation (left) with weighted hierarchical basis functions (right). [15]

and the interpolant $u(\vec{x}) \in V_n$ can be written with the following finite sum,

$$u(\vec{x}) = \sum_{|\vec{l}|_\infty \leq n, \vec{i} \in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \varphi_{\vec{l},\vec{i}}(\vec{x}),$$

where $|\vec{l}|_\infty$ is the maximum-norm of $\vec{l}$,

$$|\vec{l}|_\infty = \max_{1 \leq j \leq d} |l_j|.$$

It is neither necessary nor computationally efficient to use full grids for every problem. Therefore, sparse grids can be applied to these multi-dimensional problems. Sparse grids are able to represent any multi-dimensional function f. Hence, the sparse grid space is constructed with

$$V_n^{(1)} = \bigoplus_{|\vec{l}|_1 \leq n+d-1} W_{\vec{l}},$$

where $|\vec{l}|_1$ is the l_1-norm,

$$|\vec{l}|_1 = \sum_{j=1}^{d} l_j.$$

To obtain the sparse grid space, subspaces from the full grid space V_n, which contain lots of basis functions of small support, are left out. The resulting sparse grid interpolant $u(\vec{x}) \in V_n^{(1)}$ is given by

$$u(\vec{x}) = \sum_{|\vec{l}|_1 \leq n+d-1, \vec{i} \in I_{\vec{l}}} \alpha_{\vec{l},\vec{i}} \varphi_{\vec{l},\vec{i}}(\vec{x}).$$

Figure 2.3 shows the two-dimensional subspaces $W_{\vec{l}}$ up to $l = 3$ and a two-dimensional sparse grid of level $n = 3$.

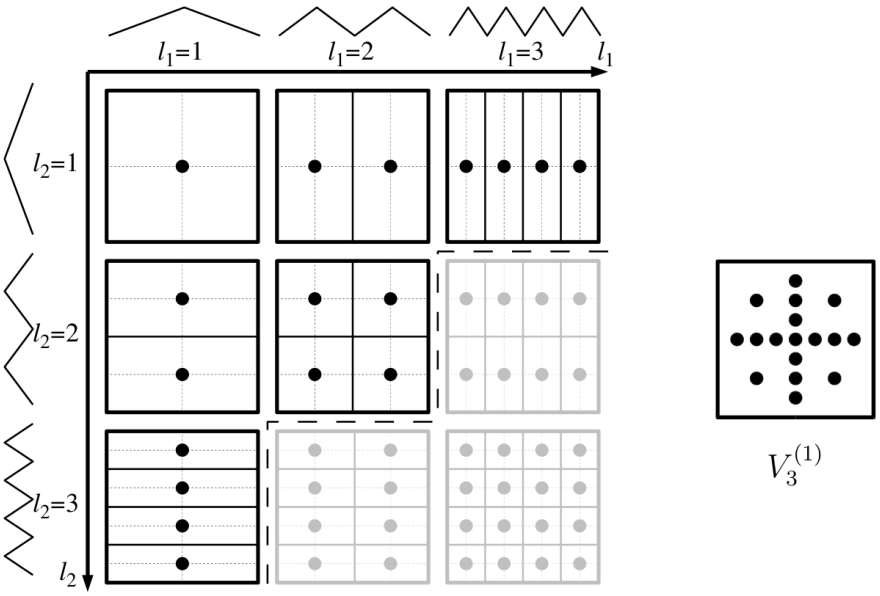


Figure 2.3: Two-dimensional subspaces $W_{\vec{l}}$ up to $l = 3$ with $h_3 = 1/8$ (left) and a sparse grid of level $n = 3$ (right). The chosen selected subspaces (black) form the sparse grid space $V_3^{(1)}$ (right). Subspaces with lots of basis functions of small support (gray) are left out. [15]

In some cases, the function f shows different smoothness characteristics in different regions. Hence, in these areas more grid points are needed to get a more exact representation of f. In this case, an adaptive refinement of the sparse grids is used to add grid points in certain regions. The refinement is done after a grid is initialized at a certain level. Therefore, an approximation error is calculated to see in which regions more grid points are needed to make the representation of f more exact in this areas. In figure 2.4, a sparse grid of level 2 is used at the beginning and afterwards gets refined two times.

2.2 Density Estimation with Sparse Grids

Density estimation is the computation of a density value that belongs to a data point in the dataset. According to Peherstorfer [14], SGDE is an application of sparse grids with the aim of approximating the unknown density function $p(\mathbf{X})$ with a random

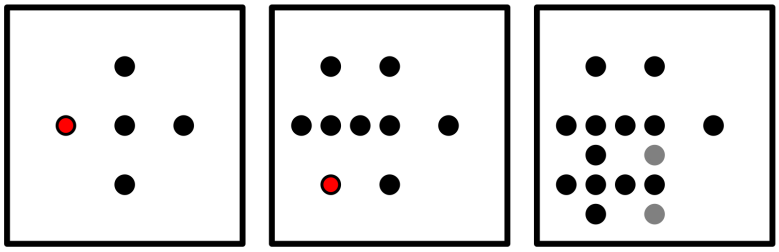


Figure 2.4: A regular sparse grid of level 2 (left) is refined at the red grid point by adding all its children in the tree of hierarchical basis functions (middle). This process is repeated again (right). The gray points (right) are missing hierarchical ancestors, which need to be created recursively. [15]

variable $\boldsymbol{X}$, where data points of a dataset $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subseteq \mathbb{R}^d$ are sampled from. This approximation is constructed with a density estimation $\hat{p}$ of p based on the data points in $\mathcal{D}$. A grid-based solution for the density estimation problem is obtained by a solution for the following minimization problem

$$\tilde{p} = \arg\min_{f \in V} \int_\Omega (f(\boldsymbol{x}) - p_\epsilon(\boldsymbol{x}))^2 \mathrm{d}\boldsymbol{x} + \lambda \|\Lambda f\|_{L^2}^2,$$

where p_ϵ is a highly-overfitted initial guess, $\lambda > 0$ is a regularization parameter, V is a function space, $\|\Lambda f\|_{L^2}^2$ is a regularization term, e.g. Λ is chosen to ∇, and $\|\cdot\|_{L^2}$ is the L^2-norm. In this approach, a smoother function $\hat{p}$ is obtained by spline smoothing.

Sparse grids density estimation looks for an estimation $\hat{p}$ in the space of (adaptive) sparse grids, $\hat{p} \in V_n^{(1)}$, of level n with a set of hierarchical basis functions Φ_n. In SGDE, the equation

$$\int_\Omega \hat{p}(\boldsymbol{x}) \varphi(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} + \lambda \int_\Omega \Lambda \hat{p}(\boldsymbol{x}) \cdot \Lambda \varphi(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \frac{1}{N} \sum_{j=1}^{N} \varphi(\boldsymbol{x}_j)$$

needs to hold for all $\varphi \in \Phi_n$. The advantage of SGDE compared to KDE is that the complexity grows with number of grid points not with the number of data points.

In the implementation of SGDE [14], a density function for every location $\boldsymbol{x} \in [0,1]^d$ is learned for a normalized dataset $\mathcal{D} \subset [0,1]^d$. SGDE takes batches of data points and for every batch, SGDE learns a density function using grid refinement to improve the results of the learned function. The split of the dataset into batches for computing is feasible because of the adaptivity of sparse grids.

Figure 2.5 shows the density function for an example dataset $\mathcal{D}$. It can be clearly seen that the density values are higher at areas with more densly packed data points than at regions with spatially distributed data points (figure 2.6).

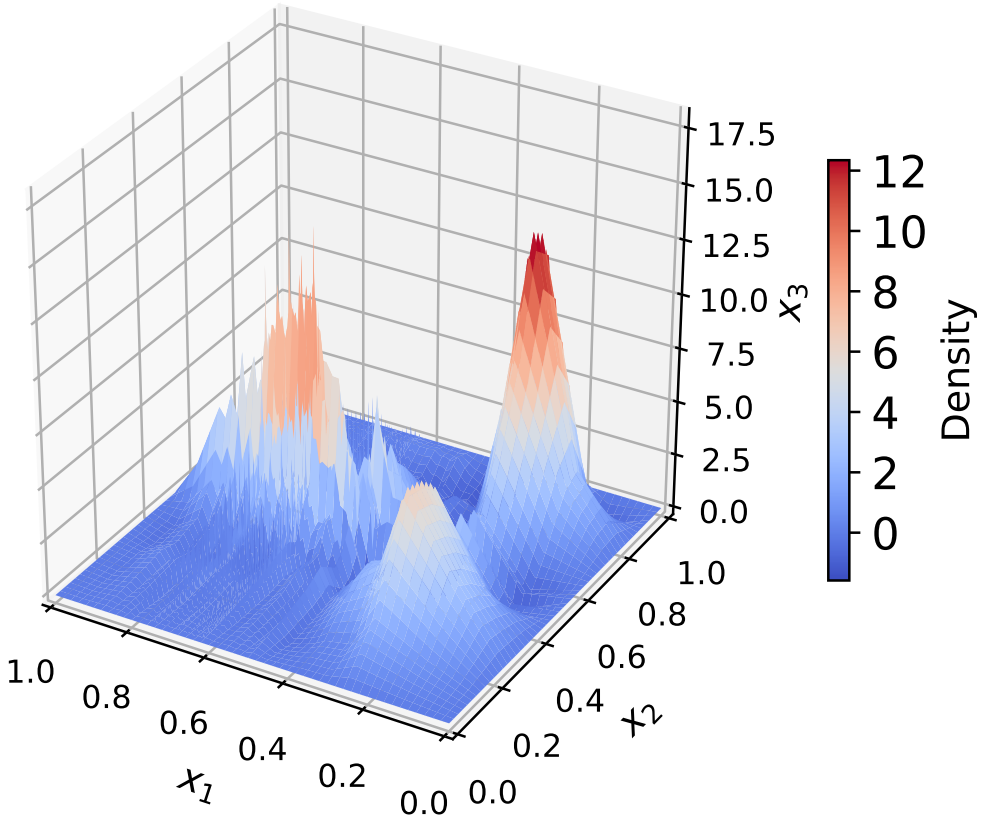


Figure 2.5: Learned density function of an example dataset.

This density function $\tilde{p}(\boldsymbol{x})$ can be evaluated at every data point $\boldsymbol{x} \in \mathcal{D} \subset [0,1]^d$, or more precisely at every point $\boldsymbol{x} \in [0,1]^d$. Figure 2.6 shows the evaluation of $\tilde{p}(\boldsymbol{x})$ at every data point $\boldsymbol{x}$ in the example dataset.

2.3 Outlier Detection with Clustering Methods

Given a dataset $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, $|\mathcal{D}| = N$, containing d-dimensional data points $\boldsymbol{x}_i \in \mathcal{D}$, $1 \leq i \leq N$, sampled from a given data space $\mathcal{S}$ ($\mathcal{D} \subset \mathcal{S}$), usually $\mathcal{S} = \mathbb{R}^d$, with an unknown probability density $\hat{f}(\boldsymbol{x}_i)$. Clustering is an unsupervised machine learning task that tries to find similarities among objects within a dataset $\mathcal{D}$ and forms groups of data points with similar properties, so-called clusters. Beside the different clusters, there is also a set of noisy points, usually called outliers, which contains objects that do not fit to any of the clusters. At the end of the clustering process it is important that similar

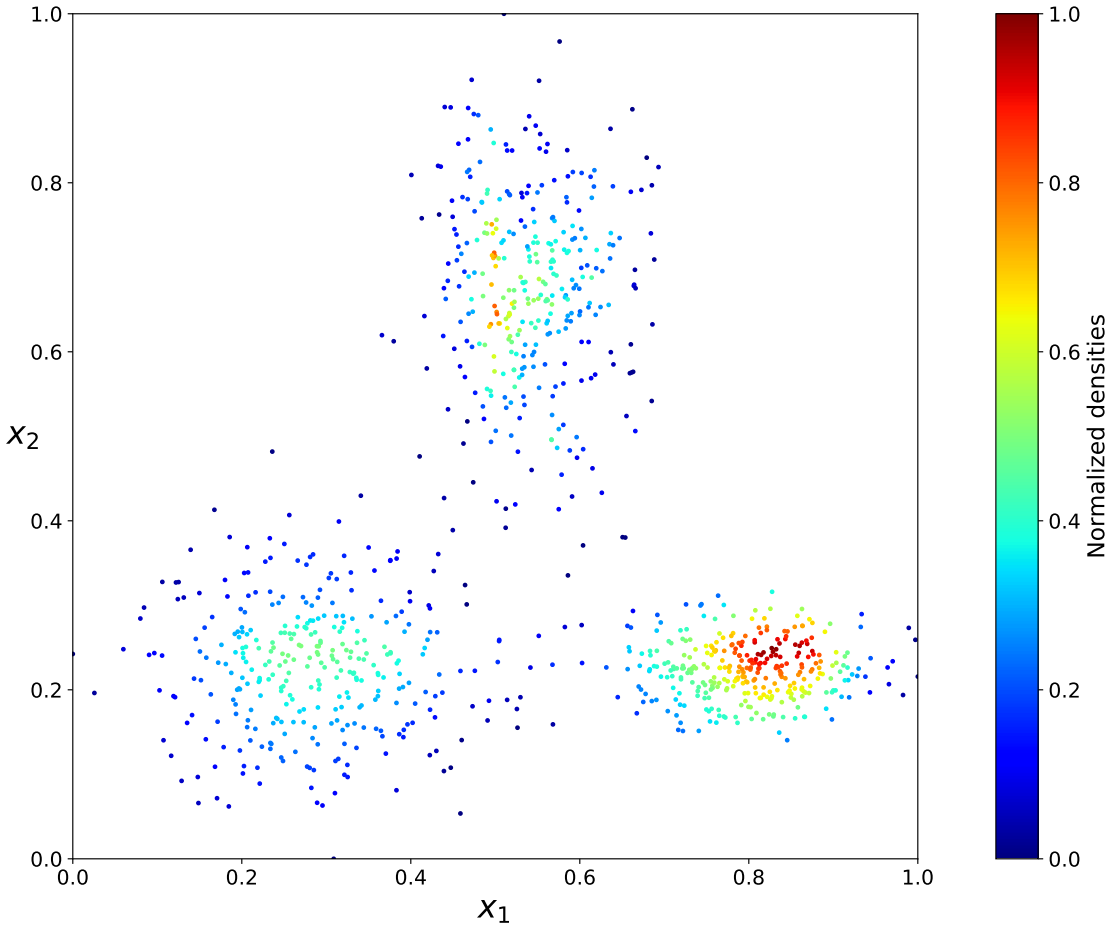


Figure 2.6: SGDE of an example dataset.

objects are assigned to the same cluster and dissimilar objects have to be assigned to different clusters or the set containing the outliers. The dissimilarity of two objects can be measured using some distance function, usually the Euclidean distance or the Manhatten distance. Today, there are many approaches with different algorithms to do clustering, e.g. distance-based clustering, centroid-based clustering, connectivity-based (hierarchical) clustering, distribution-based or density-based clustering, which will be presented in further detail in section 2.3.1.

Since clustering is an unsupervised learning task, it can be used to get a first insight into the structure of an unknown dataset. There are clustering algorithms, e.g. DBSCAN (section 2.3.2), that consider outliers during the clustering process. Hence, these algorithms can be used for outlier detection.

2.3.1 Density-based Clustering

As presented in Kriegel, Kröger, Sander, and Zimek [10], density-based clustering is a nonparametric subdomain of clustering that uses the probability density $\hat{f}(\boldsymbol{x})$ of the data points $\boldsymbol{x} \in \mathcal{D}$ to identify different clusters. Therefore, an area with a high density $\hat{f}(\boldsymbol{x})$ is considered to be a cluster. As a consequence, regions with lower density separate different clusters from each other. Advantages of density-based clustering algorithms are their neutral behaviour towards the underlying density $\hat{f}(\boldsymbol{x})$ or the variance within a cluster and they do not require information about the number of clusters. On a simple level, density-based clusters are data points in regions with a higher density than a given threshold at where the underlying density function is 'cut' off. It might be possible that different clusters are merged together to one cluster if this density threshold is too small. If the threshold is too high, regions with lower density will not be identified as clusters, but classified as outliers. OPTICS (Ordering Points To Identify the Clustering Structure) [3] is a well-known density-based algorithm. It orders data points based on their density which helps to identify cluster structures in a dataset. Another density-based clustering algorithm is DBSCAN, which is explained in further detail in the following.

2.3.2 DBSCAN

DBSCAN is a density-based clustering algorithm introduced by Ester et al. [7]. This algorithm depends on two parameters *Eps* and *MinPts*, but according to Ester et al., it is possible to reduce the number of input parameters to one. DBSCAN assigns all data points to the same cluster if they are density-reachable from each other. Defining $d(\boldsymbol{x}_j, \boldsymbol{x}_i)$ as some distance function for two data points $\boldsymbol{x}_j$ and $\boldsymbol{x}_i$, e.g. the Euclidean distance. A data point $\boldsymbol{x}_i$ is directly density-reachable from $\boldsymbol{x}_j$ if $\boldsymbol{x}_i$ is inside the *Eps*-neighborhood of $\boldsymbol{x}_j$, $\boldsymbol{x}_i \in \mathrm{N}_{Eps}(\boldsymbol{x}_j) = \{\boldsymbol{x}_k \in \mathcal{D} \mid d(\boldsymbol{x}_j, \boldsymbol{x}_k) \leq Eps\}$, where the core point codition holds,

$$|\mathrm{N}_{Eps}(\boldsymbol{x}_j)| \geq MinPts.$$

Using the definition of directly density-reachability, $\boldsymbol{x}_i$ is density-reachable from $\boldsymbol{x}_j$ if a chain $\boldsymbol{x}_j = \boldsymbol{x}_1, \ldots, \boldsymbol{x}_n = \boldsymbol{x}_i$ of data points can be found where $\boldsymbol{x}_{i+1}$ is directly density-reachable from $\boldsymbol{x}_i$ for $1 \leq i \leq n-1$. According to these definitions, it is also possible that two clusters are merged into one cluster. When the algorithm terminates, all data points that are not assigned to a cluster of normal points are classified to the noise cluster. In this thesis, the implementation of *scikit-learn* [13] is used to compute DBSCAN.

Figure 2.7 shows the results of the DBSCAN algorithm with different parameter settings. It can clearly be seen that DBSCAN finds more small clusters for lower

values of Eps and $MinPts$. If the the parameter $MinPts$ takes a higher value, DBSCAN assigns data points at outer cluster regions with lower density to the noise cluster. The explanation for this is given by the fact that $MinPts$ defines the minimal number of core points that form a cluster.

2.3.3 Local Density Estimate

The Local Density Estimate presented by Latecki et al. [11] is based on a nonparametric kernel density estimate and estimates the local density of a data point $\boldsymbol{x}_j \in \mathcal{D} \subset \mathbb{R}^d$. The LDE takes into account the local neighborhood of $\boldsymbol{x}_j$ to determine its local density. This local neighborhood is formed by the m nearest neighbors $\boldsymbol{x}_i \in mNN(\boldsymbol{x}_j)$ of $\boldsymbol{x}_j$, where m should be chosen large enough to get satisfiable results. To perform density estimation, LDE uses the reachability distance, which is the maximum value of the squared Euclidean distance $d(\boldsymbol{x}_j, \boldsymbol{x}_i) = \|\boldsymbol{x}_j - \boldsymbol{x}_i\|^2$ and the distance $d_k(\boldsymbol{x}_i)$ from $\boldsymbol{x}_i$ to its kth nearest neighbor,

$$rd_k(\boldsymbol{x}_j, \boldsymbol{x}_i) = \max\{d(\boldsymbol{x}_j, \boldsymbol{x}_i),\, d_k(\boldsymbol{x}_i)\}.$$

Using h as a fixed bandwidth, the LDE of a data point $\boldsymbol{x}_j$ is defined as

$$\mathrm{LDE}(\boldsymbol{x}_j) = \frac{1}{m} \cdot \sum_{\boldsymbol{x}_i \in mNN(\boldsymbol{x}_j)} \frac{1}{(2\pi)^{\frac{d}{2}} (h \cdot d_k(\boldsymbol{x}_i))^d} \exp\left(- \frac{rd_k(\boldsymbol{x}_j, \boldsymbol{x}_i)^2}{2(h \cdot d_k(\boldsymbol{x}_i))^2} \right).$$

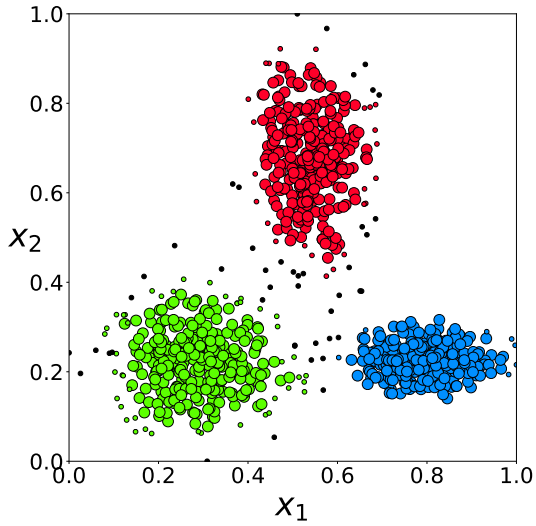
The difference between LDE and a KDE is that LDE does not sum over the whole dataset like KDE, but it only uses the local neighborhood $mNN(\boldsymbol{x}_j)$ of a data point $\boldsymbol{x}_j$. This reduces the runtime complexity from $\mathcal{O}(n^2)$ of the original KDE to $\mathcal{O}(mn \log n)$ of LDE. Since m has an influence on the computing time, it should not be chosen to large. LDE can be seen as an alternative to SGDE and it is used in LDF (section 3.3.3).

2.3.4 Outliers

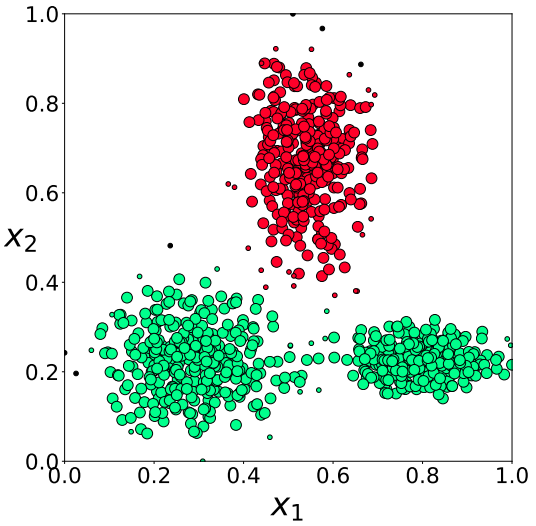
According to Hawkins [8], an outlier can be intuitively defined as "an observation which deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism." In other words, an outlier is a data point in a dataset which does not belong to any cluster. According to D. and Babu [6], outliers can be subdivided in three different categories:

Point Outliers

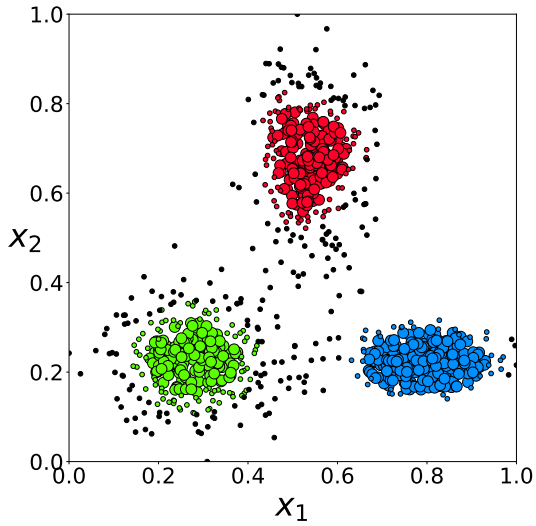
A point outlier is just a single data point that deviates from the rest of the data points in the dataset. In general, the focus of most outlier detection techniques is the identification of point outliers.



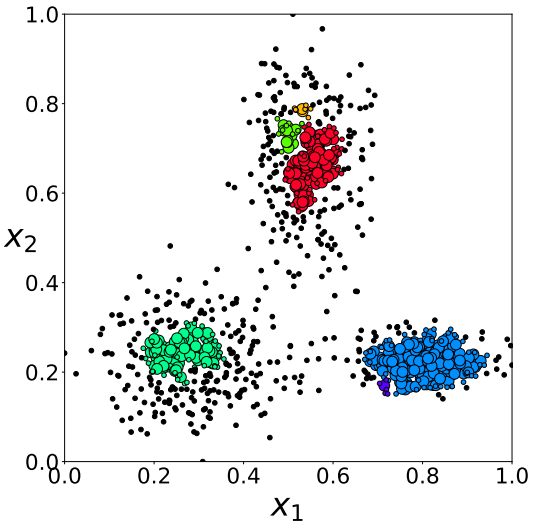
(a) DBSCAN with $Eps = 0.05$ and $MinPts = 10$.



(b) DBSCAN with $Eps = 0.07$ and $MinPts = 10$.



(c) DBSCAN with $Eps = 0.05$ and $MinPts = 30$.



(d) DBSCAN with $Eps = 0.02$ and $MinPts = 8$.

Figure 2.7: DBSCAN with different parameter settings computed for dataset $\mathcal{D}_1$ (table 3.1). Each color represents an own cluster. Data points in the noise cluster are marked as black points.

Contextual Outliers

A contextual outlier is a data point that is anomalous with respect to a specific context of the dataset, e.g. to a single cluster.

Collective Outliers

Collective outliers are data points that are not necessarily anomalous by themselves with respect to each other, but together deviate from the rest of the dataset. This category of outliers can only be found in datasets where data points have a relationship between each other, e.g. an anomalous interval of measurements inside of a time series.

2.4 Outlier Detection Techniques

Outlier detection techniques are methods that help to identify data points in a dataset as outliers. There are different types of outlier detection techniques, e.g. density-based or distance-based methods. In this thesis, different unsupervised outlier detection techniques, like LDF (section 3.3.3) and LOF (section 3.3.4), are explained in further detail. Furthermore, a way to use the Silhouette Coefficient (section 3.3.5) for outlier detection is presented. Finally, a new own approach (section 3.3.6) for an outlier detection technique is developed and presented.

2.5 Outlier Detection Measures

Outlier detection measures are measures to determine the accuracy of outlier detection techniques. There are many outlier detection measures that are based on different approaches. In this thesis, the presented measures use a ranking of the data points based on their outlierness, which is determined by the corresponding outlier detection technique. The following measures are used in the thesis and they are explained in more detail in the implementation part:

- Detection rate and false alarm rate
- Precision at n
- Adjusted Precision at n
- Average Precision
- Adjusted Average Precision
- Area under the receiver operating characteristics curve (ROC AUC)

3 Implementation

In this chapter, the implementation part of the thesis is explained in further detail. This includes necessary steps to work with the existing codebase [14]. The codebase in this thesis makes use of the SG++ library [15] to perform SGDE. Additionally, the different outlier detection techniques and methods that were implemented during the thesis are presented in this chapter.

3.1 Data Preprocessing

In the first step, the data points of a given dataset has to be converted to a format which can be used by the density estimation process, which is the d-dimensional unit-hypercube, $[0,1]^d$. Therefore, all attributes of a given dataset that are not real numbers have to be mapped to real numbers. Another way to get matching datasets is to generate own ones. In the following, this process will be explained in further detail.

3.1.1 Data Generation

Artificial datasets with two- or three-dimensional data points are created with Python by sampling from different probability distributions using the library *numpy*. Additionally, in a similar fashion, two- and three-dimensional datasets with a certain percentage of true outliers are created to validate the performance of outlier detection techniques and measures.

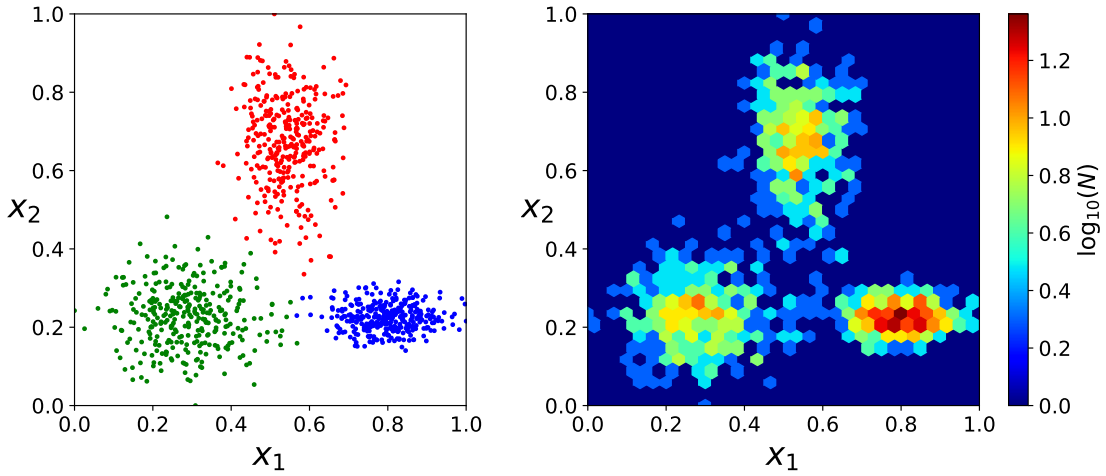
Table 3.1 gives information about the sizes, dimensionality and the number of true outliers of the generated artificial datasets. All generated datasets consist of 1000 data points because the SGDE process (section 2.2) is relatively fast for that size of a dataset and the density values are reasonable. The most generated datasets have either two- or three-dimensional data points because the validation and evaluation for high-dimensional datasets only rely on numerical measures and a meaningful visual representation cannot be used anymore.

Dataset $\mathcal{D}_1$ is a two-dimensional dataset containing three Gaussian clusters and no points are considered to be true outliers. The data points $(X_1, X_2) = \boldsymbol{x} \in \mathcal{D}_1 \subset \mathbb{R}^2$ are samples from different normal distributions, where the components X_1 and X_2 of the first Gaussian cluster with 334 data points are distributed normally with $X_1 \sim \mathcal{N}(0, 0.5)$

Dataset	Description	Dimensions	Size	Outliers
$\mathcal{D}_1$	3 Gaussians	2	1000	0
$\mathcal{D}_2$	S-Curve	2	1000	0
$\mathcal{D}_3$	3 Gaussians	3	1000	0
$\mathcal{D}_4$	S-Curve	3	1000	0
$\mathcal{D}_5$	3 Gaussians	2	1000	50
$\mathcal{D}_6$	S-Curve	2	1000	50
$\mathcal{D}_7$	3 Gaussians	3	1000	50
$\mathcal{D}_8$	S-Curve	3	1000	50
$\mathcal{D}_9$	3 Gaussians	4	1000	50

Table 3.1: Information about used generated datasets.

and $X_2 \sim \mathcal{N}(2,1)$. For the second Gaussian cluster with 333 data points, X_1 and X_2 are distributed normally with $X_1 \sim \mathcal{N}(-2,0.8)$ and $X_2 \sim \mathcal{N}(-2,0.7)$. For the third Gaussian cluster with 333 data points, X_1 and X_2 are distributed normally with $X_1 \sim \mathcal{N}(2,0.6)$ and $X_2 \sim \mathcal{N}(-2,0.3)$. Figure 3.1 shows dataset $\mathcal{D}_1$, where (a) visualizes each Gaussian cluster dyed in a different color and (b) shows the heatmap of $\mathcal{D}_1$ based on the logarithm of number of data points in a specific hexagon.



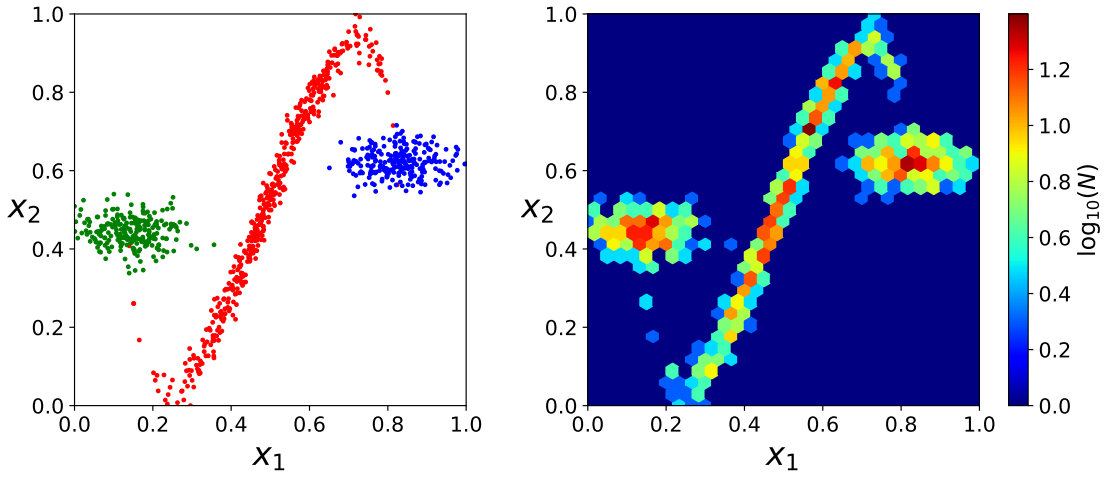
(a) Dataset $\mathcal{D}_1$, colored.

(b) Dataset $\mathcal{D}_1$, heatmap.

Figure 3.1: Dataset $\mathcal{D}_1$ (table 3.1).

Dataset $\mathcal{D}_2$ is a two-dimensional dataset containing two Gaussian clusters and a

cluster shaped like the letter 'S'. In $\mathcal{D}_2$, no points are considered to be true outliers. The data points $(X_1, X_2) = \boldsymbol{x} \in \mathcal{D}_2 \subset \mathbb{R}^2$ are samples from different normal distributions, where the components X_1 and X_2 of the 'S'-cluster with 500 data points are distributed normally with $X_1 \sim \mathcal{N}(1, 4)$. The values of X_1 are forced to be in $[-10, 10]$ and $X_2 = Y \cdot \sqrt{10^2 - X_1^2}$ with $Y \sim \mathcal{N}(X_1, 0.4)$. For the first Gaussian cluster with 250 data points, X_1 and X_2 are distributed normally with $X_1 \sim \mathcal{N}(-10, 2)$ and $X_2 = Y + 5$ with $Y \sim \mathcal{N}(-10, 4)$. For the second Gaussian cluster with 250 data points, X_1 and X_2 are distributed normally with $X_1 \sim \mathcal{N}(10, 2)$ and $X_2 = Y + 5$ with $Y \sim \mathcal{N}(10, 4)$. Figure 3.2 shows dataset $\mathcal{D}_2$, where (a) visualizes each cluster dyed in a different color and (b) shows the heatmap of $\mathcal{D}_2$ based on the logarithm of number of data points in a specific hexagon.



(a) Dataset $\mathcal{D}_2$, colored.

(b) Dataset $\mathcal{D}_2$, heatmap.

Figure 3.2: Dataset $\mathcal{D}_2$ (table 3.1).

Dataset $\mathcal{D}_3$ is a three-dimensional dataset containing three Gaussian clusters and no points are considered to be true outliers. The data points $(X_1, X_2, X_3) = \boldsymbol{x} \in \mathcal{D}_3 \subset \mathbb{R}^3$ are samples from different normal distributions, where the components X_1 and X_2 are identical to the components X_1 and X_2 of dataset $\mathcal{D}_1$ for all three Gaussian clusters. The component X_3 for the first Gaussian cluster with 334 data points is distributed normally with $X_3 \sim \mathcal{N}(5, 0.7)$. For the second Gaussian cluster with 333 data points, X_3 is distributed normally with $X_3 = Y - 3$, where $Y \sim \mathcal{N}(-10, 6)$. For the third Gaussian cluster with 333 data points, X_3 is distributed normally with $X_3 = Y + 1$, where $Y \sim \mathcal{N}(10, 0.9)$.

Dataset $\mathcal{D}_4$ is a three-dimensional dataset containing a cluster shaped like the letter 'S'

and two Gaussian clusters. In $\mathcal{D}_4$, no points are considered to be true outliers. The data points $(X_1, X_2, X_3) = \boldsymbol{x} \in \mathcal{D}_4 \subset \mathbb{R}^3$ are samples from different normal distributions, where the components X_1 and X_2 are identical to the components X_1 and X_2 of dataset $\mathcal{D}_2$ for all three clusters. The component X_3 for the 'S'-cluster with 500 data points is distributed normally with $X_3 \sim \mathcal{N}(X_1 \cdot X_2, 0.2)$, where X_1 and X_2 are the 500 X_1 and X_2 components from dataset $\mathcal{D}_2$. For the first Gaussian cluster with 250 data points, X_3 is distributed normally with $X_3 = Y + 3$, where $Y \sim \mathcal{N}(-10, 6)$. For the second Gaussian cluster with 250 data points, X_3 is distributed normally with $X_3 = Y + 3$, where $Y \sim \mathcal{N}(10, 6)$.

Figure 3.3 visualizes the datasets $\mathcal{D}_3$ and $\mathcal{D}_4$, where (a) and (b) show the front view, (c) and (d) the side view and (e), and (f) the top view of $\mathcal{D}_3$ and $\mathcal{D}_4$, respectively.

The datasets $\mathcal{D}_5$ and $\mathcal{D}_7$ are identical to the datasets $\mathcal{D}_1$ and $\mathcal{D}_3$, but the first and the second Gaussian cluster contain only 325 data points each and the third Gaussian cluster contains only 300 data points. In $\mathcal{D}_5$ and $\mathcal{D}_7$, 50 data points are considered to be true outliers. These outlier data points are samples from the uniform distribution, $X_1, X_2, X_3 \sim \mathcal{U}(0, 1)$, and they are added to the datasets after the normalization process (section 3.1.2) of the other 950 data points. Figure 3.4 shows the difference between dataset $\mathcal{D}_1$ without outliers (a) and dataset $\mathcal{D}_5$ with outliers (b).

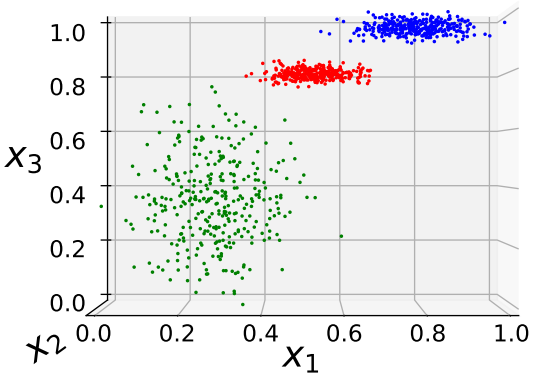
The datasets $\mathcal{D}_6$ and $\mathcal{D}_8$ are identical to the datasets $\mathcal{D}_2$ and $\mathcal{D}_4$, but the first cluster contains only 450 data points. In $\mathcal{D}_6$ and $\mathcal{D}_8$, 50 data points are considered to be true outliers. These outlier data points are samples from the uniform distribution, $X_1, X_2, X_3 \sim \mathcal{U}(0, 1)$, and they are added to the datasets after the normalization process (section 3.1.2) of the other 950 data points.

Dataset $\mathcal{D}_9$ is identital to $\mathcal{D}_7$, but it has one more dimension. X_1 and X_2 of $(X_1, X_2, X_3, X_4) = \boldsymbol{x} \in \mathcal{D}_9 \subset \mathbb{R}^4$ are sampled the same way as component X_1 of $\mathcal{D}_7$. The 50 outlier data points are samples from the uniform distribution, $X_1, X_2, X_3, X_4 \sim \mathcal{U}(0, 1)$, and they are added to the datasets after the normalization process (section 3.1.2) of the other 950 data points.

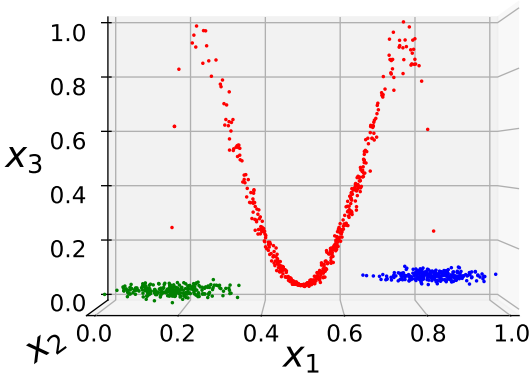
3.1.2 Data Normalization

For the SGDE part it is necessary to normalize the data points into the range $[0, 1]^d$. Therefore, a linear transformation is applied to every component x_i of a data point $(x_1, \ldots, x_d)^T = \boldsymbol{x} \in \mathcal{D}$ to transform it into the range $[a, b]$ with $a = 0$ and $b = 1$. Before applying the transformation to a specific component x_i of a data point, the minimum and maximum value of this component among all data points $\boldsymbol{x} \in \mathcal{D}$ need to be known. This minimum and maximum value are defined as

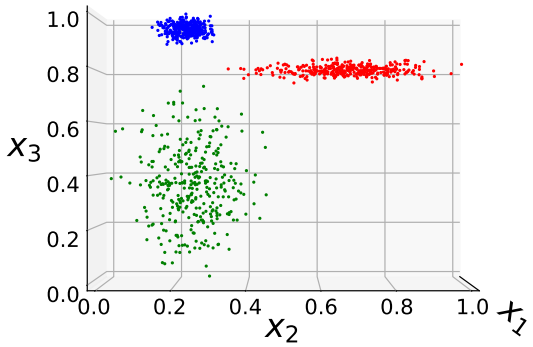
$$x_i^{min} = \min_{\boldsymbol{x} \in \mathcal{D}} x_i$$



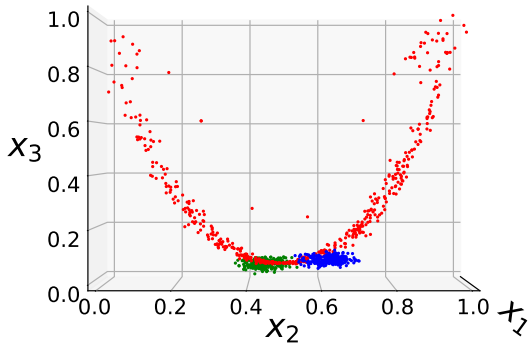
(a) Dataset $\mathcal{D}_3$, front view.



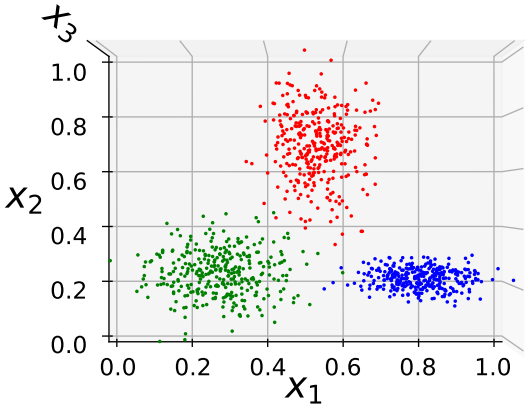
(b) Dataset $\mathcal{D}_4$, front view.



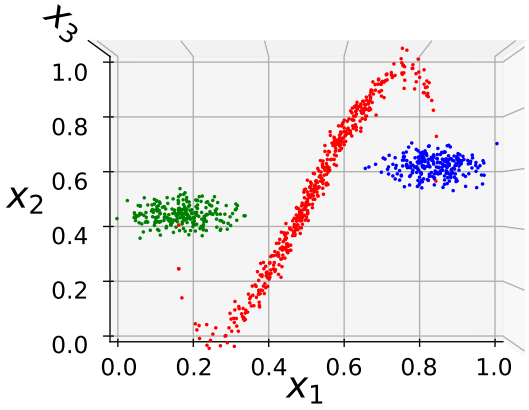
(c) Dataset $\mathcal{D}_3$, side view.



(d) Dataset $\mathcal{D}_4$, side view.

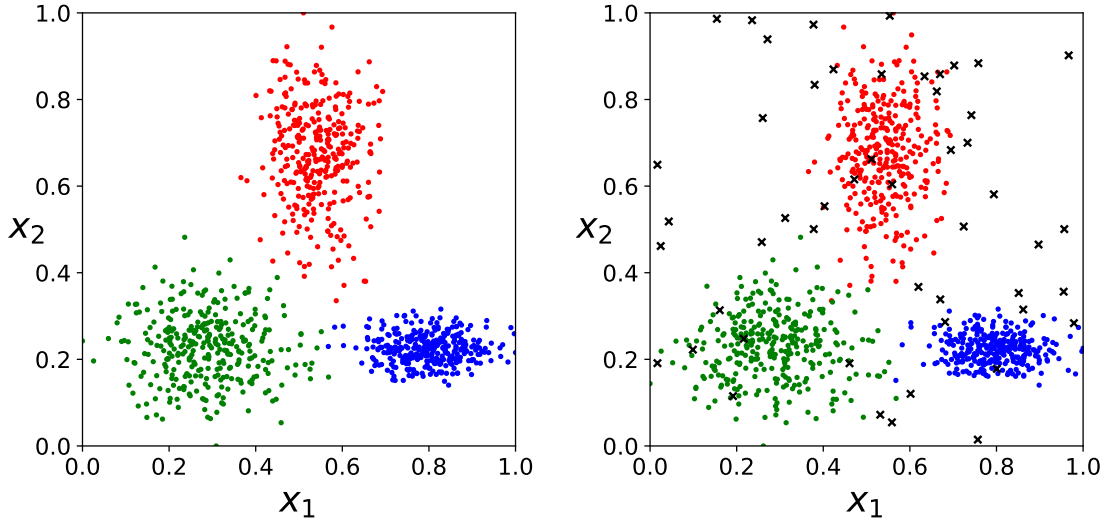


(e) Dataset $\mathcal{D}_3$, top view.



(f) Dataset $\mathcal{D}_4$, top view.

Figure 3.3: The datasets $\mathcal{D}_3$ and $\mathcal{D}_4$ (table 3.1).



(a) Dataset $\mathcal{D}_1$.

(b) Dataset $\mathcal{D}_5$ (outliers are marked with '×').

Figure 3.4: Difference between the datasets $\mathcal{D}_1$ and $\mathcal{D}_5$ (table 3.1).

and

$$x_i^{max} = \max_{\boldsymbol{x} \in \mathcal{D}} x_i.$$

Now, these values can be used to apply the following transformation to the corresponding component of every data point:

$$\tilde{x}_i = a + \frac{x_i - x_i^{min}}{x_i^{max} - x_i^{min}}(b - a). \tag{3.1}$$

The results $\tilde{x}_i$ of equation (3.1) are the components of the normalized data points $(\tilde{x}_1, \ldots, \tilde{x}_d)^T = \tilde{\boldsymbol{x}} \in \tilde{\mathcal{D}} \subset [0,1]^d$.

In the case that one or more components x_i of a data point $\boldsymbol{x} \in \mathcal{D}$ lie on at least one of the boundaries 0 or 1 of $[0,1]$, the SGDE with a linear hat basis function cannot estimate the density of $\boldsymbol{x}$. A solution to this problem is to apply the linear transformation in equation (3.1) with other parameters, e.g. $a = 0.01$ and $b = 0.99$.

After the normalization process, the data $\tilde{\mathcal{D}}$ is handed to the processing part. In the following, it is considered that $\mathcal{D}$ is already normalized.

3.2 Data Processing

The processing of the data points including SGDE and computing LDE and LDF was performed in C++. Hence, the exising codebase was used to do SGDE. It starts with reading in some *.json* files containing different parameters for SGDE, like the dimension of the data points that will be computed. After that, the program reads in the data points from a *.txt* file and stores them in a *std::vector<Vector>*, where *Vector* is implemented as a *std::vector<double>*. In the following step, a *DataController* is set up, which passes the data to the SGDE learner in the *DensityEstimation* class. This *DataController* needs an adapter to be able to read data points from the *std::vector<Vector>* and pass them to the SGDE learner. So, this adapter had to be implemented first, to get SGDE running. After the initialization of the *DataController* and the assignment of the adapter to it, a *DensityEstimation* object is initialized with this *DataController*. The *DensityEstimation* object learns the underlying density function of a given dataset. Another modification of the existing codebase was to implement a method that returns the learned density function with a shared pointer outside of the *DensityEstimation* class. After that, the density function can be used in the *main()* method and evaluated a every point $[0, 1]^d$, where d is the dimension of the data points in the dataset.

After the data processing part, the densities and the LDF values for the data points are stored in a *.txt* file to pass them to the data postprocessing part (section 3.4).

3.3 Outlier Detection Techniques

Outlier detection techniques are methods that are able to detect outliers in a given dataset. To do so, these techniques use different kind of measures, e.g. the densities of the data points obtained by SGDE. In the following, different outlier detection techniques are presented in further detail.

3.3.1 Flat Clustering Approach

The flat clustering approach is a very basic idea how to extract clusters from the densities of the data points and it helps to get familiar with the used exitsting codebase and to test the SGDE process. The idea behind this approach is to take a look at the densities $f(\boldsymbol{x})$ of all data points $\boldsymbol{x}$ in the dataset $\mathcal{D}$ and cut at a specific threshold α that separates the outliers from the inliers. This cut means that all data points $\boldsymbol{x} \in \mathcal{D}$ where $f(\boldsymbol{x}) \leq \alpha$ are considered to be outliers, otherwise inliers.

3.3.2 Belief Function

The use of a belief function proposed by Xu, Xu, and Feng [17] helps to detect outliers by calculating the credibility of a data point $\boldsymbol{x} \in \mathcal{D}$ using its probability density. This belief function is defined as

$$r(\boldsymbol{x}) = \ln\left(\frac{f(\boldsymbol{x})}{\alpha}\right),$$

where $f(\boldsymbol{x})$ is the probability density of data point $\boldsymbol{x}$. In the data processing part (section 3.2), f is the density function learned by SGDE. The parameter α is a certain probability level, usually the product of a coefficient and the expectation value of $f(\boldsymbol{x})$. If $r(\boldsymbol{x}) > 0$, the data point $\boldsymbol{x}$ is seen as an inlier or a normal datum, otherwise as an outlier.

3.3.3 Local Density Factor

LDF introduced by Latecki et al. [11] is an outlier detection method based on a density estimate. In their paper, they use LDE (section 2.3.3) as density estimate. The LDF at data point $\boldsymbol{x}_j \in \mathcal{D}$ is defined as

$$\mathrm{LDF}(\boldsymbol{x}_j) = \frac{\sum\limits_{\boldsymbol{x}_i \in mNN(\boldsymbol{x}_j)} \frac{\mathrm{LDE}(\boldsymbol{x}_i)}{m}}{\mathrm{LDE}(\boldsymbol{x}_j) + c \cdot \sum\limits_{\boldsymbol{x}_i \in mNN(\boldsymbol{x}_j)} \frac{\mathrm{LDE}(\boldsymbol{x}_i)}{m}}, \tag{3.2}$$

where $mNN(\boldsymbol{x}_j)$ is the set containing the m nearest neighbors of $\boldsymbol{x}_j$, $\mathrm{LDE}(\boldsymbol{x}_i)$ is the local density estimate of a data point $\boldsymbol{x}_i$ and c is a scaling constant, which is set to $c = 0.1$ in the implementation according to Latecki et al. [11]. This scaling constant normalizes the LDF values to $[0, 1/c]$.

The value $\mathrm{LDF}(\boldsymbol{x}_j) = 0$ means that $\mathrm{LDE}(\boldsymbol{x}_j) \gg \sum_{\boldsymbol{x}_i \in mNN(\boldsymbol{x}_j)} \frac{\mathrm{LDE}(\boldsymbol{x}_i)}{m}$, so the LDE of data point $\boldsymbol{x}_j$ is significantly greater than the average LDE of its m nearest neighbors. Therefore, $\boldsymbol{x}_j$ can be considered to be not an outlier. On the other hand, a data point $\boldsymbol{x}_j$ can be considered to be an outlier if $\mathrm{LDF}(\boldsymbol{x}_j)$ is close to $1/c$, where the edge case of $\mathrm{LDF}(\boldsymbol{x}_j) = 1/c$ is caused by $\mathrm{LDE}(\boldsymbol{x}_j) = 0$. Keeping that in mind, outliers can be detected using a threshold T with $\mathrm{LDF}(\boldsymbol{x}_j) > T$, where $T \in [0, 1/c]$.

In the thesis, this method is written in C++ in a flexible way: the implemented LDE densities can be exchanged with those obtained from SGDE (section 2.2).

3.3.4 Local Outlier Factor

LOF is an approach of Breunig et al. [4] to determine the outlyingness of a data point in a given dataset. It uses the reachability distance, defined as the maximum value of

the Euclidean distance $d(\boldsymbol{x}_j, \boldsymbol{x}_i) = \|\boldsymbol{x}_j - \boldsymbol{x}_i\|$ and the distance $d_k(\boldsymbol{x}_i)$ from $\boldsymbol{x}_i$ to its kth nearest neighbor,

$$rd_k(\boldsymbol{x}_j, \boldsymbol{x}_i) = \max\{d(\boldsymbol{x}_j, \boldsymbol{x}_i),\, d_k(\boldsymbol{x}_i)\}.$$

The reachability distance used in LOF differs from that one used in LDE by using the normal Euclidean distance. LOF focuses on the data points in the local neighborhood $N_{d_k(\boldsymbol{x}_j)}(\boldsymbol{x}_j) = \{\boldsymbol{x}_i \in \mathcal{D} \setminus \{\boldsymbol{x}_j\} \mid d(\boldsymbol{x}_j, \boldsymbol{x}_i) \leq d_k(\boldsymbol{x}_j)\}$ of $\boldsymbol{x}_j$, which in the following will be denoted as $N_k(\boldsymbol{x}_j)$. Using the above definitions, the local reachability density of $\boldsymbol{x}_j$ can be defined as

$$lrd_{MinPts}(\boldsymbol{x}_j) = \left(\frac{1}{|N_{MinPts}(\boldsymbol{x}_j)|} \cdot \sum_{\boldsymbol{x}_i \in N_{MinPts}(\boldsymbol{x}_j)} rd_{MinPts}(\boldsymbol{x}_j, \boldsymbol{x}_i) \right)^{-1}.$$

Finally, using the definition of $lrd_{MinPts}(\boldsymbol{x}_j)$, the LOF is defined as

$$\mathrm{LOF}_{MinPts}(\boldsymbol{x}_j) = \frac{1}{|N_{MinPts}(\boldsymbol{x}_j)|} \cdot \sum_{\boldsymbol{x}_i \in N_{MinPts}(\boldsymbol{x}_j)} \frac{lrd_{MinPts}(\boldsymbol{x}_i)}{lrd_{MinPts}(\boldsymbol{x}_j)}.$$

It takes one input parameter *MinPts* that defines the minimal number of points which form the local neighborhood of $\boldsymbol{x}_j$. The LOF gives a degree of outlyingness for a data point $\boldsymbol{x}_j$. A $\mathrm{LOF}_{MinPts}(\boldsymbol{x}_j) \approx 1$ means that $\boldsymbol{x}_j$ has a similar density compared to the data points in its local neighborhood. If $\mathrm{LOF}_{MinPts}(\boldsymbol{x}_j) < 1$, then $\boldsymbol{x}_j$ can be considered as an inlier, else if $\mathrm{LOF}_{MinPts}(\boldsymbol{x}_j) \gg 1$, then $\boldsymbol{x}_j$ can be considered as an outlier. In this thesis, the implementation from *scikit-learn* [13] is used to evaluate LOF.

3.3.5 Silhouette Coefficient

The Silhouette Coefficient presented by Rousseeuw [16] is a value that gives information about the quality of a performed clustering. This value can be computed for all data points in the dataset $\mathcal{D}$. Using the Euclidean distance $d(\boldsymbol{x}_j, \boldsymbol{x}_i) = \|\boldsymbol{x}_j - \boldsymbol{x}_i\|$, the average dissimilarity of $\boldsymbol{x}_j \in A$ to all other data points $\boldsymbol{x}_i \in A$ in cluster A is defined as

$$a(\boldsymbol{x}_j) := \frac{1}{|A|} \sum_{\boldsymbol{x}_i \in A} d(\boldsymbol{x}_j, \boldsymbol{x}_i).$$

The average dissimilarity of $\boldsymbol{x}_j \in A$ to all data points in another cluster $C \neq A$, which is considered to be minimal for a cluster B, is defined as

$$b(\boldsymbol{x}_j) := \min_{C \neq A} \left\{ \frac{1}{|C|} \sum_{\boldsymbol{x}_i \in C} d(\boldsymbol{x}_j, \boldsymbol{x}_i) \right\}$$

The Silhouette Coefficient $s(\boldsymbol{x}_j)$ of a data point $\boldsymbol{x}_j$ uses the values $a(\boldsymbol{x}_j)$ and $b(\boldsymbol{x}_j)$ and it is computed with

$$s(\boldsymbol{x}_j) = \frac{b(\boldsymbol{x}_j) - a(\boldsymbol{x}_j)}{\max\{a(\boldsymbol{x}_j),\, b(\boldsymbol{x}_j)\}}.$$

This definition gives the restriction $-1 \leq s(\boldsymbol{x}_j) \leq 1$ for all $\boldsymbol{x}_j \in \mathcal{D}$. A value $s(\boldsymbol{x}_j) \approx 1$ means that $a(\boldsymbol{x}_j) \ll b(\boldsymbol{x}_j)$ and that the choice of assigning $\boldsymbol{x}_j$ to cluster A was the most likely one. In the worst case, the Silhouette Coefficient takes a value of $s(\boldsymbol{x}_j) \approx -1$, where $a(\boldsymbol{x}_j) \gg b(\boldsymbol{x}_j)$. In this case, $\boldsymbol{x}_j$ should have been more likely assinged to cluster B instead of cluster A. A Silhouette Coefficient of $s(\boldsymbol{x}_j) \approx 0$ is caused by $a(\boldsymbol{x}_j) \approx b(\boldsymbol{x}_j)$, which means that $\boldsymbol{x}_j$ lies between the clusters A and B. In the thesis, this last property of the Silhouette Coefficient is used to identify outliers. In the first step, the clustering process for a dataset $\mathcal{D}$ is computed using the implementation of the k-Means algorithm from *scikit-learn* [13]. k-Means is one of the most popular clustering algorithms and it takes one input parameter k. k-Means tries to separate all data points $\boldsymbol{x} \in \mathcal{D}$ into k clusters with equal variance by minimizing the sum of the squared distance of a data point to the mean of the cluster it is assigned to. After computing the k-Means clustering, the Silhouette Coefficient $s(\boldsymbol{x})$ is computed for all data points $\boldsymbol{x} \in \mathcal{D}$ using the *scikit-learn* [13] implementation. After that, the Silhouette Coefficient values are plotted for all data points, where the values of two data points that are assigned to the same cluster are plotted side by side. In a second figure, the data points are plotted colored the same way as their corresponding plots of the Silhouette Coefficients. A way to identify outliers is to choose the l data points $\boldsymbol{x}_o$ with the smallest absolute value $|s(\boldsymbol{x}_o)|$ in every cluster C_i, $1 \leq i \leq k$. This l data points can be considered as outliers and marked as outliers in the plot because they lie in between to clusters.

Figure 3.5 shows an example plot of the Silhouette Coefficient values together with the used dataset $\mathcal{D}_2$. It can be seen that $s(\boldsymbol{x}) < 0$ for data points $\boldsymbol{x}$ in the light green and gray cluster. This means that the points on the border between these two clusters are not well-clustered. In figure 3.5, it can be seen that the cluster with the shape of the letter 'S' and the Gaussian cluster on the right side are not well-clustered according to human intuition. Applied to dataset $\mathcal{D}_2$, k-Means fails because it only cares about minimizing the sum of squared distances of data points to the cluster means and it does not take the underlying density into account.

3.3.6 Own Approach

During the thesis, a new own approach for outlier detection with SGDE is introduced. For a dataset $\mathcal{D} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ with $|\mathcal{D}| = N$, the arithmetic mean of the densities of

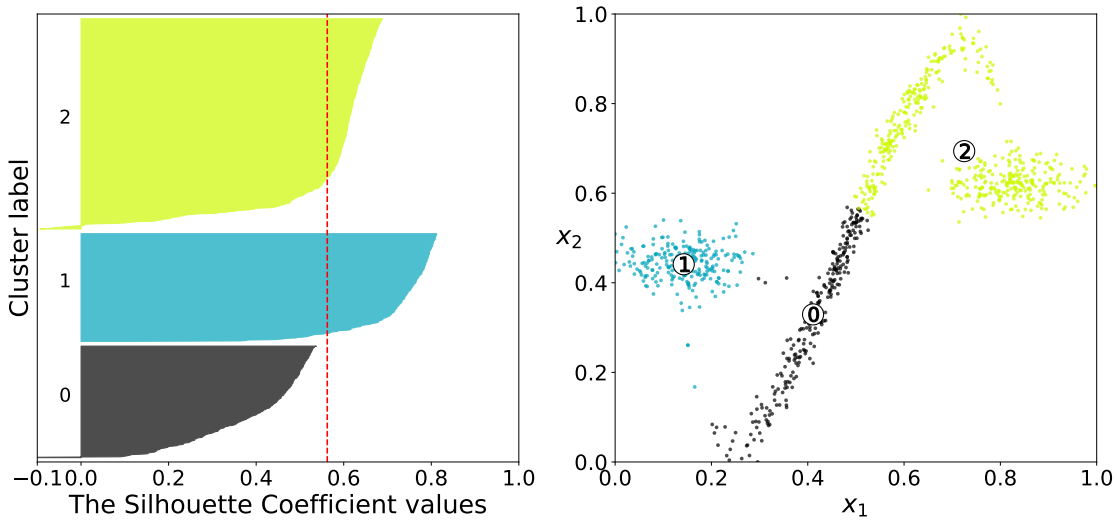


Figure 3.5: Silhouette Coefficient plot of dataset $\mathcal{D}_2$ (table 3.1). Clustering is performed with k-Means with $k = 3$. Clusters are dyed in different colors. The circles with numbers insight are the corresponding cluster centers (means). The average Silhouette Coefficient is 0.562 (red dashed line).

its data points $\boldsymbol{x}_i$ is defined as

$$\mu(\mathcal{D}) := \frac{1}{N} \sum_{i=1}^{N} f(\boldsymbol{x}_i),$$

where $f(\boldsymbol{x}_i)$ are the densities values of the data points returned by SGDE. For data points that are assigned to the same cluster $C_{\boldsymbol{x}}$ as $\boldsymbol{x} \in \mathcal{D}$, e.g. by DBSCAN (section 2.3.2), the arithmetic mean of the densities of all data points in $C_{\boldsymbol{x}}$ is defined as

$$\mu(C_{\boldsymbol{x}}) := \frac{1}{|C_{\boldsymbol{x}}|} \sum_{\boldsymbol{x}' \in C_{\boldsymbol{x}}} f(\boldsymbol{x}'),$$

where $f(\boldsymbol{x}')$ are the density values returned by SGDE. Using these two definitions, the new approach is based on two properties of a data point $\boldsymbol{x}$ that have an effect on being an outlier. The first feature (1) is measured by the density of a data point compared to the arithmetic mean $\mu(\mathcal{D})$ of the density values of all data points in the dataset. If a data point has a low density compared to $\mu(\mathcal{D})$, it is more likely an outlier. The second property (2) to classify a data point $\boldsymbol{x}$ as an outlier is measured by its density of compared to the arithmetic mean $\mu(C_{\boldsymbol{x}})$ of the density values of all data points that are in the same cluster $C_{\boldsymbol{x}}$ as $\boldsymbol{x}$. If $\boldsymbol{x}$ has a low density compared to $\mu(C_{\boldsymbol{x}})$, $\boldsymbol{x}$ is more

likely an outlier. The combination of (1) and (2) measured for a data point leads to the formula of the new approach,

$$v(\boldsymbol{x}) = \underbrace{\frac{f(\boldsymbol{x})}{\mu(\mathcal{D})}}_{(1)} \cdot \underbrace{\frac{f(\boldsymbol{x})}{\mu(C_{\boldsymbol{x}})}}_{(2)},$$

where a lower value $v(\boldsymbol{x})$ tells that $\boldsymbol{x}$ is more likely an outlier. If either $(1) < 1$ or $(2) < 1$, or $(1) < 1$ and $(2) < 1$ for a data point $\boldsymbol{x}$, $v(\boldsymbol{x})$ will take a small value, $0 \leq v(\boldsymbol{x}) < 1$, and $\boldsymbol{x}$ is considered to be an outlier. Consequential, outliers can be detected with $v(\boldsymbol{x}) \leq \alpha$, where $\alpha > 0$ is some threshold.

This approach is implemented in Python and starts by computing the DBSCAN algorithm (section 2.3.2) with different parameters *Eps* and a fixed value for *MinPts*. For each parameter setting, the average Silhouette Coefficient (section 3.3.5) is computed. Since a high Silhouette Coefficient gives information about a good clustering, *Eps* is set to the value where the Silhouette Coefficient is at the maximum. After that, DBSCAN is computed with this fixed *Eps* value and different values for the *MinPts* parameter. The optimal *MinPts* value is chosen the same way as *Eps*. In this first step, the single values for the DBSCAN parameters has to be chosen carefully because it needs at least two classes, the noise cluster and a normal cluster, to compute the average Silhouette Coefficient. Finally, DBSCAN is computed with both optimal values for *Eps* and *MinPts*. The class labels for every data point are stored in the DBSCAN object and they are used together with the SGDE values to calculate $\mu(C_{\boldsymbol{x}})$. The value $\mu(\mathcal{D})$ is also computed with the SGDE values. After that $v(\boldsymbol{x})$ is calculated using $\mu(\mathcal{D})$, $\mu(C_{\boldsymbol{x}})$ and the SGDE values $f(\boldsymbol{x})$ for all $\boldsymbol{x} \in \mathcal{D}$.

3.4 Data Postprocessing

In the data postprocessing part, different outlier detection measures are applied to the data points and their densities obtained by SGDE to measure the performance of the outlier detection techniques that are applied to the data in the data processing part. The data postprocessing was completely performed in Python.

3.5 Outlier Detection Measures

Outlier detection measures are metrics to determine the accuracy of outlier detection techniques. To validate the outlier detection results, datasets with true outliers are used, e.g. $\mathcal{D}_5$. In the following, several outlier detection measures are explained in further detail.

3.5.1 Detection Rate and False Alarm Rate

Two typical measures to evaluate outlier detection algorithms are the detection rate and the false alarm rate. These two rates are defined as presented in Latecki et al. [11],

$$\text{DetectionRate} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$
$$\text{FalseAlarmRate} = \frac{\text{FP}}{\text{FP} + \text{TN}},$$

where TP (true positives) is the number of correctly detected outliers, FN (false negatives) is the number of outliers that are not detected, FP (false positives) is the number of normal data points that are incorrectly detected as outliers and TN (true negatives) is the number of normal data points that are correctly identified as normal data points. Table 3.2 shows the relation between this four possible outcomes which occur in outlier detection.

	Predicted Outliers	**Predicted Normal**
Actual Outliers	True Positives (TP)	False Negatives (FN)
Actual Normal	False Positives (FP)	True Negatives (TN)

Table 3.2: Confusion matrix with possible scenarios for detecting outliers [11].

So, the detection rate is the ratio of correctly detected outliers and the false alarm rate is the ratio of normal data points that are incorrectly detected as outliers. In this thesis, the detection rate and the false alarm rate are implemented in Python to evaluate the outlier detection performed by DBSCAN (section 2.3.2), where data points are assigned to the noise cluster. Hence, the labels for the data points stored in the DBSCAN object are used to see if true outliers or true inliers are assigned to the noise cluster or not. With this information, the four numbers TP, FP, TN and FN are obtained easily and the detection rate and false alarm rate can be calculated. Furthermore, the detection rate and the false alarm rate are computed with the k most likely outliers identified by LDF (section 3.3.3) or by the new own approach (section 3.3.6).

3.5.2 Precision at n

The precision at n (P@n) measure [5] is computed for a dataset $\mathcal{D}$ containing outliers $\mathcal{O} \subset \mathcal{D}$ and inliers $\mathcal{I} \subseteq \mathcal{D}$ ($\mathcal{D} = \mathcal{I} \cup \mathcal{O}$) using

$$\text{P@}n = \frac{|\{\boldsymbol{o} \in \mathcal{O} \mid rank(\boldsymbol{o}) \leq n\}|}{n}.$$

It gives validation ratio based on the correctly ranked ouliers in the top n ranks. The ranking, which represents an outlierness score of the data points, must be unique, where data points with the same rank can be ranked in an arbitrary, consistent way. In the implementation, the ranking is based on the decreasing order of the LDF values or the increasing order of the values obtained from the new approach.

3.5.3 Adjusted Precision at n

The adjusted precision at n measure is a modification of precision at n. It is proposed by Campos et al. [5]. This modification is based on the fact that the maximum possible value among the results of the original precision at n measure is $|\mathcal{O}|/n$ if $n > |\mathcal{O}|$ and 1 otherwise. Furthermore, the expected value of a completely random outlier ranking is $|\mathcal{O}|/N$ for $|\mathcal{D}| = N$. So, the modified precision at n computes values for $n \leq |\mathcal{O}|$ with

$$\text{Adjusted P@}n = \frac{\text{P@}n - \frac{|\mathcal{O}|}{N}}{1 - \frac{|\mathcal{O}|}{N}}. \tag{3.3}$$

For $n > |\mathcal{O}|$, 1 has to be replaced by $|\mathcal{O}|/n$ in equation (3.3). A disadvantage of the adjusted precision at n measure is, that it is very sensitive to choice of n. In the implementation, the ranking of the data points is created with the decreasing values of LDF or the increasing values of the new approach.

3.5.4 Average Precision

The average precision (AP) measure [5] averages the P@n measure for all ranks of the outlier points $\boldsymbol{o} \in \mathcal{O}$. It is defined as

$$\text{AP} = \frac{1}{|\mathcal{O}|} \sum_{\boldsymbol{o} \in \mathcal{O}} \text{P@}rank(\boldsymbol{o}). \tag{3.4}$$

The ranking needed by the precision at n measure is performed by ordering the data points in the decreasing order of their LDF values or in the increasing order of the values obtained by the new approach.

3.5.5 Adjusted Average Precision

The adjusted average precision measure, proposed by Campos et al. [5], is a modification of the average precision measure. This modification is performed the same way as that of the adjusted precision at n measure. So, the fact that maximum value of 1 is received by a perfect ranking and the expected value is $|\mathcal{O}|/N$ for a random ranking, gives the modification

$$\text{Adjusted AP} = \frac{\text{AP} - \frac{|\mathcal{O}|}{N}}{1 - \frac{|\mathcal{O}|}{N}}.$$

3.5.6 ROC AUC

The ROC AUC measure [5] gives information about the ranking of a dataset $\mathcal{D} = \mathcal{I} \cup \mathcal{O}$ with true inliers $\mathcal{I}$ and true outliers $\mathcal{O}$. A ROC AUC value of 1 corresponds to a perfect ranking, while a ROC AUC value of 0 is caused by an inverted perfect ranking. More precisely, the ROC AUC measure gives the true positive rate over n objects in the top ranks, where n is taken from the ranks of all inliers in $\mathcal{I}$. It is defined as

$$\text{ROC AUC} := \underset{\boldsymbol{o}\in\mathcal{O},\boldsymbol{i}\in\mathcal{I}}{mean} \begin{cases} 1 & \text{if } score(\boldsymbol{o}) > score(\boldsymbol{i}) \\ \frac{1}{2} & \text{if } score(\boldsymbol{o}) = score(\boldsymbol{i}) \\ 0 & \text{if } score(\boldsymbol{o}) < score(\boldsymbol{i}). \end{cases}$$

In the implementation, the score of $\boldsymbol{x} \in \mathcal{D}$ corresponds to the LDF of $\boldsymbol{x}$, $score(\boldsymbol{x}) = \text{LDF}(\boldsymbol{x})$, or to the values obtained by the new approach, $score(\boldsymbol{x}) = v(\boldsymbol{x})$, which have to be inverted afterwards for the second case.

3.6 Data Visualization Techniques

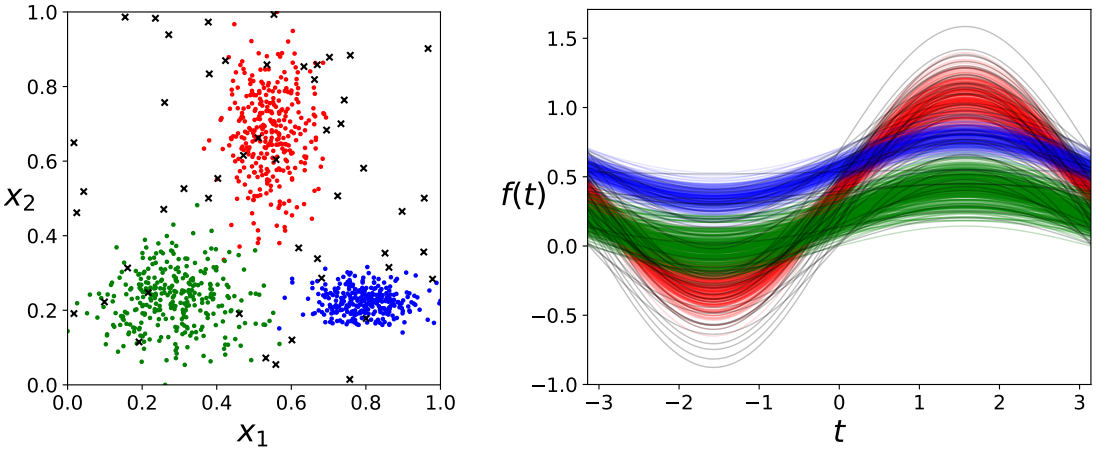
Data visualization techniques are displaying methods to visualize high-dimensional data. Therefore, the data is transformed into another representation. In the following, the Andrews curves visualization method is explained in further detail.

3.6.1 Andrews Curves

Visualizing high-dimensional data with two-dimensional curves based on sine and cosine functions is proposed by Andrews [2]. For every data point $\boldsymbol{x} = (x_1, \ldots, x_d)^T \in \mathcal{D} \subset \mathbb{R}^d$ construct a finite Fourier series of the following form:

$$f_{\boldsymbol{x}}(t) = \frac{x_1}{\sqrt{2}} + x_2 \sin(t) + x_3 \cos(t) + x_4 \sin(2t) + x_5 \cos(2t) + \ldots \tag{3.5}$$

After that, all functions $f_{\boldsymbol{x}}(t)$ are plotted in the range $-\pi < t < \pi$. This range represents the whole function because of the periodic behaviour of the sine and cosine functions. Figure 3.6 shows dataset $\mathcal{D}_5$ (a) and the corresponding curves (b) constructed from $\mathcal{D}_5$.



(a) Dataset $\mathcal{D}_5$ (outliers are marked with '×').

(b) Andrews curves plot of dataset $\mathcal{D}_5$.

Figure 3.6: Andrews curves plot of dataset $\mathcal{D}_5$ (table 3.1).

One helpful property of these curves is that they preserve distances:

$$\|f_{\boldsymbol{x}_i}(t) - f_{\boldsymbol{x}_j}(t)\|_{L^2} = \int_{-\pi}^{\pi} \left(f_{\boldsymbol{x}_i}(t) - f_{\boldsymbol{x}_j}(t)\right)^2 \mathrm{d}t = \pi \, \|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2,$$

with $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathcal{D}$. Therefore, Andrews curves can be used to identify cluster structures or outliers visually, but there is no guarantee for that. In this thesis, the Andrews curves visualization technique is implemented in Python.

4 Evaluation

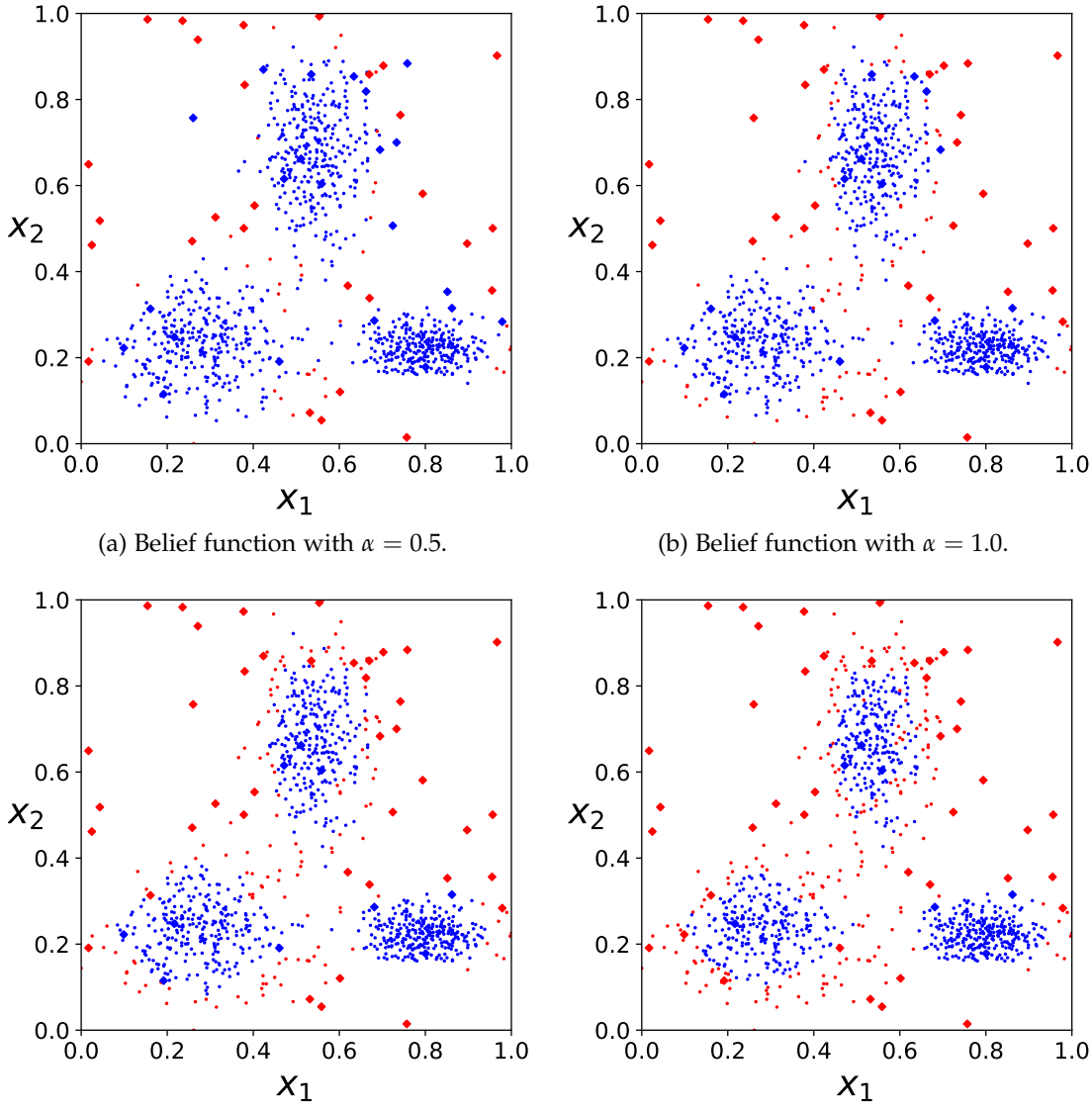
In this chapter, the results of the outlier detection techniques presented in this thesis are evaluated and compared. Therefore, the generated artificial datasets (table 3.1) are used to analyze and validate the mentioned outlier detection methods. Additionally, real datasets from Campos et al. [5] are used for validation and testing reasons. These real datasets are randomly chosen examples of real-life datasets and do not reflect the whole range of possible datasets available. As a consequence, further work with the focus of testing a bigger range of real datasets and parameter settings, which extends past the time and scope of this thesis, is necessary to draw conclusions on a larger basis of experiment outcomes. Table 4.1 gives information about the used real datasets.

Dataset	Dimensions	Size	Outliers	Duplicates
Wilt	5	4671	93	yes
Shuttle	9	1013	13	no

Table 4.1: Information about used real datasets.

Since the flat clustering approach (section 3.3.1) and the belief function (section 3.3.2) both aim to consider a certain density value as a boundary between outliers and inliers, the results for both methods with the same threshold α are equal. Figure 4.1 shows the results of the belief function approach for different threshold values α. The true outliers in dataset $\mathcal{D}_5$ are sampled from a uniform distribution, $X_1, X_2 \sim \mathcal{U}(0,1)$. As a consequence, theses true outliers are not limited to lie between normal clusters, but they can also lie inside normal clusters. In figure 4.1, it can be seen that the flat clustering approach or the belief function approach is able to detect outliers if they lie outside of clusters and if the threshold is well-chosen. The detection of true outliers that lie very close on the boundary of a cluster is possible by increasing α. Detecting true outliers which lie in (dense) cluster regions is not possible with the belief function approach without misclassifying a great number of normal data points as outliers. This correlation can be seen in figure 4.2, where the false alarm rate grows with an increasing detection rate caused by a growing threshold α.

The modification of the Silhouette Coefficient plot (section 3.3.5) is shown in figure 4.3.



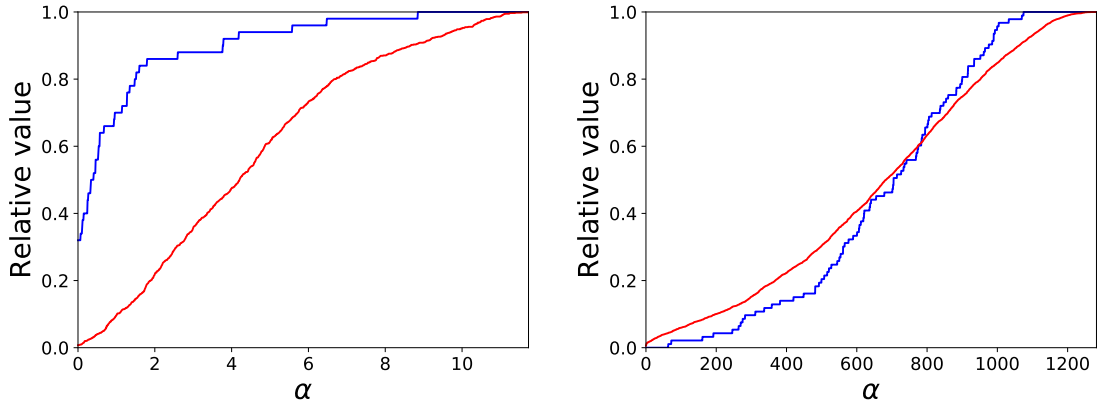
(a) Belief function with $\alpha = 0.5$.

(b) Belief function with $\alpha = 1.0$.

(c) Belief function with $\alpha = 1.5$.

(d) Belief function with $\alpha = 2.0$.

Figure 4.1: Results of the belief function approach applied to dataset $\mathcal{D}_5$ (table 3.1). True outliers are marked as big dots with diamond shape. Red dots are classified as outliers, blue dots are classified as normal data. SGDE uses 295 grid points.



(a) Belief function approach applied to dataset $\mathcal{D}_5$.

(b) Belief function approach applied to the Wilt dataset.

Figure 4.2: Detection rate (blue) and false alarm rate (red) of the belief function approach applied to dataset $\mathcal{D}_5$ (table 3.1) and the Wilt dataset (table 4.1).

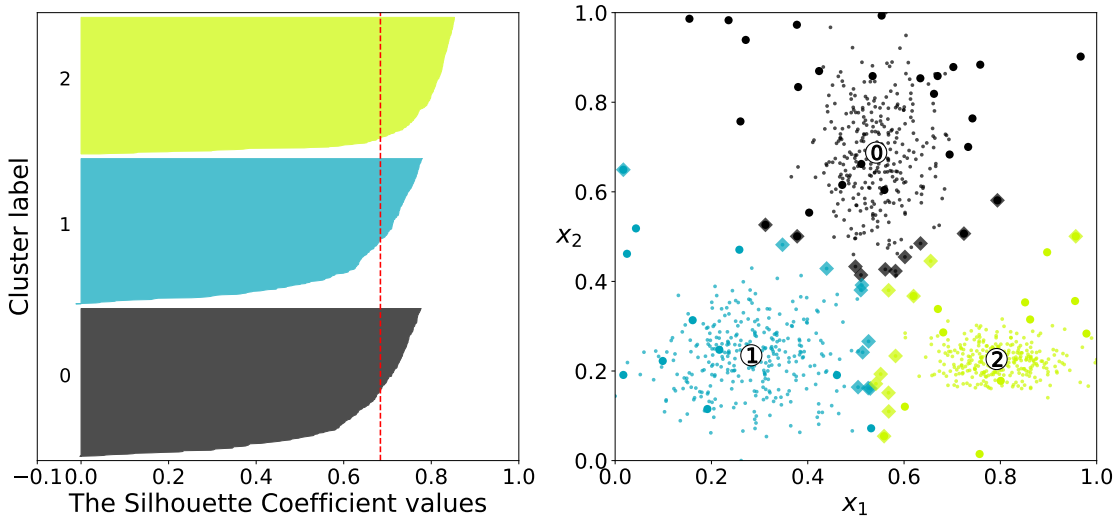
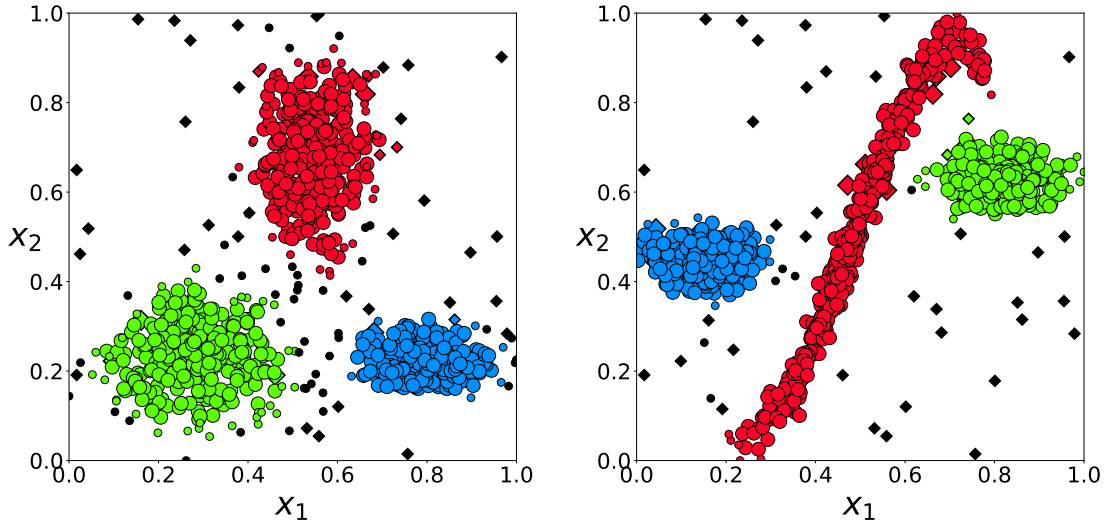


Figure 4.3: Silhouette Coefficient plot of dataset $\mathcal{D}_2$ (table 3.1). Clustering is performed with k-Means with $k = 3$. Distinct clusters are dyed in different colors. The average Silhouette Coefficient is 0.683 (red dashed line). True outliers are marked as dots of bigger size. Detected outliers are marked as dots with diamond shape.

As already mentioned in the implementation, the Silhouette Coefficient focuses on the quality of clustering. Thus, the modified plot only detects outliers which lie on the boundary between different clusters. So, the Silhouette Coefficient can be used to identify and remove noisy data points that lie between two clusters to receive clear boundaries for clustering. Compared to the belief function approach, which detects ouliers that are in areas with less density with respect to a specific threshold α, the Silhouette Coefficient can be used to detect outliers on cluster boundaries no matter what density these regions have.

The DBSCAN algorithm (section 2.3.2) is another choice to detect outliers. During the execution of the algorithm, outliers are assigned to a noise cluster for noisy data points. Figure 4.4 shows the results of the DBSCAN algorithm applied to the datasets $\mathcal{D}_5$ and $\mathcal{D}_6$. It can be seen that a higher detection rate and at the same time a smaller false alarm rate are received for dataset $\mathcal{D}_6$ for the same parameter setting. This can be explained by the fact that the data points in $\mathcal{D}_6$ which belong to normal clusters are more densly packed with less single points on the cluster boundaries compared to the data points in $\mathcal{D}_5$. Hence, less normal data points are misclassified as outliers.



(a) DBSCAN of $\mathcal{D}_5$ with $Eps = 0.05$ and $MinPts = 10$. DetectionRate = 0.64 and FalseAlarmRate = 0.047.

(b) DBSCAN of $\mathcal{D}_6$ with $Eps = 0.05$ and $MinPts = 10$. DetectionRate = 0.7 and FalseAlarmRate = 0.006

Figure 4.4: DBSCAN of datasets $\mathcal{D}_5$ and $\mathcal{D}_6$ (table 3.1). Each color represents an own cluster. Data points in the noise cluster are marked as black dots. True outliers are marked as dots with diamond shape.

Comparing the plots of the belief function approach in figure 4.1 (d) and DBSCAN in figure 4.4 (a) it can be seen that DBSCAN is a better choice for outlier detection because even with a badly chosen parameter setting it only misclassifies data points that are no core points of a cluster.

The LDF gives a degree of outlyingness for every data point in the dataset. Figure 4.5 shows the results of the LDF approach applied to dataset $\mathcal{D}_5$ using different parameter settings for SGDE. It can be seen that different settings for the parameters of SGDE have an influence on the result of LDF because the computed densities are different.

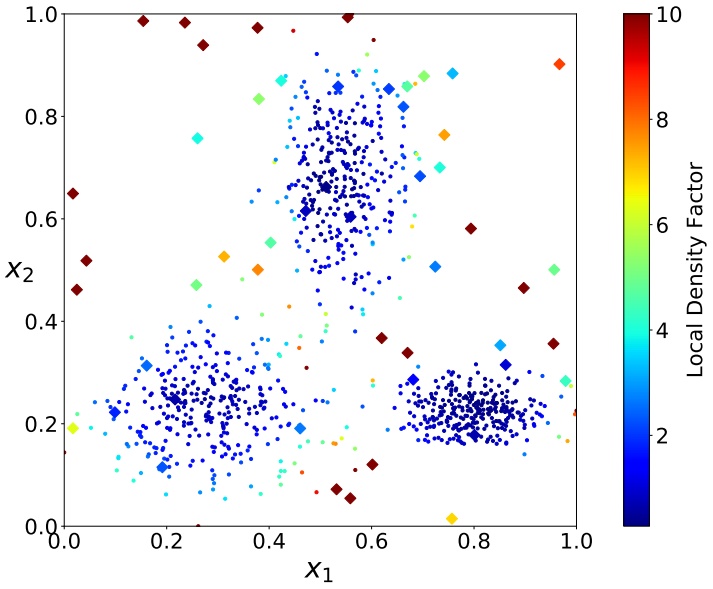
Figure 4.6 shows the evaluation of the result of LDF in figure 4.5 (a). The average precision is AP = 0.509, the adjusted average precision is Adjusted AP = 0.483 and ROC AUC = 0.875.

Figure 4.7 shows the evaluation of LDF applied to dataset $\mathcal{D}_9$. For this dataset, SGDE uses 849 grid points to perform density estimation. The average precision is AP = 0.737, the adjusted average precision is Adjusted AP = 0.723 and ROC AUC = 0.989. The Adjusted P@n curve in figure 4.7 (b) tells that all outlier points are ranked in the top $n \approx 100$ ranks. Furthermore, the high P@n values for the first $n \approx 100$ ranks and the high ROC AUC value give information about a nearly perfect outlier ranking. This means that LDF performed well for dataset $\mathcal{D}_9$ with a batch size of 50 and 10 refinement steps.

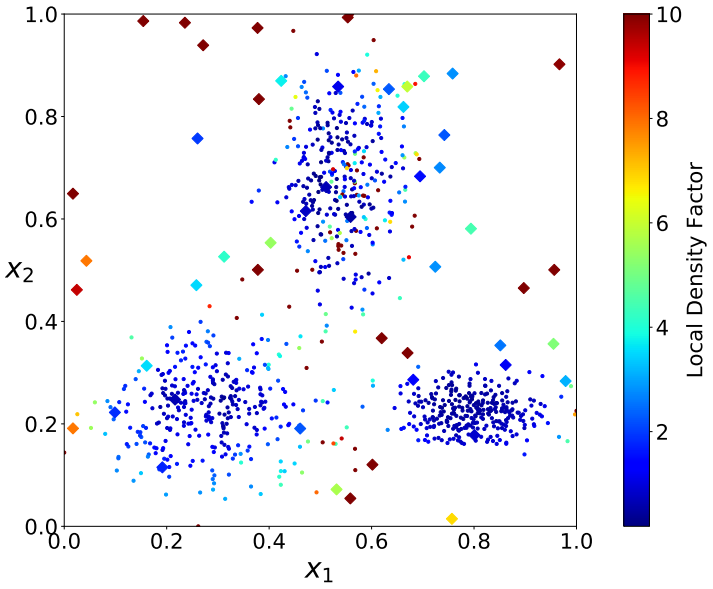
The LOF depends on the parameter k, which is the size of the local neighborhood of a data point. Figure 4.8 shows the LOF computed with different values for k. It can be clearly seen that a higher value for k gives tighter boundaries for the regions of the LOF values. Hence, outlier detection is improved, but the computing time will take longer.

Comparing the plot for the LDF in figure 4.5 (a) and the plot for the LOF in figure 4.8 (d) it can be seen that for this dataset the LOF is a good technique to detect outliers around all clusters. In this case, the result of LDF is similar to the result of LOF and both methods achieve good results for outlier detection.

The new approach presented in this thesis focuses on the density of a data point compared to the arithmetic mean of the densities of all data points and the arithmetic mean of the densities of the data points in the same cluster, respectively. Figure 4.9 shows the results of the new approach applied to dataset $\mathcal{D}_5$ using different parameter settings for DBSCAN. SGDE uses 295 grid points after 10 refinement steps to perform density estimation. The results in figure 4.9 (a) are based on the execution of DBSCAN with $Eps = 0.05$ and $MinPts = 10$. DBSCAN finds 3 clusters besides the noise cluster and it achieves an average Silhouette Coefficient of 0.616. Furthermore, the obtained values for the detection rate and the false alarm rate are DetectionRate = 0.64 and FalseAlarmRate = 0.047. Based on this results of DBSCAN, the new approach achieves an average precision of AP = 0.486, an adjusted average precision of Adjusted AP = 0.459 and ROC AUC = 0.863. The corresponding P@n and Adjusted P@n curves can be seen

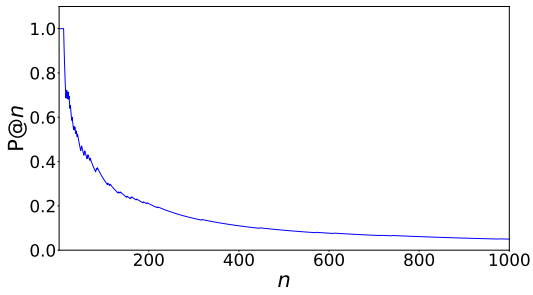


(a) LDF with 295 grid points and batch size 50.

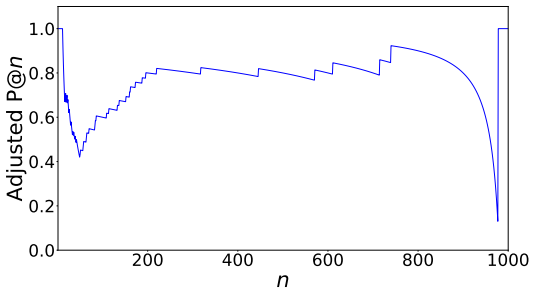


(b) LDF with 317 grid points and batch size 200.

Figure 4.5: LDF of dataset $\mathcal{D}_5$ (table 3.1) with different batch sizes for the SGDE learner, $m = 30$ and $c = 0.1$. The start level of the sparse grids is $n = 3$ and 10 refinement steps are performed. True outliers are marked as big dots with diamond shape.

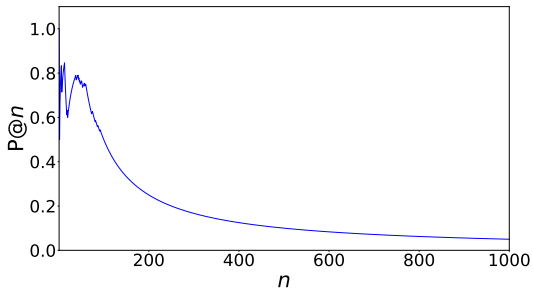


(a) P@n.

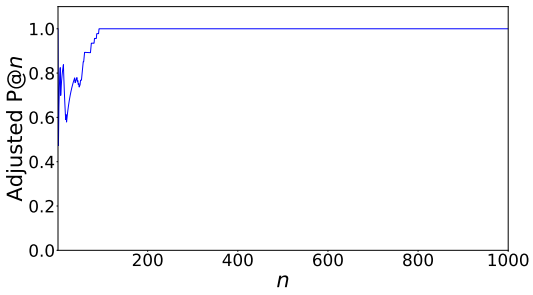


(b) Adjusted P@n.

Figure 4.6: P@n and Adjusted P@n for the LDF applied to dataset $\mathcal{D}_5$ (table 3.1). The fast decrease of the Adjusted P@n curve in the range of $n \in [0, 50]$ tells that some normal points have a higher outlier rank than true outliers. The increasing value after this range tells that more and more outliers are ranked one after another. The fast decrease of the Adjusted P@n value for $n \approx 975$ gives information that a lot of normal points are ranked in this range of n. An Adjusted P@n value of 1 tells that all outliers are ranked in the top n ranks.

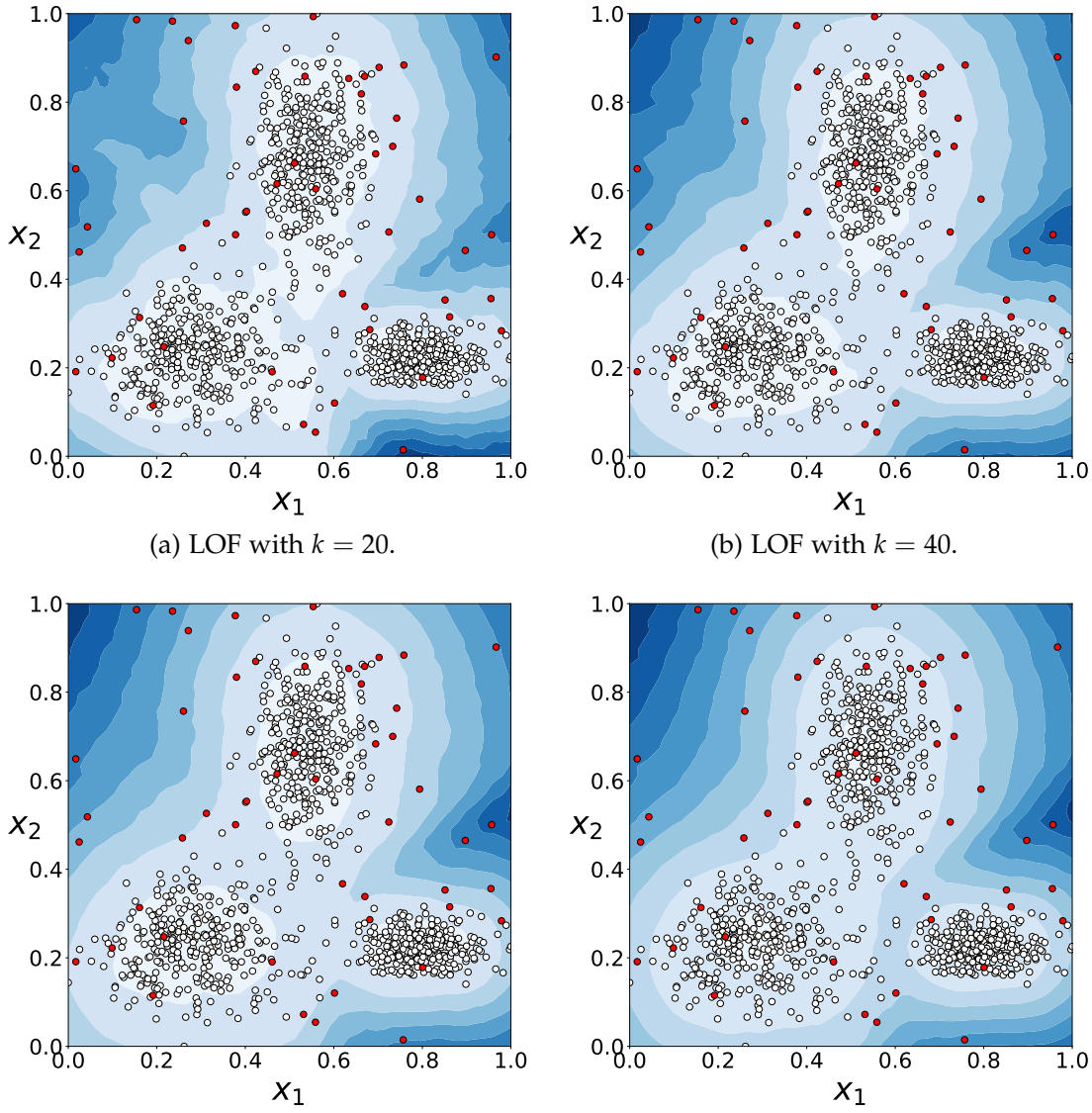


(a) P@n.



(b) Adjusted P@n.

Figure 4.7: P@n and Adjusted P@n for the LDF applied to dataset $\mathcal{D}_9$ (table 3.1). The high P@n values in the range $n \in [0, 100]$ and the Adjusted P@n curve, which attains a value of Adjusted P@n = 1 at $n \approx 100$, give information about a good outlier ranking and a good performance of the LDF for this dataset with respect to the corresponding parameter settings, e.g. the batch size is chosen to 50 and SGDE performs 10 refinement steps.



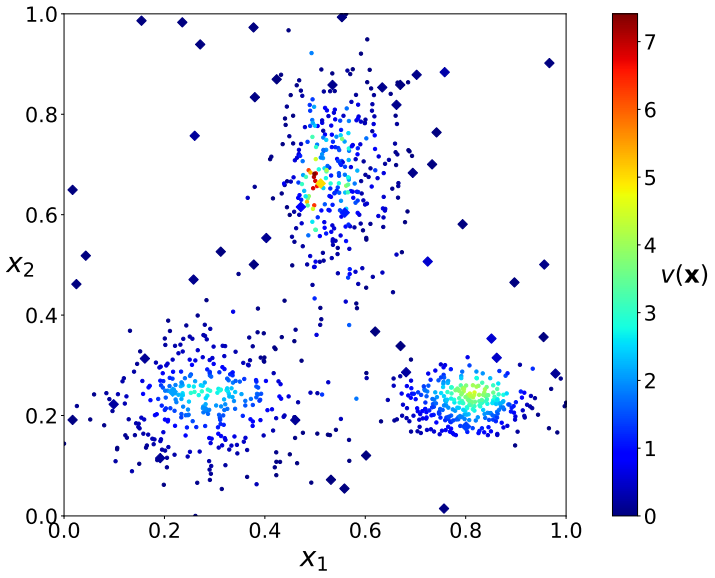
(a) LOF with $k = 20$.

(b) LOF with $k = 40$.

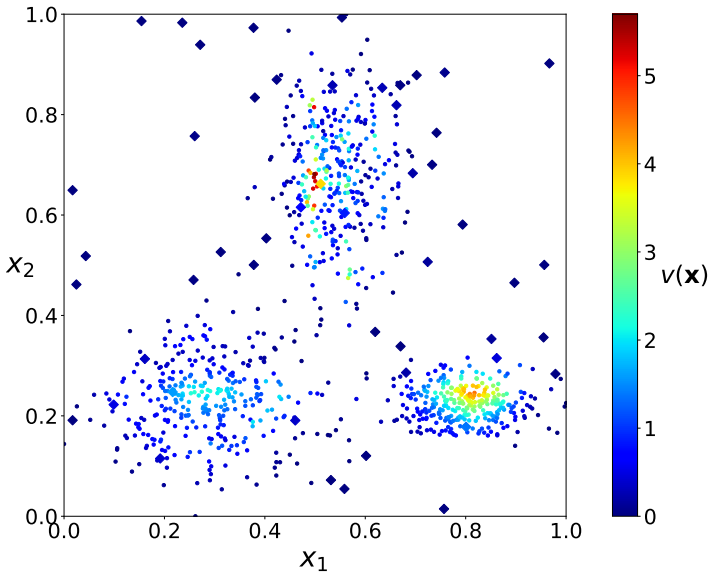
(c) LOF with $k = 60$.

(d) LOF with $k = 80$.

Figure 4.8: LOF of dataset $\mathcal{D}_5$ (table 3.1) with different values for k. White dots are true inliers and red dots are true outliers. The darker the blue background becomes, the higher the LOF value and the more likely is a data point an outlier.



(a) Based on DBSCAN with $Eps = 0.05$ and $MinPts = 10$.



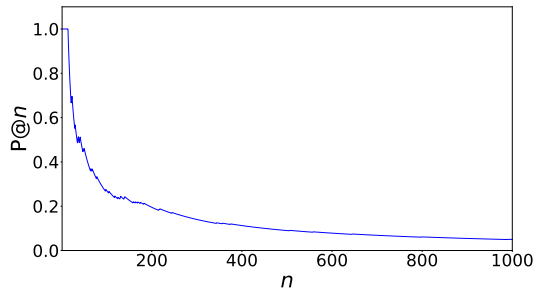
(b) Based on DBSCAN with $Eps = 0.05$ and $MinPts = 40$.

Figure 4.9: Comparison of the new approach applied to dataset $\mathcal{D}_5$ (table 3.1) based on different parameter settings for DBSCAN. True outliers are marked as big dots with diamond shape.

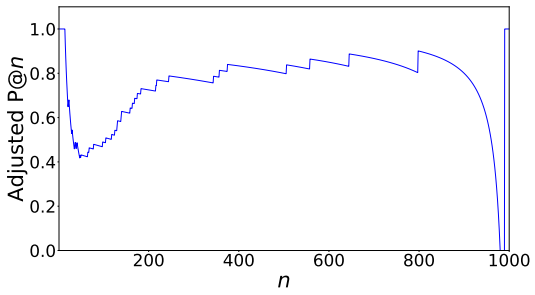
in figure 4.10 (a) and (b).

Figure 4.9 (b) shows the results of the new approach based on DBSCAN with $Eps = 0.05$ and $MinPts = 40$. Using this parameter setting, DBSCAN finds 3 clusters with an average Silhouette Coefficient of 0.342. Furthermore, it achieves a detection rate of DetectionRate = 0.86 and a false alarm rate of FalseAlarmRate = 0.303. Based on this results of DBSCAN, the new approach performs at an average precision of AP = 0.490, an adjusted average precision of 0.463 and ROC AUC = 0.883.

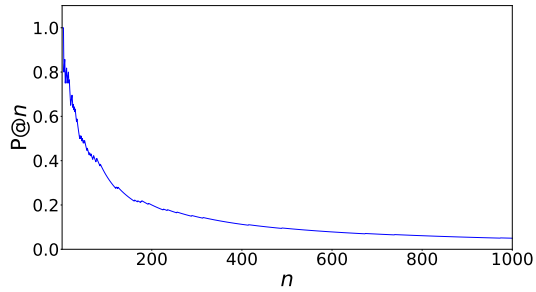
Comparing figure 4.9 (a) and (b), it can be seen that the clustering of DBSCAN has an influence on the result of the new approach. Figure 4.10 shows the evalutation of figure 4.9 (a) and (b). The result in figure 4.9 (b) is caused by a better outlier ranking because all outliers are detected at a lower rank n than in figure 4.9 (a).



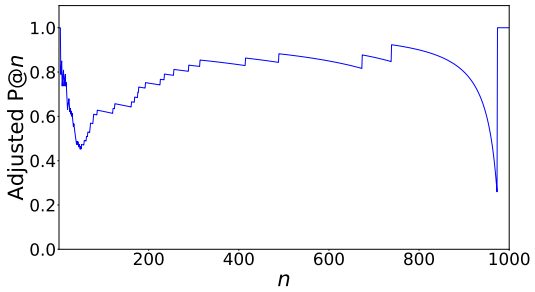
(a) P@n for figure 4.9 (a).



(b) Adjusted P@n for figure 4.9 (a).



(c) P@n for figure 4.9 (b).



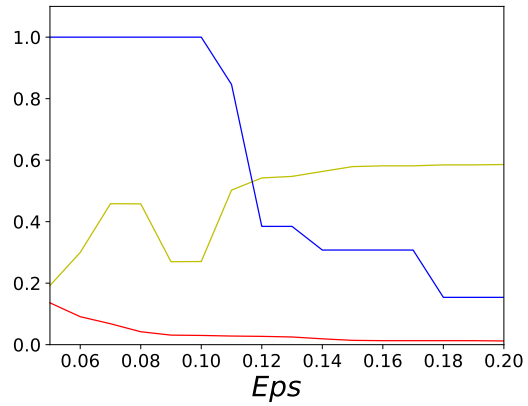
(d) Adjusted P@n for figure 4.9 (b).

Figure 4.10: P@n and Adjusted P@n curves for the new approach based on figure 4.9 (a) and (b). The decrease of both Adjusted P@n curves in the range of $n \in [800, 999]$ is caused by many normal data points which have lower outlier ranks than actual outliers. The result in figure 4.9 (b) has a better performance because all outliers are detected at a lower rank n, according to the Adjusted P@n curve in (d).

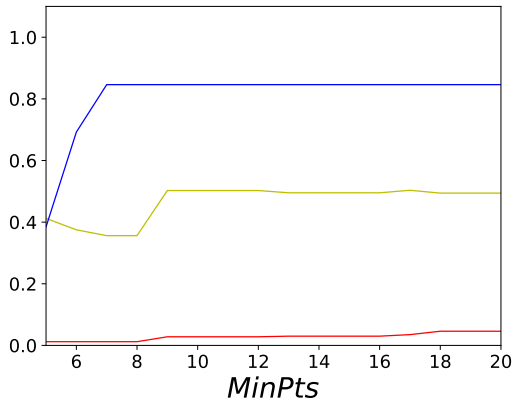
The Shuttle dataset in table 4.1 is a dataset for which the DBSCAN algorithm achieved

good results in comparison to other real datasets from Campos et al. [5], but in general the results are relatively bad. Figure 4.11 shows the comparison between the detection rate and false alarm rate for DBSCAN, LDF and the new approach applied to the Shuttle dataset.

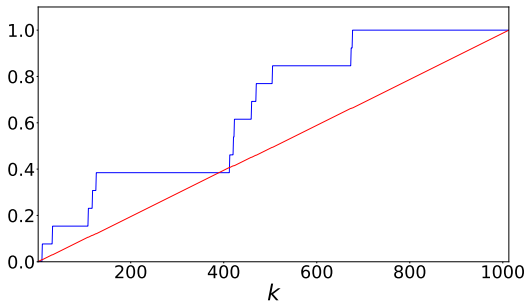
It turns out that the used real datasets perform relatively bad with density-based methods in general. One cause for this is the fact that most of the datasets contain only a small number of data points in contrast to a large number of dimensions. As a consequence, the few data points are widely spread over the data space which is one of the disadvantages of density-based methods. Furthermore, lots of the real datasets from Campos et al. [5] consider whole classes/clusters as outliers and for algorithms like DBSCAN or other density-based approaches, data points in a normal cluster cannot be detected as such.



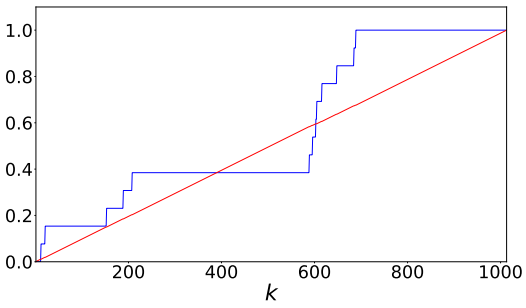
(a) DBSCAN depending on *Eps*.



(b) DBSCAN depending on *MinPts*.



(c) LDF.



(d) New approach.

Figure 4.11: Comparison between DBSCAN, LDF and the new approach applied to the Shuttle dataset (table 4.1). The detection rate (blue) and false alarm rate (red) for DBSCAN either depend on *Eps* or on *MinPts* and for LDF and the new approach it depends on the k most likely outliers, with respect to the ranking. Additionally, (a) and (b) show the avarage Silhouette coefficient (yellow) for the clustering performed by DBSCAN. It does not outgrow 0.6 and that means that only a moderate clustering is performed. The stepwise increasing detection rates in (c) and (d) tell that outliers are found in different steps. Values for k where the detection rate is constant are caused by the effect of misclassifying normal data points as outliers. This assumption is supported by linear increasing false alarm rates.

5 Conclusion

In this thesis, different outlier detection techniques and measures are presented. Some of these methods use SGDE as a metric. During the workflow of the thesis, these approaches were implemented in a framework which now can be used to detect outliers based on SGDE. Based on the results in chapter 4, the belief function approach can be used to obtain reasonable results, but they depend on the structure of the dataset. During the evaluation of the results, it turned out that LDF in combination with SGDE was the most efficient technique for finding outliers. Furthermore, the compution of the LDF for a dataset with 1000 data points was performed in a few seconds and reasonable values were received. Compared to LDF, the new approach also shows relatively good results. A very important point to keep in mind is that datasets with a high sparsity are not suitable for density-based outlier detection or clustering techniques. The real datasets used in the thesis are very sparse and therefore the corresponding results turned out be relatively bad. In general, for the artificial datasets it turned out, that outlier detection with SGDE is a good approach and it is worthwhile to put further effort into this approach. Additionally to the work done in this thesis, other clustering algorithms could be modified in a way to take advantage for outlier detection like the modification of the Silhouette Coefficient plot explained in section 3.3.5. For the future, it might be interesting to compute the new approach presented in this thesis with other methods than SGDE and DBSCAN, e.g. the density values could be calculated with LDE and a different algorithm than DBSCAN could be used to perform clustering. Future work with the same purpose as this thesis could also try lots of more different settings for SGDE and other methods to improve the results. Furthermore, the presented techniques can be applied to many other real datasets which are suitable for density-based methods.

List of Figures

Bibliography

[1] C. C. Aggarwal and P. S. Yu. "Outlier Detection for High Dimensional Data." In: *SIGMOD Rec.* 30.2 (May 2001), pp. 37–46. ISSN: 0163-5808. DOI: 10.1145/376284.375668.

[2] D. F. Andrews. "Plots of High-Dimensional Data." In: *Biometrics* 28.1 (1972), pp. 125–136. ISSN: 0006341X, 15410420.

[3] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander. "OPTICS: Ordering Points to Identify the Clustering Structure." In: *SIGMOD Rec.* 28.2 (June 1999), pp. 49–60. ISSN: 0163-5808. DOI: 10.1145/304181.304187.

[4] 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*. SIGMOD '00. Dallas, Texas, USA: ACM, 2000, pp. 93–104. ISBN: 1-58113-217-4. DOI: 10.1145/342009.335388.

[5] 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." In: *Data Mining and Knowledge Discovery* 30.4 (July 2016), pp. 891–927. ISSN: 1573-756X. DOI: 10.1007/s10618-015-0444-8.

[6] D. D. and S. S. Babu. "Methods to detect different types of outliers." In: *2016 International Conference on Data Mining and Advanced Computing (SAPIENCE)*. Mar. 2016, pp. 23–28. DOI: 10.1109/SAPIENCE.2016.7684114.

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

[8] D. M. Hawkins. "Introduction." In: *Identification of Outliers*. Dordrecht: Springer Netherlands, 1980, pp. 1–12. ISBN: 978-94-015-3994-4. DOI: 10.1007/978-94-015-3994-4_1.

[9] V. Hodge and J. Austin. "A Survey of Outlier Detection Methodologies." In: *Artif. Intell. Rev.* 22.2 (Oct. 2004), pp. 85–126. ISSN: 0269-2821. DOI: 10.1023/B:AIRE.0000045502.10941.a9.

[10] H.-P. Kriegel, P. Kröger, J. Sander, and A. Zimek. "Density-based clustering." In: *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1.3 (), pp. 231–240. DOI: 10.1002/widm.30. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/widm.30.

[11] L. J. Latecki, A. Lazarevic, and D. Pokrajac. "Outlier Detection with Kernel Density Functions." In: *Machine Learning and Data Mining in Pattern Recognition*. Ed. by P. Perner. Berlin, Heidelberg: Springer Berlin Heidelberg, July 2007, pp. 61–75. ISBN: 978-3-540-73499-4.

[12] C. .-.-. Lu, D. Chen, and Y. Kou. "Algorithms for spatial outlier detection." In: *Third IEEE International Conference on Data Mining*. Nov. 2003, pp. 597–600. DOI: 10.1109/ICDM.2003.1250986.

[13] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. "Scikit-learn: Machine Learning in Python." In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[14] B. Peherstorfer. "Model Order Reduction of Parametrized Systems with Sparse Grid Learning Techniques." Dissertation. Department of Informatics, Technische Universität München, Oct. 2013.

[15] D. Pflüger. *Spatially Adaptive Sparse Grids for High-Dimensional Problems*. München: Verlag Dr. Hut, Aug. 2010. ISBN: 9783868535556.

[16] P. J. Rousseeuw. "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis." In: *Journal of Computational and Applied Mathematics* 20 (1987), pp. 53–65. ISSN: 0377-0427. DOI: https://doi.org/10.1016/0377-0427(87)90125-7.

[17] Y. Xu, N. Xu, and X. Feng. "A New Outlier Detection Algorithm Based on Kernel Density Estimation for ITS." In: *2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. Dec. 2016, pp. 258–262. DOI: 10.1109/iThings-GreenCom-CPSCom-SmartData.2016.67.