

CHRISTIAN-ALBRECHTS UNIVERSITÄT  
ZU KIEL

DOCTORAL THESIS

---

**Advances in Unsupervised Learning and  
Applications**

**Background knowledge driven subspace clustering, semantic password  
guessing and learned index structures**

---

*Author:*

Maximilian HÜNEMÖRDER

*from*  
Munich

2023

*A thesis submitted in fulfillment of the requirements  
for the degree of Doctor of Engineering*

*in the*

Technische Fakultät

*Supervisor/1st Examiner:* Prof. Dr. Peer KRÖGER

*2nd Examiner:* Prof. Dr. Eirini Ntoutsis

*Tag der Mündlichen Prüfung:* 27.06.2023

# Declaration of Authorship

Ich, Maximilian HÜNEMÖRDER, erkläre, dass diese Arbeit mit dem Titel, "Advances in Unsupervised Learning and Applications" und die darin dargestellte Arbeit meine eigene ist. Ich bestätige, dass

- die Abhandlung, abgesehen von der Beratung durch die Betreuerin oder den Betreuer nach Inhalt und Form meine eigene Arbeit ist.
- die Arbeit noch nie ganz oder zum Teil schon einer anderen Stelle im Rahmen eines Prüfungsverfahrens vorgelegen hat, veröffentlicht worden ist oder zur Veröffentlichung eingereicht wurde, außer die Teile dieser Dissertation, welche vorherigen Publikationen entsprechen. Für diese ist mein Eigenanteil in dieser Arbeit angegeben.
- dass die Arbeit unter Einhaltung der Regeln guter wissenschaftlicher Praxis der Deutschen Forschungsgemeinschaft entstanden ist;
- mir kein akademischer Grad entzogen wurde.

**Unterschrift Doktorand:**

---

*Maximilian* HÜNEMÖRDER

**Datum:** Kiel, July 6, 2023

---



# Abstract

Maximilian HÜNEMÖRDER

*Advances in Unsupervised Learning and Applications*

Over the past few years, advances in data science, machine learning and, in particular, unsupervised learning have enabled significant progress in many scientific fields and even in everyday life. Unsupervised learning methods are usually successful whenever they can be tailored to specific applications using appropriate requirements based on domain expertise. This dissertation shows how purely theoretical research can lead to circumstances that favor overly optimistic results, and the advantages of application-oriented research based on specific background knowledge. These observations apply to traditional unsupervised learning problems such as clustering, anomaly detection and dimensionality reduction. Therefore, this thesis presents extensions of these classical problems, such as subspace clustering and principal component analysis, as well as several specific applications with relevant interfaces to machine learning. Examples include password guessing using semantic word embeddings and learning spatial index structures using statistical models. In essence, this thesis shows that application-oriented research has many advantages for current and future research.



# Abstract

Maximilian HÜNEMÖRDER

*Advances in Unsupervised Learning and Applications*

In den letzten Jahren haben Fortschritte in der Data Science, im maschinellen Lernen und insbesondere im unüberwachten Lernen zu erheblichen Fortentwicklungen in vielen Bereichen der Wissenschaft und des täglichen Lebens geführt. Methoden des unüberwachten Lernens sind in der Regel dann erfolgreich, wenn sie durch geeignete, auf Expertenwissen basierende Anforderungen an spezifische Anwendungen angepasst werden können. Diese Dissertation zeigt, wie rein theoretische Forschung zu Umständen führen kann, die allzu optimistische Ergebnisse begünstigen, und welche Vorteile anwendungsorientierte Forschung hat, die auf spezifischem Hintergrundwissen basiert. Diese Beobachtungen gelten für traditionelle unüberwachte Lernprobleme wie Clustering, Anomalieerkennung und Dimensionalitätsreduktion. Daher werden in diesem Beitrag Erweiterungen dieser klassischen Probleme, wie Subspace Clustering und Hauptkomponentenanalyse, sowie einige spezifische Anwendungen mit relevanten Schnittstellen zum maschinellen Lernen vorgestellt. Beispiele sind das Erraten von Passwörtern mit Hilfe semantischer Worteinbettungen und das Lernen von räumlichen Indexstrukturen mit Hilfe statistischer Modelle. Im Wesentlichen zeigt diese Arbeit, dass anwendungsorientierte Forschung viele Vorteile für die aktuelle und zukünftige Forschung hat.



# Acknowledgements

First and foremost, I would like to thank my PhD supervisor, Prof. Dr. Peer Kröger, for supervising my thesis, Prof. Dr. Eirini Ntoutsi, who kindly agreed to review my thesis and everyone on the examination committee. I would also like to thank all the people at LMU and CAU that I met along the way, especially the following people: Dani and Anna for always being there to hang out with me and develop new clustering algorithms, Julian for giving me a new perspective on subspace clustering research, Nanni for literally travelling to the other side of the world with me, and Miri, Andrea, Andreas, Max, Claudius, and Thomas for their support along the way. I would also like to thank all my co-authors, especially Theresa Ullmann, Terézia Slanínáková, Levin Schäfer, and Michael Eichberg. Finally, I owe a lot to my family, my parents, who helped me to pursue an academic career, and of course Rike, who is my reason to keep going everyday and who came with me to the other side of Germany to find a new home in Kiel.



# Contents

<b>Declaration of Authorship</b>	<b>iii</b>
<b>Abstract</b>	<b>v</b>
<b>Acknowledgements</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Structure of the Thesis and Contributions . . . . .	1
1.2 The Paradigms of Learning . . . . .	3
<b>2 Unsupervised Learning</b>	<b>5</b>
2.1 Feature Selection, Extraction and Representation Learning . .	5
2.2 Clustering . . . . .	7
2.2.1 Partition- and Model-based Clustering . . . . .	7
2.2.2 Density-Based Clustering . . . . .	9
2.2.3 Subspace Clustering . . . . .	9
2.3 Anomaly Detection . . . . .	14
2.4 Problems of Unsupervised Learning . . . . .	15
<b>3 Applications</b>	<b>17</b>
3.1 Learned Index Structures . . . . .	17
3.2 Vector Semantics and Password Guessing . . . . .	19
3.3 Reinforcement Learning . . . . .	21
<b>4 On coMADs and Principal Component Analysis</b>	<b>23</b>
<b>5 CODEC - Detecting Linear Correlations in Dense Clusters using coMAD-based PCA</b>	<b>33</b>
<b>6 SIDEKICK: Linear Correlation Clustering with Supervised Background Knowledge</b>	<b>39</b>
<b>7 Implicit Hough Transform Neural Networks for Subspace Clustering</b>	<b>49</b>
<b>8 OAB - An Open Anomaly Benchmark Framework for Unsupervised and Semisupervised Anomaly Detection on Image and Tabular Data Sets</b>	<b>59</b>
<b>9 Over-optimistic evaluation and reporting of novel cluster algorithms: an illustrative study</b>	<b>71</b>

<b>10 Towards a Learned Index Structure for Approximate Nearest Neighbor Search Query Processing</b>	<b>101</b>
<b>11 SePass: Semantic Password Guessing using k-nn Similarity Search in Word Embeddings</b>	<b>111</b>
<b>12 Stirring the Pot - Teaching Reinforcement Learning Agents a Push-Your-Luck board game</b>	<b>129</b>
<b>13 Conclusion and Future Work</b>	<b>135</b>
<b>Bibliography</b>	<b>137</b>

# Chapter 1

## Introduction

Das Leben ist zu kostbar, um es  
dem Schicksal zu überlassen.

---

Walter Moers  
*Die 13 1/2 Leben des Käptn Blaubär*

Data Science, Machine Learning, and Artificial Intelligence are no longer just sub-disciplines of Computer Science and Statistics. These topics have become ubiquitous and are ultimately permeating every part of our daily lives and society itself. Businesses are thriving by working with big data and data science; social media is fighting fake news with AI; journalists can work with natural language models to break down complex issues into understandable written text; artists can use generative models for inspiration and create new, unseen works of art; humanities scholars can now use AI to explore topics that data science has not been applied to before. The fourth paradigm of science [37] has become a new paradigm for society as a whole.

By its very nature, computer science is the practical application of mathematics to the real world. Data science, on the other hand, as an extension of both computer science and statistics, can be proven by application alone, similar to other engineering fields. However, it also draws on a deep theoretical background. As a result, data science research can make significant scientific advances while solving real-world problems, and research can benefit from looking at its applications. The focus of this thesis is a comprehensive study of various methods from unsupervised learning and ML in general. This thesis analyses in depth a number of applications and problems in the design of such methods.

### 1.1 Structure of the Thesis and Contributions

This thesis is structured in a top-down manner, as illustrated in Figure 1.1. The remainder of this Chapter deals with an overview of the different learning paradigms of data science and machine learning. Chapter 2 will then focus on unsupervised learning and a survey of the main unsupervised tasks, namely feature learning, clustering, and anomaly detection. Next, Chapter 2 Section 2.4 looks at issues concerning unsupervised learning and acts as a link to Chapter 3, where two application domains for unsupervised learning,

i.e., learned index structures and semantic word embeddings, are introduced. Chapter 3 Section 3.3 then concerns reinforcement learning. Chapters 4 – 12 collect a series of published papers, which constitute the following main contributions of this thesis:

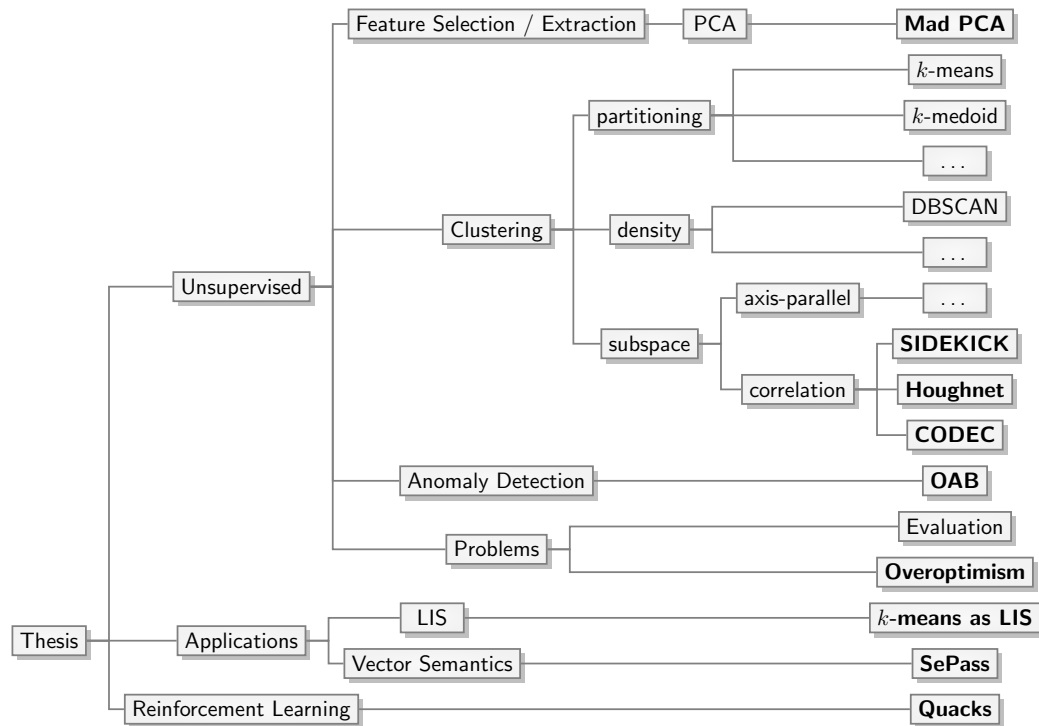


FIGURE 1.1: Overview of the structure of this thesis. Contributions are highlighted in Bold text.

- A novel robust extension of Principal Component Analysis (Chapter 4) and a clustering algorithm based on this extension (Chapter 5)
- Two novel subspace clustering algorithms; one based on the use of supervised background knowledge (Chapter 6) and a second one based on Hough Transform Networks (Chapter 7)
- A comprehensive benchmark suite for anomaly detection experiments (Chapter 8)
- An examination of overoptimism in clustering algorithm research and development (Chapter 9)
- Three applications of unsupervised learning; exploring the design of a learned index structure for Nearest Neighbor Queries (Chapter 10), a novel way of learning to guess passwords using semantic word embeddings (Chapter 11) and a new task for reinforcement learning based on a board game (Chapter 12)

Finally, Chapter 13 will then provide a conclusion and future research directions that can evolve from the work introduced by this thesis.

## 1.2 The Paradigms of Learning

Since there are many definitions of machine learning paradigms and data science tasks, the goal of this section and the following chapter is to provide descriptions for all tasks relevant to this thesis. Machine learning is commonly divided into three main paradigms: supervised, unsupervised, and reinforcement learning. In addition, there are many subcategories that do not fit neatly into one of these categories: There is semi-supervised learning [22], self-supervised learning and transfer learning [34] to name a few.

The supervised learning paradigm encompasses all data science tasks where instances have been labeled a priori, usually by experts in the corresponding domain. The goal of any such a supervised learning task is then to find a statistical process that models the relationship between these labels  $Y = (y_1, \dots, y_n)$  and their corresponding data instances  $X = (x_1, \dots, x_n)$ , or more specifically the model estimates the conditional probability density  $Pr(Y|X)$ . [36, p. 485]

Thus, supervised learning can be thought of as "learning with a teacher", where the model is the student and the teacher is a set of labeled training data. Depending on the complexity of the problem, such training data usually needs to consist of a large number of  $x$  and  $y$  pairs in order to learn to make correct predictions. The performance of the model is then evaluated on a previously unseen data set, the test set, to ensure generalization and avoid overfitting. The main supervised tasks are classification and regression, depending on whether the labels provided are qualitative or quantitative. Algorithms can range from simple methods such as k-nearest neighbors and support vector machines to complex methods such as deep neural networks.

Unsupervised learning, i.e., learning without a teacher, deals with unlabeled data. An in-depth survey of unsupervised learning tasks will be provided in the next Chapter, c.f. Chapter 2

While supervised and unsupervised tasks are easily categorized using labeled or unlabeled data, reinforcement learning does not fit into either category. Instead of learning from labels, i.e., knowledge about the data, reinforcement learning is about learning from experience and is inherently tied to its applications. Therefore, reinforcement learning will be introduced and further discussed in Chapter 3 Section 3.3.



## Chapter 2

# Unsupervised Learning

Unsupervised learning aims at finding patterns in data without supervision, i.e., without labels. The main unifying factor of all tasks, that can be categorized as unsupervised learning, is, therefore, to find an approximation that is as close as possible to the underlying distribution of the data. Given a data set  $X$ , the goal is to find the probability density  $Pr(X)$ <sup>1</sup>. [36, p. 485 – 487]

This means that unsupervised learning depends heavily on assumptions about the data set and the task. Assumptions, such as the model complexity<sup>2</sup>, the density of the distribution, the number of clusters, or more complex ideas like the relationship between features, for example, the relationship between individual pixels in an image. These assumptions usually vary depending on the task at hand.

The following sections introduce the main unsupervised tasks relevant to the publications included in this thesis: Feature selection and extraction, clustering and anomaly detection. However, the main focus of this chapter is on clustering, and in particular subspace clustering, as these tasks are covered by the majority of the publications included in this thesis. The final section of this chapter, then deals with the problems of evaluation and research of new unsupervised methods.

## 2.1 Feature Selection, Extraction and Representation Learning

One of the fundamental preprocessing tasks for any data science method is Feature Selection or Dimensionality Reduction since many downstream tasks – both supervised and unsupervised – require well-chosen or automatically derived features. While it is usually a reasonable choice to use hand-crafted features derived from expert domain knowledge, the existence of such features cannot always be assumed. In addition, the task may be too complex to be fully understood, there may be experts with multiple differing opinions, or the data set may simply be too large, for example, when working with a text corpus sourced from the world wide web.

---

<sup>1</sup>In contrast to supervised learning, where the distribution of  $X$  is usually not as important as the relationship between  $X$  and the corresponding labels  $y$ .

<sup>2</sup>It could be a simple linear model or a deep neural network with many layers.

In a supervised setting, features can be transformed in a way as to fit the ground truth and the current task. In an unsupervised setting, however, where no ground truth is available, features must be selected according to assumptions about the underlying distribution of the data. These features are often assumed to be not a subset of the original attributes but a linear or nonlinear combination of different attributes. In general, when working with linear combinations of features, Principal Component Analysis (PCA) is still a widely used strategy, mainly because of its simplicity. While there are a variety of nonlinear methods such as Isomap [79], T-SNE [56], and self-supervised representation learning methods, such as autoencoders and generative models, each method, including PCA, has different desirable properties and there is no one-size-fits-all solution to feature selection.

PCA is a mature multivariate data analysis method that dates back to the work of Karl Pearson in 1901 [29]. While extensively studied by statisticians in the last century, it is still relevant in the advent of data science and big data, presumably because it is one of the most popular basic methods for feature transformation and reduction. The goal of PCA is, given a dataset  $X$  in  $\mathcal{R}^d$ , to find a set of orthogonal components  $V$  that constitute a linear approximation of  $X$ . To quote Pearson directly, its purpose is to

*«represent a system of points in plane, three, or higher dimensioned space by the "best-fitting" straight line or plane».* [29, p. 559]

These components are aligned along the directions of the highest variance and ordered from highest to lowest. The components can then be used to reveal linear dependencies, indicate hidden correlations, transform the data, or perform dimensionality reduction by selecting only the components representing the highest variance and discarding the rest.

The components correspond to an affine hyperplane of rank  $r \leq d$ , where  $r$  is the number of components considered. PCA can then be performed using a least-squares optimization minimizing the reconstruction error between a projection matrix  $P = V_r V_r^T$  and  $X$ . However, it is usually solved by computing the eigenvectors and values of the covariance matrix  $C$  or, less commonly, the correlation matrix of the data set. Another widely used method for PCA is Singular Value Decomposition (SVD), as the two are mathematically closely related. [76]

One of the main problems with this basic version of PCA is that it is very sensitive to outliers. A single extreme outlier can strongly influence the resulting components, even if most other instances are distributed in a different direction. This is due to the fact that PCA minimizes the mean squared error, which is based on the mean and is, therefore, strongly influenced by extreme values. This motivated the research presented in Chapter 4. Our goal was to find a robust way to perform PCA that relies on only minor modifications to the classical algorithm. Based on a mathematical paper introducing coMAD, we provide a robust variant of PCA using the comedian or coMAD matrix [48], a median-based alternative to covariance. We then factorize this comedian matrix to find robust principal components analogous to covariance-based PCA. We show that this relatively simple concept can greatly reduce the influence of outliers on the resulting components.

## 2.2 Clustering

Clustering means partitioning a data set into sets of similar instances with respect to a chosen similarity measure while separating dissimilar instances. The resulting subsets are called clusters. There are many subcategories of clustering algorithms, depending on assumptions about the data set, i.e., the similarity measure used, the type of data, and other factors. While there are many taxonomies of clustering algorithms, the following categorization of algorithms is relevant in the context of this thesis:

- **Partition-based Clustering** assumes a predetermined number of clusters  $k$  and a objective function.
- **Model-based Clustering**, is similar to partition-based clustering but with the additional assumption that a specific distribution generated the clusters.
- **Density-based Clustering** has no fixed number of clusters but instead requires that clusters are areas of high density seperated by areas of low density.
- **Subspace Clustering** combines feature extraction with clustering to find clusters in distinct lower-dimensional subspaces.

This list leaves out some popular categories since these are not directly related to the publications provided by this thesis, e.g., hierarchical clustering, MeanShift, or Deep Clustering [83]

### 2.2.1 Partition- and Model-based Clustering

$K$ -means is arguably the most popular clustering task and can be solved by different algorithms, such as the homonymous  $k$ -means algorithm. Given a data set  $X \in \mathbb{R}^d$  and a target number of clusters  $k$ , the goal of  $k$ -means is to find a set of centroids  $m$  and the corresponding assignment vector  $A$  such that, the objective function

$$\mathcal{L}_{\text{kmeans}} = \sum_{i=1}^n \|x^i - m_{A^i}\|^2 \quad (2.1)$$

is minimal with respect to the combination of centroids and the assignments,

$$\arg \min_{m,A} \mathcal{L}_{\text{kmeans}} \quad (2.2)$$

This problem is NP-hard, even considering 2-dimensional datasets [82] and optimizing  $\mathcal{L}$  is not convex.

The  $k$ -means problem is typically solved by the  $k$ -means algorithm, i.e., alternating between assigning each instance to its nearest centroid and finding the optimal centroids for the corresponding cluster members until convergence. It can be shown that the  $k$ -means algorithm converges in finite steps,

since both moving the centroids and reassignment lead to a lower  $\mathcal{L}$  and there is only a finite number of possible assignments.

The main problem of the  $k$ -means algorithm is that it is not guaranteed to find a global minimum, instead resulting in local minima depending on the initialization of the centroids. There are several ways to extend  $k$ -means to find a global minimum. Simply running multiple iterations with different initializations and using either the  $k$ -means loss or the silhouette coefficient to compare results and select an initialization can accomplish this. Additionally, Kmeans++ [9] is a popular method to initialize the centroids not randomly but swayed in such a way that they are spread out in the space the data inhabits.

$K$ -means can also be solved using gradient descent. This may seem redundant at first glance since the traditional  $k$ -means algorithm provably converges. Still, gradient descent versions can be used for end-to-end differential algorithms in order to work with complex data types, such as images. Regularization terms can be easily added to the basic  $k$ -means algorithm to adapt the cluster definition to fit specific tasks. There are three basic variants: Batch [17], Stochastic and Minibatch [75]  $k$ -means. In addition,  $k$ -means can also be seen as a matrix factorization problem [12]. This means that the matrix product  $A \times C$  of an assignment matrix  $A$  and the centroids  $C$  become a representation of the data  $X$ , where each point is represented by its nearest centroid. Then using the mean squared error of the difference between  $A \times C$  and  $X$  as a Loss:  $\mathcal{L} = MSE(A \times C, X)$ , this can then be solved by gradient descent. In any case, the assignments for  $k$ -means are hard assignments. Other methods that allow for soft assignments or introduce Gaussian mixture models can be categorized as model-based clustering.

While the  $k$ -means objective function assumes Euclidean space, other partitioning algorithms work with various non-euclidean spaces.  $K$ -medoid algorithms, for example, can be applied to euclidean data as well as to arbitrary data. The  $k$ -medoid problem consists of finding cluster centroids, which are themselves instances of the data set. Consequently, this works with any data as long as a distance measure can be applied. A popular  $k$ -medoid algorithm is PAM [67], or Partitioning Around Medoids, which solves the problem using a greedy approach. There are also CLARA [47, p. 126] and CLARANS [63, 64] as extensions of PAM. CLARA performs multiple runs of PAM on different samples of the data set. In each run, CLARA finds the medoids for a sample of the data set using PAM, and then assigns the rest of the data set to these medoids. In the end, the result with the smallest average distance to the medoid is selected. CLARANS, on the other hand, searches the entire data space and takes samples at each step. In addition, there is some fairly recent work on speeding up PAM<sup>3</sup> by Schubert et al. [73, 72].

Another issue with  $k$ -means style algorithms is that minimizing the sum of squares error of each centroid and its corresponding cluster members involves the assumption that each cluster is spherical. This can be remedied by

<sup>3</sup>and thus CLARA and CLARANS

algorithms that combine multiple centroids – in a fashion similar to hierarchical clustering – to obtain arbitrarily shaped clusters, or by using a different notion of clustering, such as density-based clustering.

### 2.2.2 Density-Based Clustering

Density-based clustering algorithms work under the assumption that clusters are areas of high density separated by areas of low-density occurrences of data instances. This density is usually described as the ratio of the number of objects to the distances between the objects in a cluster. The most common definition comes from the popular DBSCAN [28] algorithm.

Given a minimum number of points  $m$ , and a range  $e$ , core points of clusters are all points that have at least  $m$  neighbors in an  $e$ -neighborhood. These core points are then *directly-density-reachable*. Additionally, all points that are part of such a core neighborhood but themselves do not have  $m$  directly density-reachable instances are called border points and are only *density-reachable*. The advantage of this definition is that the number of clusters does not have to be fixed, and the clusters can have any shape. In particular, DBSCAN can also detect noise points, which has to be done in an extra step for  $k$ -means and other clustering algorithms.

A disadvantage is that the hyperparameters  $m$  and  $e$  need to be set, which are not as immediately obvious for domain experts as the number of clusters. Epsilon primarily depends on the distance metric used. OPTICS [7] and HDBSCAN [21] eliminate the need to set the  $e$  parameter by using a hierarchical clustering approach. In addition, there are reasonable heuristics for choosing the hyperparameters, as described, for example, by Schubert et al. [74]

While there are other algorithms such as DenPeak [68] and DenMune [1] that work with different definitions of density, only the definition from DBSCAN is relevant in the context of this thesis.

### 2.2.3 Subspace Clustering

Clustering high-dimensional data is a complex task, because there are many problems that are connected to high dimensionality. These problems are colloquially joined under the term "*The Curse of Dimensionality*". Richard Bellman, a mathematician often credited with first using the term, put it this way:

« [...] *what casts the pall over our victory celebration? It is the curse of dimensionality, a malediction that has plagued the scientists from earliest days.* » [14, p. 94]

Inspired by the work of Kriegel et al. [53, 52], this curse can be decomposed into several subproblems, that are important in the context of clustering:

- **Problem 1 - Optimization Complexity**

This first sub-problem corresponds to Bellman's original definition of the curse, which describes a combinatorial problem that arises when brute forcing an optimization task. Inherently, any search task becomes more complex with each additional feature, since each feature adds an additional degree of freedom, i.e., exponentially more possible samples to the search space. [25, p. 5]

- **Problem 2 - Sparsity and Distance Measures**

As the dimensionality increases, we need more samples to "fill" a space. If we increase the dimensionality while keeping the same number of instances, the data becomes sparse. The problem is that this leads to exponentially larger distances between instances, while the relative pairwise distances tend to be the same for all samples.

- **Problem 3 - Noise Features**

Given the sparsity from **Problem 2**, we can expect that in very high-dimensional data sets, the amount of irrelevant or noisy features should also increase.

- **Problem 4 - Feature Correlations**

Similar to **Problem 3**, we can assume that many dimensions allow many possible correlations between the features; depending on the tasks, any of these correlations could be relevant to domain experts. Finding these relevant correlations therefore becomes more complex with increasing dimensionality.

**Problem 1** is the original definition of the Curse of Dimensionality. It describes one of the main motivations behind machine learning, which is to find simple descriptions of complex data in order to compress the dimensionality of a high-dimensional data set. It tells us that as dimensionality increases, it becomes infeasible to simply materialize the function, so machine learning allows us to find a compressed function that easily fits into memory. In supervised learning, this also means that the higher the dimensionality of a data set, the more training samples are needed to achieve good generalization.<sup>4</sup>

**Problem 2** is particularly problematic for any nearest neighbor based algorithm, since if all pairwise distances become nearly equal, then every sample likely is the nearest neighbor of every other sample.

A common way to deal with **Problem 3**, i.e., a large number of noise features, is to apply feature reduction before clustering. But while this may solve parts of the curse, this global reduction can still remove relevant local feature correlations, which may be crucial for certain applications, see **Problem 4**. The field of subspace clustering attempts to find such relationships by combining feature extraction and clustering into a single task, i.e, not only grouping the data into distinct clusters, but also assigning a linear subspace to each cluster.

---

<sup>4</sup>This is of course also true for unsupervised learning, where the probability density function underlying the data can be better estimated the more samples we have generated by it.

A linear subspace can be defined as a set of vectors which is a subset of a space  $\mathcal{R}^d$  and has the following three attributes: (i) The set must contain the corresponding zero vector. (ii) the set must be closed under addition. (iii) The set must be closed under scalar multiplication. This means, for example, that a 2d hyperplane in  $\mathcal{R}^3$  intersecting the origin spans a linear subspace because it contains the zero vector, i.e., the origin  $(0, 0, 0)$ , and every vector lying on the hyperplane. Of course, subspaces can also intersect, which usually results in a lower-dimensional subspace. For example, two 2d hyperplanes intersect in a one-dimensional subspace.

### Axis-Parallel Subspace Clustering

This definition of subspace clustering implies an infinite number of subspaces in any space  $\mathcal{R}^d$ , which makes a naive approach, i.e., searching each subspace for each cluster, infeasible. A subset of algorithms therefore works with axis-parallel subspaces, i.e., the set of subspaces of vectors where at least one dimension is always zero (or a constant number), e.g.  $\{(x_1, x_2, 5) | x \in \mathcal{R}^3\}$ . In  $\mathcal{R}^d$  this results in  $O(2^d)$  possible axis-parallel subspaces. This is also well in line with **Problem 3**, which suggests that noise features are in the original feature set and can be learned to be ignored. Algorithms are then designed to search this reduced set of subspaces in either a top-down or bottom-up fashion. Top-down variants attempt to cluster the entire feature space and then find subsets of dimensions that fit these clusters. Bottom-up approaches usually start by finding clusters in individual dimensions and then combine the dimensions in some way. These methods can then reduce the search space by using a variant of the a priori property from frequent item set mining.

Examples of algorithms designed to find axis-parallel subspaces are:

- **CLIQUE** [6], the first bottom-up axis-parallel subspace clustering algorithm, works with an equal-sized grid<sup>5</sup>. First, subspaces<sup>6</sup> containing clusters are identified by finding dense cells using a method with a slight similarity to DBSCAN. Given a hyperparameter  $t$ , they select only cells where the ratio of the number of instances in the cell divided by the number of total instances is greater than  $t$ . This is repeated in a bottom-up fashion, first for each single dimension, then for subspaces consisting of a pair of dimensions, then for triples, and so on, while constantly pruning those subspaces that contain smaller ones that do not count as clusters, in the manner of the Apriori algorithm for frequent item set mining. The found cells and corresponding subspaces are then collected into clusters using a graph search algorithm that searches for connected components of a graph of subspace cells, where each node

<sup>5</sup>While the main content of the original CLIQUE paper presents a version of CLIQUE where the cell size is the same for each dimension, the authors mention that individual cell sizes are also possible, although the density threshold  $t$  will need to be adjusted

<sup>6</sup>a subspace consisting of a single hyper rectangle, i.e., a grid cell, or a collection of such cells that are later merged into clusters

is a cell and edges connect those cells that belong to the same subspace cluster.

- **ENCLUS** [23] is similar to CLIQUE but uses entropy to find interesting subspaces.
- **MAFIA** [33] is an extension of CLIQUE that replaces the equispaced grid with an adaptive grid.
- **SUBCLU** [46] is a density-based axis-parallel subspace clustering algorithm that works in a bottom-up fashion. The authors show that the definition of density-connectedness from DBSCAN [28] is monotonous with respect to subspaces, i.e., if two points in subspaces A and B are density-connected, then they are also connected in AB as well, and conversely, if they are not density connected in either A or B, then the same is true for AB. This again allows for pruning similar to the Apriori algorithm.
- **PROCLUS** [4], the first top-down axis parallel subspace clustering algorithm, is an extension of CLARANS from Section 2.2.1 and therefore based on  $k$ -medoid clustering.

### Arbitrarily Oriented Subspace Clustering

Another way to navigate the infinite possible subspaces to find relevant arbitrarily oriented subspace clusters is to focus on relevant correlations in subsets of the data. According to **Problem 4** of the Curse, the more features a data set consists of, the higher the probability that some of them are correlated. While global correlations can be found using either statistical measures or dimensionality reduction methods such as PCA, local feature correlations can be found by looking only at subsets of the entire data set, i.e., correlation clusters.

In data mining, finding such clusters is usually referred to as correlation clustering to distinguish it from axis-parallel clusters, which do not imply any correlation between the points of the cluster.<sup>7</sup> On the other hand, in various research fields, especially in computer vision, the term subspace clustering itself is equivalent to correlation clustering.<sup>8</sup> Therefore, this thesis will use the terms correlation and subspace clustering interchangeably. Thus, subspace clustering is the task of finding arbitrarily oriented linear subspace clusters, which may be affine depending on the intended task. An affine subspace has the same properties as a linear subspace except that it does not have to include the zero vector.

<sup>7</sup>Correlation clusters, of course, only indicate correlation; whether or not there is actual causation must be determined by a domain expert. Also, since arbitrarily oriented subspaces could theoretically be perfectly parallel to the original axes, correlation clustering includes all axis-parallel subspace clusters.

<sup>8</sup>which is reasonable, since the mathematical definition of a linear subspace corresponds to an arbitrarily oriented subspace, and axis-parallel subspaces are a special case, as mentioned above. Moreover, the term *correlation clustering* is already overloaded, since it also refers to a task from graph mining, cf. [10].

Examples of correlation clustering algorithms are:

- **ORCLUS** [5], based on PROCLUS, consists of applying  $k$ -means with a large  $k$ , then calculating the PCA for each resulting cluster and merging clusters if they belong to similar subspaces. These three steps are repeated until a predetermined number of clusters  $k$  is returned.
- **4C** [15] is a correlation clustering method based on DBSCAN. A cluster is defined as a correlation-connected set if the objects in the clusters are density-connected in terms of DBSCAN and exhibit a similar linear correlation. The algorithm works analogously to DBSCAN, only checking for correlation reachability instead of density reachability. To compare the correlation, 4C utilizes a distance measure that is closely related to the Mahalanobis [57] distance, only using an inverted covariance matrix instead.
- **CASH** [3] relies on the Hough transform, a transformation of the original data space into a parameter space, to find dense grid cells in the resulting parameter space. These dense clusters then correspond to linear subspaces in the original data space.
- **$k$ -planes** [18] works very similar to  $k$ -means, only using hyperplanes as centroids for a cluster instead of the corresponding means.

Many correlation clustering methods, especially those from the data mining community, are based on performing PCA at a local cluster level. An advantage of such a PCA-based method is that it is relatively easy to derive quantitative models, that might help explain the correlations found. Based on the paper introducing this concept [2], Chapter 6 of this thesis provides *SIDEKICK* an algorithm that extends this work to find correlation clustering using a minimal amount of supervised background knowledge.

Another approach, Sparse Subspace Clustering (SSC) [27, 26], comes from the computer vision community. It exploits the so-called self-expressiveness property, i.e., any object lying in a linear subspace can be expressed by a linear combination of other objects from the same subspace. The first step of SSC is to find an affinity matrix that shows the similarity between all objects in the database based on whether they belong to the same linear combination. In the second step, this affinity matrix has to be separated into clusters using a different clustering algorithm to obtain a subspace clustering. Typically, spectral clustering is used. A disadvantage of SSC is that the affinity matrix depends on the size of the data set, which means that SSC does not scale well. However, it handles noisy data well by applying a sparsity constraint to the objective function.

In Chapter 7, this thesis provides a novel subspace clustering algorithm based on Houghnets [11]. It is designed to be a stepping stone between the computer vision and data mining approaches for subspace clustering. This approach uses a loss that essentially fits  $k$  hyperplanes in such a way that each object is at least considered to be part of one of the hyperplanes using soft assignment, similar to a model-based approach. We show that our

method is more robust against outliers than other correlation clustering algorithms while running much faster than SSC. Additionally, this work compares some of the different subspace clustering algorithms, which were mentioned above.

## 2.3 Anomaly Detection

The overall performance of any data science task can be affected by anomalies and outliers in the data set. These are objects or groups of objects that are very far from the *normal*<sup>9</sup> rest of the data, in one way or another. These could be noise measurements from a power surge in an instrument or random snapshots when working with image data, for example. These anomalous data objects can then affect statistical measures in unwanted ways, such as the mean, and consequently other measures that depend on the mean, such as PCA, see Section 2.1 in this chapter. One way to deal with such outliers consists of extending any existing algorithm to ignore them. However, generic anomaly detection methods can remove these objects before solving the original task. Although anomaly detection is traditionally thought of as an unsupervised task, it is often not so easy to pigeonhole it into a single learning paradigm. Anomaly detection can be unsupervised, semi-supervised, or supervised, depending on the exact application.

Unsupervised anomaly detection is often referred to as outlier detection. Since this is an unsupervised task, there are no labels for normal or anomalous data. Consequently, there must be some assumption about the nature of such an outlier. For example, the notion of what constitutes an outlier can be based on density, as in the case of the Local Outlier Factor [19]. Another way to define outlier detection is based on a contamination parameter  $\epsilon$ , for example this can be done when using Isolation Forests [54]. The task then becomes finding the  $\epsilon$  percent of instances that are least likely to have been generated by the same statistical process as the normal part of the data. In a real-world application, this would mean that to be extra sure of production quality, the worst 10% of items produced on a production line would be discarded, or at least sorted out for more detailed evaluation.

Supervised anomaly detection is a highly unbalanced classification problem, where a small class of instances is labeled as anomalous and a large class of instances is labeled as normal. By using some way to deal with this imbalance, any classification algorithm can be used to classify objects as either normal or anomalous.

---

<sup>9</sup>The definition of what data is normal is generally not definitive. For example, normal data can be defined by similarity, i.e., instances that are close together versus insular objects that are far from all others, or it can be determined by frequency of occurrence, i.e., events that occur frequently versus events that occur only once or very rarely. But sometimes certain rare circumstances also can be known to be normal. So usually the notion of normality should be defined depending on the application and domain expertise.

Finally, semi-supervised anomaly detection<sup>10</sup>, is probably the setting that shows up most often in real-world applications. Here, the input to the algorithm is a set of objects labeled as normal data, but no objects labeled as anomalous. The task is to find a function that describes the normal data as well as possible. Then, any unknown samples that do not fit this learned function well can be labeled as potential outliers. An example of such a semi-supervised algorithm is the one-class SVM [71]. Using the same industrial example as before, it would be easy to keep track of the *normal* items during production, since these are only the ones that are shipped and not returned, a simple method to generate labels for the normal data. Then an algorithm trained on these normal objects can sort out any articles that are not similar, and the resulting normal articles could be used to have more data to feed the algorithm.

These examples show that anomaly detection is inherently closely related to its real-world application. To facilitate research on anomaly detection in a real-world setting, we introduce a benchmark framework *OAB*, Chapter 8, that allows users to quickly evaluate newly designed anomaly detection algorithms and compare them to related algorithms on tabular and image data.

## 2.4 Problems of Unsupervised Learning

When researching unsupervised algorithms, especially clustering, one often encounters problems with the evaluation of novel algorithms. There are generally two types of evaluation measures for clustering, internal and external, and both suffer from similar problems. Internal measures, such as the silhouette coefficient, are often implicitly or explicitly optimized by the objective functions of clustering algorithms, and will therefore overestimate the performance of the corresponding algorithms. As a result, while these measures can adequately show how different variations, initializations, or hyperparameter settings of the same algorithm perform in comparison to each other, they are not as well suited for comparing different algorithms.

External measures, conversely, work with a priori labels and therefore seem to be better for comparing different clustering algorithms. However, using a single labeling for a data set can also have the same problem as an internal measure, since this single partitioning may fit a certain assumption behind a chosen algorithm better than another label concerning a different aspect of the data. For example, *k*-means assumes that there are *k* clusters in the data set and that their compactness can be reasonably minimized by Least Square Loss (LSL). However, instances in real-world data sets may have different connotations, and therefore different possible groupings, depending

---

<sup>10</sup>Semi-supervised anomaly detection is sometimes referred to as novelty detection, although novelty detection can also refer to whether the anomalies are already in the data set or whether only new data points are being analyzed

on the application. Reducing this to a single class labeling and then designing clustering algorithms based on it seems like a long and complex form of supervised learning.<sup>11</sup>

In Chapter 9, in order to illustrate such problems, we took a simple published clustering algorithm, ROCK [13], and tried to implicitly overestimate its performance by using hyperparameter optimization strategies to fit the optimal data set to the clustering algorithm instead of changing the design of ROCK. We have shown that it is relatively easy for a researcher to inadvertently find a version of a data set in which the chosen algorithm outperforms all competitors. This can lead to publications that overestimate the performance of a clustering algorithm that may work on a theoretical data set, but do not hold up in real-world applications. This means that it is beneficial to combine the application and research of unsupervised learning methods. The next chapter discusses a collection of such possible application areas.

---

<sup>11</sup>An extreme example of this problem is the MNIST dataset and similar image datasets regularly used in deep clustering research. The commonly used labeling allows only a single interpretation of the data set, grouping by the number shown. It is quite easy to imagine a different labeling based on handwriting style.

## Chapter 3

# Applications

One solution to the over-optimism described at the end of the last Chapter is to design unsupervised algorithms specifically for a particular application. Specific application domains and data types allow for strong assumptions, which is especially important for unsupervised learning. For this reason, unsupervised methods have been successful with data types such as image, text, or graph data. This chapter highlights some interesting application domains and the contributions of this thesis to them.

### 3.1 Learned Index Structures

Database Indexing is one of the standard topics in data mining research. However, the community has recently been shaken by the introduction of learned index structures (LIS). The concept of LIS was introduced by Kraska et al. [51] in 2018. The authors state that, from a certain perspective, database index structures can be viewed as statistical models. Database indexes are data structures designed to speed up database queries. Different types of index structures are better at speeding up different types of queries. However, different data distributions can also affect query execution speed. While traditional methods take this into account to some degree – for example, B-trees can have smaller leaf nodes in places where indexes are closer together – improving or replacing them with a machine learning model can help indexes fit the intended dataset more accurately. While there are speed improvements over other traditional index structures, the main advantage is the smaller model size, since smaller index structures lead to faster database management systems [49, 58, 31].

In general, an index structure is designed to work on a set of distinct keys  $S \subset \mathcal{U}$ , where  $\mathcal{U}$  is usually the set of natural numbers but may also be the set of possible strings or dates, for example, as long as the index values can be ordered. An index structure is required to be able to perform various queries using  $S$ : Usually, these are membership queries:  $member(x) = \text{TRUE}$  if  $x \in S$  else  $\text{FALSE}$ , i.e., whether the key exists in the set and range queries:  $range(x, y) = S \cap [x, y]$ , i.e., return the set of possible keys between  $x$  and  $y$ , that exist in the index set  $S$ . [31, 66]

A trivial solution to perform these queries is a binary search or a binary search tree (BST). The B-tree, an extension of BSTs, is still a commonly used index structure for DBMSs. While in a BST each node can have only two

children, a B-tree allows more than two children, thus reducing the height of the resulting search tree. The leaf nodes of a B-tree are all at the same level and contain the actual keys. The combination of these two attributes makes B-trees well suited for indexing, since finding a particular leaf node usually means following a single path through the tree. Both reducing the height and keeping all leaf nodes at the same level reduce the length of this path and therefore improve query execution speed.

To solve this indexing problem with a learned model, one can use the  $rank(x)$  function, i.e., the number of elements of  $S$  smaller than  $x$ . We can now learn a function  $f(x)$ , that approximates  $rank(x)$ , utilizing a supervised regression model with the sorted array  $S$  as the input and  $rank(x)$  as the labels and minimizing the absolute training error  $\mathcal{L} = \sum_{i=0}^n |f(x_i) - rank(x_i)|$ . The membership query then becomes  $member(x) = \text{TRUE}$  if  $S[f(x)] = x$  and the range query becomes finding  $f(x)$ , i.e., finding the approximate rank  $f(x)$  and then iterating over  $S$ , until  $y$  is found. [31, 66]

Note that this LIS can be easily extended so that it is not approximate by using the training error. This is because the train set, i.e., the collection of keys on which the LIS is trained, is the same as the test set, i.e., the set of keys on which the LIS is later queried. This means that the training errors can be used to guarantee that the data is within the predicted rank plus error range. This usually significantly smaller range can then be searched using binary search, for example.

While this task is straightforward when considering an auto-incremented index attribute with no missing keys – then the solution is as simple as using  $rank(x)$  as the index itself – LIS are able to outperform traditional index structures if the keys are of a different data type, e.g., the timestamp of the entry. There might be more import dates, i.e., on some days there might be a lot of entries and on other days not so many. An example might be an incident log for a support ticket system.

Typically, one would not use a single model for an index, but rather a tree of models, i.e., ensemble learning using a mixture of experts.

Kraska et al. [38] present two other examples of one-dimensional indices that can be replaced or extended by different types of ML models:

- **Hashmaps**, which are typically used for point queries, can also be replaced by a piecewise linear model and even avoid collisions, sometimes better than traditional hash functions, although the authors found that this is only possible if the model can *overfit*<sup>1</sup> the data [69]
- **Bloom filters** are designed for membership queries only, i.e., to decide whether a key exists. False positives are possible. False negatives, however, are not. This can be improved by using a binary classifier before applying the Bloom filter to filter out even more candidates. While the

<sup>1</sup>Since the train and test set are the same, overfitting is not always a bad thing concerning LIS

classifier will produce both false positives and false negatives, the resulting false negatives can be filtered again by the bloom filter. Subsequent papers [61, 81] add multiple steps of bloom filters and classifiers to improve the performance.

All of these examples are one-dimensional index structures. However, multidimensional indexes are needed when performing spatial queries, such as nearest neighbor queries. In this case, the previously discussed methods do not work because these methods rely on ordered keys, and for indexes of more than one dimension, there is usually no natural order. Traditional examples of index structures that solve this spatial data problem include R-trees [35], Z-orderings and vector quantization [70], and Locality Sensitive Hashing [32], among others.

Some related work has been published on learning multidimensional index structures, such as Flood [62]. However, these approaches usually produce a lot of overhead, so in the paper presented in chapter 10, we show that a multidimensional index structure can be as simple as using k-means and a classifier in combination. We use *k*-means to partition the data and a classifier to predict the corresponding cluster containing the query object. Our approach is similar to the learned metric index of Slaninakova et al. [8, 77], published independently around the same time.

Finally, index structures are not the only building block of a database management system that could be replaced by a statistical model. The field of instance optimization explores self-tuning DBMSs and is an exciting glimpse into the future of database management systems. [50]

## 3.2 Vector Semantics and Password Guessing

It has been over 100 years since Markov introduced n-gram models and the concept of treating language as a statistical process. The advent of deep learning and massive text corpora built from the World Wide Web allows for using the same concept to mathematically encode the meaning of words into continuous vector spaces, where semantically similar words have similar representations. Now, anyone can generate almost too convincingly human-sounding text with projects like ChatGPT<sup>2</sup>. [45, p. 55, 56]

Two words are usually considered semantically similar because they are often associated in some way. However, semantic similarity does not mean that two words have the same meaning. While there are **synonyms**, i.e., two words that mean the same, other associations include antonyms, i.e., words that have opposite meanings like *hot/cold*, words from the same semantic field, i.e., concerning a similar topic, for example, *cat* and *dogs*, **hypernyms**, i.e., one word is an umbrella term that includes the other one, like *animal* and *cat*, words that have the same **connotation**, i.e., positive, negative, or any kind of sentiment, i.e., happy or sad, and many more. [45, p. 105]

In 1957 Osgood et al. [65], while studying the different connotations of words, asked people to rate words in three different aspects and noted that

---

<sup>2</sup><https://openai.com/blog/chatgpt/>, visited 12.01.2023

this resulted in each word having a three-dimensional vector representation. The combination of Osgood’s intuition and the concept that words are semantically similar depending on how often they are used together, and vice versa, gave rise to vector semantic models. These continuous vector spaces, which combine such seemingly diametrical concepts of vector mathematics and word semantics, are called word embeddings. [45, p. 106]

However, in order to create a word embedding one does not always need to create a survey and ask random people to annotate each word. Using a co-occurrence matrix instead, such as a term-document matrix, one can use unsupervised<sup>3</sup> learning methods to learn semantic vector representations for each word. A term-document matrix column contains one column per distinct word that occurs anywhere in a corpus<sup>4</sup> and one row for each document in the corpus. This usually results in sparse data since corpora typically contain tens of thousands of different words, and corpora trained from the Internet typically contain millions of web pages, i.e., documents.

A common baseline model to turn this sparse matrix of ones and zeros into a proper vector space is tf-idf, which is short for *term-frequency-inverse-document-frequency*, a method of scoring a word based on its total number of occurrences relative to the number of documents in which it appears. However, tf-idf still results in a sparse representation, and embedding models usually aim to find a dense vector representation of the co-occurrence matrix. In addition, to using a simple matrix factorization, popular embedding models include:

- **word2vec** [59] a method that, given a target word, uses a sampling strategy to create positive examples from words that occur in the same context and negative examples from the rest of the vocabulary, and then trains a logistic regression model that aims to maximize the similarity of the positive word pairs and minimize the similarity of the target word and the negative examples. The weights of this model can then be used as embeddings.
- **Fasttext** [16, 43, 44], a similar algorithm to word2vec, which uses smaller components of words, so-called n-grams, instead of complete words. This allows fasttext to create representations for out-of-vocabulary words.

A common feature of these methods is that they produce static embeddings, i.e., each word has a single vector representation. However, with more modern techniques, e.g. BERT [24], it is possible to find context-specific word embeddings, i.e., a word can have multiple representations depending on the context.

Based on the concept of unsupervised semantic word embeddings, this thesis contributes a novel method for password guessing in Chapter 3.2.

---

<sup>3</sup>Or more specifically self-supervised, i.e., using a supervised method on an unsupervised task, by using the input itself as a label.

<sup>4</sup>A text corpus is a collection of text documents that is used to analyze or learn from a particular language, multiple languages, or particular dialects

While the field of password guessing has already been combined with natural language models, our approach is the first to use static word embeddings to find unknown passwords based on their semantic similarity to a leaked password list.

### 3.3 Reinforcement Learning

Reinforcement Learning is the third primary learning paradigm because it is neither supervised nor unsupervised. Instead, it focuses on learning from interacting with an environment. In essence, an agent or agents interact with an environment and perform actions to maximize some reward that is returned when an action is submitted to the environment. The core of the reinforcement learning problem is to define this reward. Therefore, an obvious application of reinforcement learning is board- and video game AI, since there are usually well-defined rules, obvious winners, clear winning conditions, and obvious rewards for actions. In addition, the number of possible actions to take at any given time is usually relatively small, while the number of possible game states is often large. However, reinforcement learning has also been successfully applied outside of the context of gaming, for example, to find a novel way to compute matrix multiplication, which requires fewer steps than any previously human-designed method [30]. A more detailed review of reinforcement learning algorithms is beyond the scope of this thesis. Instead, see *Reinforcement Learning: An Introduction* by Richard Sutton et al. [78].

In chapter 12 we provide a novel task for reinforcement learning in the form of a popular German board game. It is a challenging task for reinforcement learning methods because it offers a large number of different rule combinations, which usually do not adapt well to changes in the environment. We provide an environment for training reinforcement learning algorithms and show that it is possible to learn a reasonable strategy for a single rule combination using reinforcement learning. In the future, we aim to add support for different rule combinations.



## Chapter 4

# On coMADs and Principal Component Analysis

This chapter consists of a preprint version of the following publication:

Daniyal Kazempour, Maximilian Archimedes Xaver Hünemörder, and Thomas Seidl. “On coMADs and Principal Component Analysis”. In: *Similarity Search and Applications - 12th International Conference, SISAP 2019, Newark, NJ, USA, October 2-4, 2019, Proceedings*. Ed. by Giuseppe Amato, Claudio Genaro, Vincent Oria, and Milos Radovanovic. Vol. 11807. Lecture Notes in Computer Science. Springer, 2019, pp. 273–280. DOI: 10.1007/978-3-030-32047-8\_24. URL: [https://doi.org/10.1007/978-3-030-32047-8\\_24](https://doi.org/10.1007/978-3-030-32047-8_24)

It was reproduced with permission from Springer Nature.

**Statement of Originality:** The idea of using an existing method, the CoMAD [48], and combining it with PCA was conceptualized by D. Kazempour. D. Kazempour and **M. Hünemörder** discussed the idea and **M. Hünemörder** wrote software and performed experiments based on preliminary work by D. Kazempour. The concept and results were discussed with T. Seidl and then written up by D. Kazempour and **M. Hünemörder**.

- **Conception:** Kazempour (Lead), **Hünemörder** (Support)
- **Planning:** Kazempour (Lead), **Hünemörder** (Lead), Seidl (Support)
- **Execution:** **Hünemörder** (Lead), Kazempour (Support)
- **Manuscript:** Kazempour (Lead), **Hünemörder** (Lead), Seidl (Support)

## On coMADs and Principal Component Analysis

Daniyal Kazempour, M.A.X. Hünemörder, and Thomas Seidl

Ludwig-Maximilians-Universität München, Munich, Germany  
{kazempour, huenemoerder, seidl}@dbs.ifi.lmu.de

**Abstract.** Principal Component Analysis (PCA) is a popular method for linear dimensionality reduction. It is often used to discover hidden correlations or to facilitate the interpretation and visualization of data. However, it is liable to suffer from outliers. Strong outliers can skew the principal components and as a consequence lead to a higher reconstruction loss. While there exist several sophisticated approaches to make the PCA more robust, we present an approach which is intriguingly simple: we replace the covariance matrix by a so-called coMAD matrix. The first experiments show that PCA based on the coMAD matrix is more robust towards outliers.

**Keywords:** Covariance · coMAD · Principal Component Analysis.

### 1 Introduction

When dealing with vast amounts of data and a large number of features performing principal component analysis (PCA)[5] is a common approach. PCA yields the principal components, i.e. the directions of highest variance in the data. Furthermore, PCA can be used to reveal hidden correlations and is sometimes used to detect arbitrary oriented linear correlated clusters. For example it is used in correlation clustering algorithms like 4C[2] or ORCLUS[1]. However PCA based on the covariance matrix is highly sensitive towards outliers, particularly strong ones, can have an impact on the resulting principal components. This is due to the fact that outliers can influence the mean for each of the features of a data set. That is why in statistics the median is used as a robust measure against outliers. For the measure of dispersion of a feature the so called median absolute deviation from the median, short MAD, is the method of choice. In this work we propose to use a coMAD matrix instead of a covariance matrix on which the eigenvalues and eigenvectors are computed. We will elaborate on the MAD and the coMAD matrix in detail. In our first tests it can be seen that with heavy noise the principal components of a PCA are heavily deflected, while those resulting from a PCA based on the coMAD remain stable.

### 2 Related Work

There are many approaches to make the PCA more robust towards noise. With the term robust, we understand the following: Given data where a significant

2 D. Kazempour, M.A.X. Hünemörder and T. Seidl

amount of the objects exhibits a linear correlation and some objects are outliers. The method is considered as robust, if the increasing number as well as increasing distance of outliers, does not significantly affect the direction of computed principal components in comparison to the case where there would be no outliers in the data set. In the work of [6] the authors develop a theory of Robust Principal Component Analysis (RPCA) and describe a robust M-estimation algorithm for capturing linear multivariate representations of high dimensional data, exemplary on images. M-estimators are a class of extremum estimators which can be regarded as a generalization of maximum-likelihood estimation. The authors further state that while methods such as RANSAC and Least Median Squares are more robust compared to M-estimation, it is not clear how to apply the techniques efficiently on high-dimensional data. In another work [4] the authors propose the ROBPCA method which combines the concept of a so-called 'projection pursuit' with a robust scatter matrix estimation on which the eigenvectors and eigenvalues are computed. Their method relies on several criteria and definitions. They use e.g. for the computation of outlierness the so called Stahel-Donoho affine-invariant outlyingness. They further compute a reweighted mean and covariance matrix based on the Rousseeuw and Van Driessen consistency factor. In another work [7] a generative RPCA model is proposed which relies on the Bayesian framework in which data noise is modelled as a mixture of Gaussians (MoG).

While all of the mentioned methods rely on various more complex and sophisticated methods, we challenge the task of robust PCA by asking: What if we exchange the covariance matrix against a coMAD matrix? Since the simple median and MAD is robust against outliers, the coMAD and MAD-PCA should be, too. In the following section we first define the coMAD and contrast it against the covariance. Then we support our claim in the experimental evaluation section, in which we also elaborate briefly on a metric we use to measure the robustness of our method in which we compare the reconstruction errors from a coMAD-based PCA against the covariance-based solution. We later on critically question the appropriateness of the MAE evaluation. Our approach is based on the MAD and the comedian as defined in the work of [3]. However we have taken the liberty of renaming the comedian to coMAD since the covariance is traditionally not named the 'comean'.

### 3 MAD and coMAD

Given a data matrix  $D$  where each of its rows represents a data record and its columns represent the features  $(A_1, \dots, A_d)$ . The first step of performing PC is computing the covariance matrix  $\Sigma$ , which is defined as:

$$\Sigma_D = \begin{pmatrix} \text{var}(A_1) & \cdots & \text{cov}(A_1, A_d) \\ \vdots & \ddots & \vdots \\ \text{cov}(A_d, A_1) & \cdots & \text{var}(A_d) \end{pmatrix}$$

With covariance being

$$\text{cov}(A_i, A_j) := E(((A_i - E(A_i))(A_j - E(A_j))))$$

For the case that  $A_i = A_j$  it is defined as the variance  $\text{var}(A_i) = E((A_i - E(A_i))^2) = E((A_j - E(A_i))^2) = \text{var}(A_j)$ . The covariance is thus a generalization of the variance. At this point it can be seen that from each of the features the expected value  $E$  (mean) is subtracted. The mean however is sensitive towards outliers which skew the mean value significantly. A more robust measure is the median. The analogon to the variance is the median absolute deviation from the median (MAD) which is defined as:

$$\text{mad}(A_i) = \text{med}(|A_i - \text{med}(A_i)|)$$

We can now generalize MAD like the covariance is a generalization of the variance. Then the coMAD can be defined as:

$$\text{com}(A_i, A_j) := \text{med}((A_i - \text{med}(A_i))(A_j - \text{med}(A_j)))$$

Like  $\text{cov}(A_i, A_j)$ ,  $\text{com}(A_i, A_j)$  is also a measure of covariance. Therefore, building on the definition of the coMAD, we can define a coMAD matrix  $A$ :

$$A_D = \begin{pmatrix} \text{com}(A_1, A_1) & \cdots & \text{com}(A_1, A_d) \\ \vdots & \ddots & \vdots \\ \text{com}(A_d, A_1) & \cdots & \text{com}(A_d, A_d) \end{pmatrix}$$

Now we can perform PCA using  $A_D$  instead of  $\Sigma_D$ , i.e. the eigenvalues and their corresponding eigenvectors of the coMad-matrix are computed.

While we shall see later in the experimental section the effects of noise on a covariance and a coMAD-based PCA, we further ask in this work-in-progress if there is a way to quantify the quality of the different methods. For this purpose we shall elaborate first on what happens after the eigenpairs (eigenvector, eigenvalue) of a PCA are computed. The eigenvectors are put into an eigenvector matrix  $U$  where each column corresponds to an eigenvector. What we do now is to use the eigenvectors with the  $k$ -largest eigenvalues, where  $k$  is widely noted in the literature as the number of principal components which cover more than 85% of the variance as a rule of thumb. The percentage of variance in a given dataset is explained by the following ration for each of the eigenvalue  $\lambda_i$ :

$$\varphi(k) = \frac{\sum_{i=1}^k \lambda_i}{\sum_{i=1}^d \lambda_i}$$

We discard as an example the second principal component of a performed PCA, which yields  $U_{k=1}$ . We project now the data  $D$  down to a lower dimensional representation  $Y$  through:

$$Y = D \cdot U_{k=1}$$

4 D. Kazempour, M.A.X. Hünemörder and T. Seidl

If we now want to reconstruct the data back to its original two-dimensional representation we achieve this with:

$$Z = Y \cdot U_{k=i}$$

This procedure of projecting the data ( $D$ ) down to a lower dimension ( $Y$ ) and reconstructing ( $Z$ ) it gives us the opportunity to compute the Mean Absolute Error (MAE) which is defined as:

$$MAE(D, Z) = \frac{\sum_{i=1}^n |d_i - z_i|}{n}$$

, where  $d_i \in D$ ,  $z_i \in Z$  and  $n$  denoted the number of objects for which holds  $n = |D| = |Z|$ .

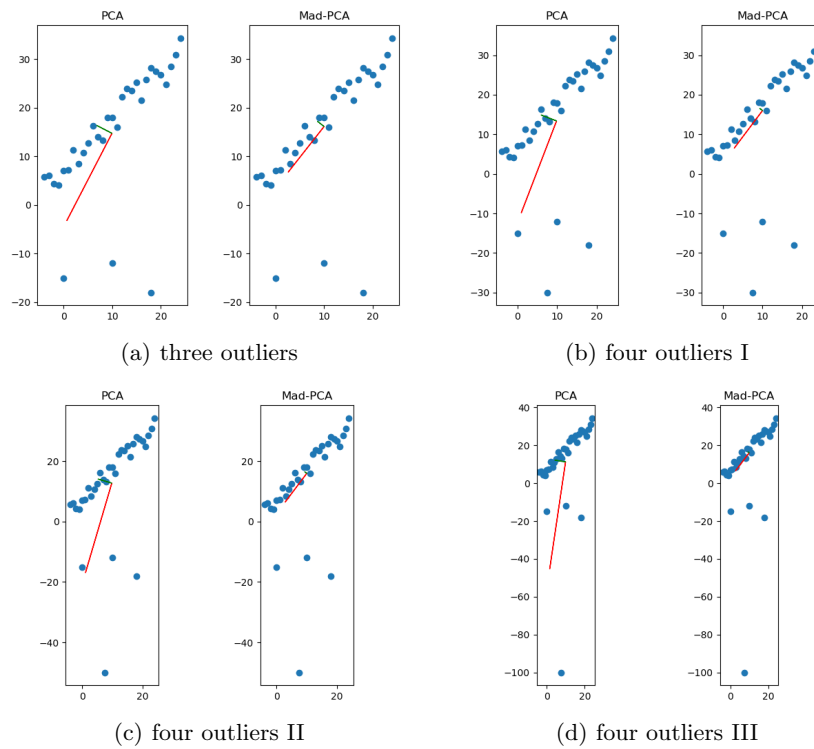
## 4 Experiments and Discussion

To provide an intuition, we apply this method on the following toy examples as seen in Figure 4<sup>1</sup>. In Figure 4 (a) we have a data set with a subset of objects which clearly exhibits a linear correlation. To this data set we added three outliers. On the left side it can be observed that the principal components of a PCA using the covariance matrix are deflected towards the direction of the three outliers. In contrast the principal components from the coMAD-based PCA remain with barely noticeable deflection in the direction of the objects belonging to the linear correlation. If we add now a fourth outlier which is even more apart, one can observe in Figure 4 (b) that the deflection of the principal components of a covariance-based PCA increases, exhibiting an by far larger eigenvalue. The coMAD variant remains again barely affected. In Figure 4 (c) and (d) we increase the distance of the fourth outlier, being more distant to the other outliers as well as to the linear correlated objects. In Figure 4 (d) the deflection is massive in the case of the covariance-based PCA, while principal components of the coMAD variant remain robust. These simple synthetic experiments reveal that a coMAD-based PCA excels regarding robustness against noise.

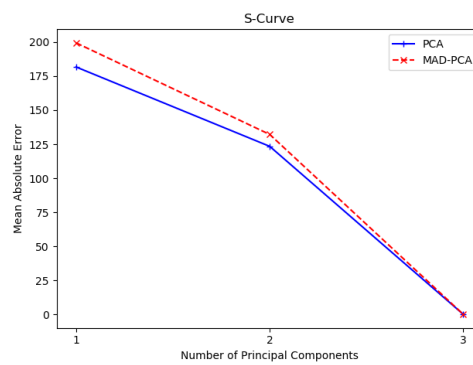
In a next step we conduct experiments on different data sets comparing the resulting MAE for the covariance and for the coMAD variant. For each of the data sets we repeat the MAE computation for choosing a range from  $k = 1, \dots, k = d$  principal components. The reason for this approach is twofold: first, one can observe how the MAE decreases per data set by increasing the number of principal components. Second: one can observe certain number of eigenvectors which result in an increase or decrease of the MAE. For all the experiments we used the data sets offered by the sklearn<sup>2</sup> library.

<sup>1</sup> We used python with several libraries. The code can be found at: <https://github.com/huenemoerder/MAD-PCA>

<sup>2</sup> <https://scikit-learn.org/stable/modules/classes.html#module-sklearn.datasets>



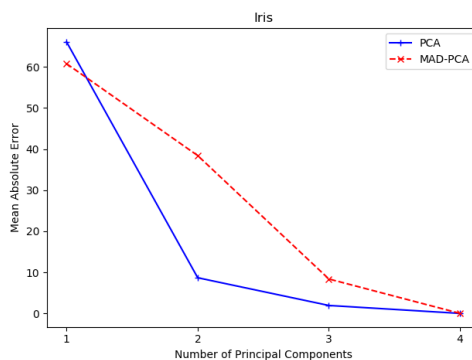
**Fig. 1.** PCA using covariance vs. coMAD matrices



**Fig. 2.** MAE with increasing number of principal components on the synthetic s-curve data set. orange: coMAD variant; blue: covariance variant

6 D. Kazempour, M.A.X. Hünemörder and T. Seidl

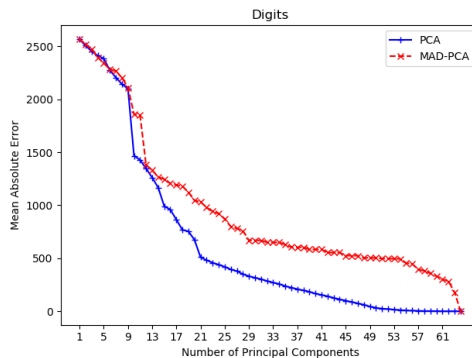
We begin as a first experiment on a synthetic data set. We generated an s-curve (50 samples, 3 features, 20 gaussian noise, random state = 42) without outliers. By that we have a null-case where we would not expect the coMAD to surpass the covariance approach at all. This data set has been chosen for its inherently non-linear shape. In Fig.2 we see on the horizontal axis the number of principal components and on the vertical axis the MAE. The coMAD approach is mostly the same like the covariance method, since we do not have outliers but just an increase of noise. However taking the first two principal components, the MAE is by around 25 units lower for the covariance method compared to the coMAD.



**Fig. 3.** MAE with increasing number of principal components on the iris data set. orange: coMAD variant; blue: covariance variant

The second experiment is conducted on the iris data set (450 samples, 4 features). In Fig.3 it can be seen significant differences in the MAE between the covariance and coMAD PCA variants. Taking the first principal component yields an lower MAE for the coMAD variant compared to the covariance version. For the second principal component we get already a visible difference where the covariance excels in comparison to the coMAD-based PCA. However, the improvement of the MAE from the covariance approach reduces drastically with the third principal component. It remains for future work to further investigate the reasons for why in the coMAD yields a lower MAE for the second principal component and a higher for the third PC compared to the covariance-based approach.

Since iris is small with regards to number of features as well as number of samples, we move in our next experiment to a larger scale. We test now both approaches on the pendigit data set which represents handwritten digits containing over 1797 samples in total, and 64 features. The results can be observed in Fig.4 . Here we observe that the MAE of the coMAD variant is marginally



**Fig. 4.** MAE with increasing number of principal components on the digits data set. orange: coMAD variant; blue: covariance variant

higher compared to the covariance version. This observation raises several questions: 1. Is the coMAD PCA inferior to the covariance PCA? The answer is: it depends. In the synthetic experiments one could clearly observe that the coMAD based method is superior. It is resilient towards outliers. But why do we get worse MAE results compared to the covariance variant? If we think of what happens in the case of the reconstruction error we recognize the following: It turns out that the MAE is lower for the covariance based approach since we obtain principal components which are heavily deflected. While this deflection is bad, since it means that it does not represent the direction of linear correlated objects in the data set, it is good for the reconstruction, since it means that the deflected principal component minimizes the distance from the outliers to the principal components, and as such, minimizes the error. The observed behaviors regarding the reconstruction are therefore expected. However, we may question at this point whether the MAE itself is a good measure for our purpose. The MAE is suitable if we want to evaluate of how well the detected principal components minimize the distance to each data object in the data set. However, it does not reflect how well the principal components maintain a small distance of the objects belonging to a linear correlation.

## 5 Conclusion and Future Work

In this work-in-progress we have presented the coMAD in context of PCA. Based on the idea that a median is more robust compared to mean we defined a coMAD matrix on which a PCA is performed. The idea is intriguingly simple compared to competing methods. Experimental results show that the coMAD approach yields principal components which seem unaffected by any noise, while the covariance-based PCA experiences heavy deflection of its resulting principal components.

The potential of the coMAD for PCA may reveal in future work, when for correlation clustering methods such as 4C the coMAD is used instead of the

8 D. Kazempour, M.A.X. Hünemörder and T. Seidl

covariance. Especially in the 4C scenario where for each data object within an  $\varepsilon$ -radius a PCA is conducted, we have a small sample size of objects, making it especially prone to just a small number of noise. It also remains to future work to further study those cases where the coMAD approach delivers higher as well as a lower MAE compared to the covariance variant. Detecting and characterizing those cases opens the scene for the development of novel approaches dealing with such special cases. Further it remains of special interest to develop criteria to evaluate the quality of a coMAD-based PCA against a covariance variant, since the MAE does not satisfy the task of determining the resilience of principal components against noise, but rather quantifies the quality of the principal components with regards to the minimization of the distance to all objects in the data set. We hope to kindle with this work future research on the coMAD.

## References

1. Aggarwal, C.C., Yu, P.S.: Finding generalized projected clusters in high dimensional spaces, vol. 29. ACM (2000)
2. Böhm, C., Kailing, K., Kröger, P., Zimek, A.: Computing clusters of correlation connected objects. In: Proceedings of the 2004 ACM SIGMOD international conference on Management of data. pp. 455–466. ACM (2004)
3. Falk, M.: On mad and comedians. *Annals of the Institute of Statistical Mathematics* **49**(4), 615–644 (1997)
4. Hubert, M., Rousseeuw, P.J., Vanden Branden, K.: Robpca: a new approach to robust principal component analysis. *Technometrics* **47**(1), 64–79 (2005)
5. Pearson, K.: Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* **2**(11), 559–572 (1901)
6. De la Torre, F., Black, M.J.: Robust principal component analysis for computer vision. In: Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001. vol. 1, pp. 362–369. IEEE (2001)
7. Zhao, Q., Meng, D., Xu, Z., Zuo, W., Zhang, L.: Robust principal component analysis with complex noise. In: International conference on machine learning. pp. 55–63 (2014)

## Acknowledgement

This work has been funded by the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A. The authors of this work take full responsibilities for its content.



## Chapter 5

# CODEC - Detecting Linear Correlations in Dense Clusters using coMAD-based PCA

This chapter consists of the following publication:

[41] Maximilian Archimedes Xaver Hünemörder, Anna Beer, Daniyal Kazempour, and Thomas Seidl. "CODEC - Detecting Linear Correlations in Dense Clusters using coMAD-based PCA". in: *Proceedings of the Conference on "Lernen, Wissen, Daten, Analysen", Berlin, Germany, September 30 - October 2, 2019*. Ed. by Robert Jäschke and Matthias Weidlich. Vol. 2454. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pp. 111–114. URL: [http://ceur-ws.org/Vol-2454/paper\\_74.pdf](http://ceur-ws.org/Vol-2454/paper_74.pdf)

Nothing was changed compared to the final version which was published under a Creative Commons Attribution 4.0 International License (see <https://creativecommons.org/licenses/by/4.0/>)

**Statement of Originality:** The idea for CODEC was conceived in a discussion between D. Kazempour, A. Beer, and **M. Hünemörder**. The concept was then discussed with T. Seidl, who provided additional insights. **M. Hünemörder** then performed the experiments, and the manuscript was composed by D. Kazempour, A. Beer, and **M. Hünemörder**

- **Conception:** **Hünemörder** (Lead), Kazempour (Lead), Beer (Lead)
- **Planning:** All Authors
- **Execution:** **Hünemörder** (Lead)
- **Manuscript:** **Hünemörder** (Lead), Kazempour (Lead), Beer (Lead), Seidl (Support)

## CODEC - Detecting Linear Correlations in Dense Clusters using coMAD-based PCA

Maximilian Archimedes Xaver Hünemörder, Daniyal Kazempour, Anna Beer,  
and Thomas Seidl

Ludwig-Maximilians-Universität München, Munich, Germany  
{huenemoerder, kazempour, beer, seidl}@dbs.ifi.lmu.de

**Abstract.** The coMAD (co-median absolute deviation) is a measure for the joint median of two random variables. Previous experiments have shown that a coMAD-based PCA is more robust towards noise and outliers, yielding eigenvectors which represent linear correlation better than its covariance-based competitors. In this preliminary work we introduce CODEC - CORrelations in DENSE Clusters - a method for detecting linear correlations in dense clusters utilizing a coMAD-based PCA. The idea of CODEC is intriguingly simple: first a density-based clustering is performed using the well established clustering method DBSCAN. Then on each of the clusters PCA is performed. Instead of using the covariance matrix we use the coMAD matrix as a basis for performing PCA.

**Keywords:** Comedian · Correlation Clustering · Principal Component Analysis · CoMAD.

### 1 Introduction

As Data Analysis is becoming more and more important in recent years, a multitude of possible interesting properties of datasets have emerged. Clustering, which constitutes a huge field of data analysis with diverse sub-categories, leverages these properties to find sets of points which are similar to each other, but dissimilar to points of other sets or clusters. This similarity can be based on density, on the distance to centroids, or how well they fit to a certain distribution. On the other hand, points which are correlated only in certain dimensions can also be interpreted as similar, which is covered by subspace and correlation clustering algorithms. Surprisingly only few algorithms combine both concepts, and, to the best of our knowledge, none of them investigate found clusters further regarding possible correlations. Especially density-based clusters, which can be of any shape, can contain correlated data (rather than centroid-based clusters for example, which tend to have similar extensions in every dimension). Thus, we introduce CODEC, a new prototype algorithm which finds correlations in density-based clusters found by DBSCAN using an improved version of PCA to

---

Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

2 M.A.X. Hünemörder et al.

account for dispersions. We have found that using the coMAD matrix instead of the covariance matrix, we find less distorted main components of correlation clusters.

We provide an overview over related work in Section 2, including an introduction of the improved PCA using the coMAD (co-median absolute deviation) matrix. In Section 3 the algorithm is explained in detail and tested in Section 4. Section 5 concludes this short paper and gives ideas for future work.

## 2 Related Work

There already exist several sophisticated methods which detect linear correlated clusters, such as ORCLUS[2], 4C[3] or CASH[1]. ORCLUS and 4C rely on Principal Component Analysis (PCA). ORCLUS combines the PCA with k-Means and 4C uses PCA and a DBSCAN[4]-like approach. In this work-in-progress, we aim to harness the robustness of the coMad, originally introduced as the “comedian” in [5] to detect linear correlations in dense clusters.

## 3 Method

Since the coMAD is one of the core elements of our method, we will first elaborate on the definition of the coMAD. Suppose we are given a data matrix  $D$  of dimensionality  $d$ , where the rows represent a data record and their respective columns represent the features  $(A_1, \dots, A_d)$ . If we consider the variance, its analogon in the median context would be the median absolute deviation from the median (MAD):

$$mad(A_i) = med(|A_i - med(A_i)|)$$

Then the analogon to the covariance is the coMAD which is a generalization of the MAD:

$$com(A_i, A_j) := med((A_i - med(A_i))(A_j - med(A_j)))$$

Finally we use the definition of the coMAD to construct the coMAD matrix  $\Lambda$  known as:

$$\Lambda_D = \begin{pmatrix} mad(A_1) & \cdots & com(A_1, A_d) \\ \vdots & \ddots & \vdots \\ com(A_d, A_1) & \cdots & mad(A_d) \end{pmatrix}$$

Based on the coMAD matrix  $\Lambda$  the PCA is performed which yields the corresponding eigenpairs.

Our algorithm then proceeds as follows: First, dense clusters are detected by applying DBSCAN on the data set. Then on each of the dense clusters PCA is applied, using the coMAD instead of the covariance matrix. As a result we obtain a set of eigenvectors for each cluster, which show the direction of linear

correlations within the clusters. The intuition here is that instead of the direction of highest variance (which would be the classical result using the covariance matrix), the eigenvectors now point in the direction of the highest MAD, therefore the direction where most points are situated. This leads to the PCA being less contaminated by noise points that diverge from the main direction of linear correlation.

One may think that a dense cluster should already be almost without any noise. This however depends on the density of the clusters which is implicitly determined by the choice of the hyperparameters of DBSCAN, namely  $\varepsilon$ -range and *minpts* for the minimum number of objects required to be located within an  $\varepsilon$ -range. The larger the  $\varepsilon$ -ranges and the lower *minpts*, the less dense are clusters. Further the so called 'border points' can, depending on the  $\varepsilon$ -range, have similar effects as outliers and therefore skew the principal components of a PCA. In Section 4 we show a case where PCA based on the covariance matrix yields skewed results compared to PCA using the coMAD matrix.

## 4 Experiments

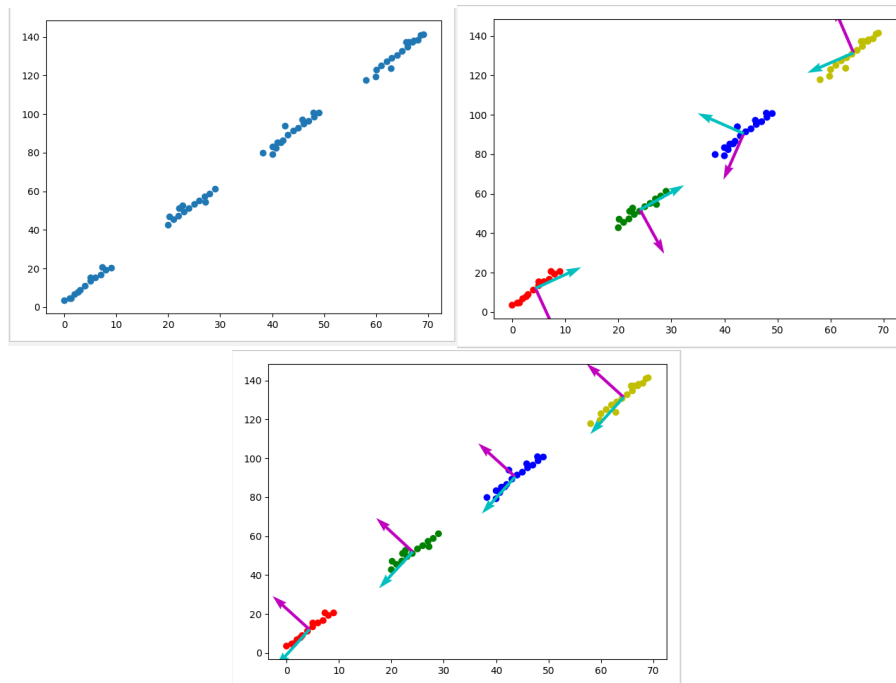
As a first experiment of our preliminary work, we constructed a data set with four clusters exhibiting linear correlations, both locally and globally, as it can be seen in Figure 1 (top left). Furthermore, each of these clusters are not perfectly linearly correlated but studded with noise. From a high-level view one could state that the noise should not have any impact on the result of the PCA. However, if we apply a covariance-based PCA on each of the dense clusters, the resulting eigenvectors are significantly skewed, as it can be seen in Figure 1 (top right). Especially in the blue cluster the noise leads to a massive distortion of the expected direction. The effects of using a coMAD-based approach become visible in Figure 1 (bottom), where despite the noise the detected eigenvectors remain robust. The algorithm and data generator used for the experiment were implemented in python and are publically available <sup>1</sup>.

## 5 Conclusion and Future Work

In summary we developed a method to find correlations in density-based clusters accurately and robust to noise and jitter. We showed that using the coMAD matrix for PCA delivers more intuitive results than using the covariance matrix, especially for real-world data which is usually not correlated perfectly. We plan to examine further combinations of correlation clustering and density-based clustering in future work. Investigating the trade-off between efficiency of the computation and improvement of the results using the coMAD matrix is a further subject of future work.

<sup>1</sup> <https://github.com/huenemoerder/CODEC>

4 M.A.X. Hünemörder et al.



**Fig. 1.** Top left: test data set; top right: the computed eigenvectors with a covariance-based PCA; bottom: the computed eigenvectors with a coMAD-based PCA

## References

1. Achtert, E., Böhm, C., David, J., Kröger, P., Zimek, A.: Global correlation clustering based on the hough transform. *Statistical Analysis and Data Mining: The ASA Data Science Journal* **1**(3), 111–127 (2008)
2. Aggarwal, C.C., Yu, P.S.: Finding generalized projected clusters in high dimensional spaces, vol. 29. ACM (2000)
3. Böhm, C., Kailing, K., Kröger, P., Zimek, A.: Computing clusters of correlation connected objects. In: *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. pp. 455–466. ACM (2004)
4. Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: *Kdd*. vol. 96, pp. 226–231 (1996)
5. Falk, M.: On mad and comedians. *Annals of the Institute of Statistical Mathematics* **49**(4), 615–644 (1997)



## Chapter 6

# SIDEKICK: Linear Correlation Clustering with Supervised Background Knowledge

This chapter consists of a preprint version of the following publication:

[42] Maximilian Archimedes Xaver Hünemörder, Daniyal Kazempour, Peer Kröger, and Thomas Seidl. "SIDEKICK: Linear Correlation Clustering with Supervised Background Knowledge". In: *Similarity Search and Applications - 12th International Conference, SISAP 2019, Newark, NJ, USA, October 2-4, 2019, Proceedings*. Ed. by Giuseppe Amato, Claudio Gennaro, Vincent Oria, and Milos Radovanovic. Vol. 11807. Lecture Notes in Computer Science. Springer, 2019, pp. 221–230. DOI: 10.1007/978-3-030-32047-8\_20. URL: [https://doi.org/10.1007/978-3-030-32047-8\\_20](https://doi.org/10.1007/978-3-030-32047-8_20)

It was reproduced with permission from Springer Nature.

**Statement of Originality:** The main inspiration for this paper was originally formulated in "*Deriving quantitative models for correlation clusters*" by Aichert et al. [2]. It was then further fleshed out by D. Kazempour, **M. Hünemörder**, and P. Kröger. Based on these discussions **M. Hünemörder** wrote code and performed further experiments, and extended the manuscript, which was reviewed and edited by D. Kazempour, T. Seidl and P. Kröger.

- **Conception:** Kazempour (Lead), Kröger (Lead), **Hünemörder** (Support)
- **Planning:** All Authors
- **Execution:** **Hünemörder** (Lead)
- **Manuscript:** **Hünemörder** (Lead), Kazempour (Support), Kröger (Support), Seidl (Support)

# SIDEKICK: Linear Correlation Clustering with Supervised Background Knowledge

M.A.X. Hünemörder, Daniyal Kazempour, Peer Kröger, and Thomas Seidl

Ludwig-Maximilians-Universität München, Munich, Germany  
{huenemoerder, kazempour, kroeger, seidl}@dbs.ifi.lmu.de

**Abstract.** While explainable AI (XAI) is gaining in popularity, other more traditional machine learning algorithms can also benefit from increased explainability. A semi-supervised approach to correlation clustering opens up a promising design space that might provide such explainability to correlation clustering algorithms. In this work, semi-supervised linear correlation clustering is defined as the task of finding arbitrary oriented subspace clusters using only a small sample of supervised background knowledge provided by a domain experts. This work describes a first foray into this novel approach and provides an implementation of a basic algorithm to perform this task. We have found that even a small amount of supervised background knowledge can significantly improve the quality of correlation clustering in general. With confidence it can be stated, the results of this work have the potential to inspire several more semi-supervised approaches to correlation clustering in the future.

**Keywords:** clustering, subspace, correlation, semi-supervised, background knowledge

## 1 Introduction

Explainable Artificial Intelligence (XAI) is rapidly gaining popularity among the data science community [6]. XAI is mainly motivated by the fact that neural networks, especially deep neural networks are often times treated as "black boxes", i.e. procedures and errors can be close to impossible to be comprehended by humans [8]. This creates problems that are interdisciplinary and manifold. For example, one major issue is the lack of trust in machine learning algorithms, both concerning the input and the results. XAI could help to build trust in AI, not only concerning the general population, but also researchers from domains other than computer science, i.e. users of such algorithms [8]. Even these domain experts want to be assured that the results they are seeing are direct and uncontaminated answers to their input and therefore require explainable results and processes [13]. But outside of AI and deep learning, other data mining tasks and algorithms could also benefit from increased explainability [14].

Exploring the design space at the core of this work, a semi-supervised approach to correlation clustering, has the potential of producing explainable correlation clustering algorithms. In this context linear correlation clustering is defined as the task of finding arbitrary oriented subspace clusters [15]. This is, firstly, because quantitative models explaining the results of correlation clustering algorithms can easily be derived and made interpretable by domain experts [2]. Secondly, because semi-supervised algorithms such as constrained k-means clustering and similar algorithms allow for background knowledge driven machine learning, i.e. human ideas and opinions can influence the results of such algorithms. This allows domain experts to more accurately raise their intended queries towards the data [9]. Since to the best of our knowledge there has not been any

semi-supervised algorithm that tackles the data mining task of correlation clustering, the goal of this work is to design, implement and evaluate an algorithm that detects linear correlated clusters given a small subset of a priori labeled data instances. The major contributions of this work are the introduction of SIDEKICK (SuperviseD Expert Knowledge Influenced Correlation Clustering), a first semi-supervised correlation clustering algorithm and a novel notion of correlation in the context of clustering ( $\phi$ -correlated clusters).

## 2 Related Work

Since we are introducing the first linear correlation clustering algorithm that considers supervised background knowledge, we owe definitions as well as an overview on the related work in both fields, namely (a) linear correlation clustering and (b) clustering with supervised background knowledge. The task of linear correlation clustering is defined as finding clusters within a given data set that are located within interesting subspaces which are arbitrarily oriented [15]. It further means that the data objects within each of the clusters exhibit a linear correlation between a subset of their features. In this context a broad range of related work exists. ORCLUS [7] was the first of its kind tailored for detection of such clusters, followed by other algorithms such as 4C [10], HiCO [5], ERiC [4], COPAC [3] and CASH [1].

All named methods so far excel at certain aspects, but are not capable of dealing with supervised background knowledge. This is an aspect which our method is addressing. On the other hand there are semi-supervised methods for other types of clustering. Such clustering tasks, where results are influenced by semantic decisions by domain experts or data analysts are called *constrained clustering* tasks. The earliest algorithm tackling this task, Constrained K-means, performs a variation of the classic k-means algorithm under the restriction that instances have to be members of their corresponding clusters using a *must-link* constraint and two instances are not allowed to be in the same cluster, i.e. a *cannot-link* constraint [19]. There are multiple categories of constraints. At the time of this work we know of other contributions introducing instance-level, cluster-level and model-level constraints. Instance-level constraints are constraints specified between two instances, for example the must-link and cannot-link constraints from above. Cluster-level constraints are specified for multiple instances belonging to a cluster. For example  $\epsilon$ - and  $\delta$ -constraints [11]. Model-level constraints use a different approach. Here a user is shown the result of a certain clustering algorithm. Then the user can decide whether they like this specific result or not. If they do not like it, the algorithm is repeated but with automatic constraints that ensure that the result does not resemble the undesired results from earlier iterations [12].

### 2.1 Deriving Quantitative Models for Correlation Clusters

The main inspiration and basis of SIDEKICK is a 2006 paper with the title "Deriving Quantitative Models for Correlation Clusters" [2]. The primary idea behind this publication was to add an additional post-processing step to existing correlation clustering algorithms. In essence, a model is derived by using PCA on each cluster. The smallest number of Eigenvectors of this cluster  $C_i$  that explain a percentage higher than a predefined threshold  $\alpha$  are called *strong Eigenvectors*  $\hat{V}_C$  of  $V_C$ . The remaining Eigenvectors are the *weak Eigenvectors*  $\hat{V}_C$  of  $V_C$ . The weak eigenvectors are used to derive a Hesse Normal Form hyperplane equation system, that can be solved by Gauß-Jordan Elimination for better readability. The paper also suggests a method of using the generated models to predict the cluster membership of additional instances. In order to predict the cluster

membership of an arbitrary point  $p$  the distance between  $p$  and the correlation hyperplane of each Cluster  $C_i$  needs to be calculated. The distance is equal to the length of the vector between the orthogonal projection of  $p$  onto the correlation hyperplane and the point  $p$  itself. The projection can be calculated using the strong eigenvectors  $s_1, s_2, \dots, s_\lambda \in \tilde{V}_{C_i}$  and the point, both normalized by the mean vector  $\bar{c}_i$ , in the following fashion:

$$d(p, C_i) = \left\| p - \sum_{j=1}^{\lambda} \langle p, s_j \rangle s_j \right\|$$

By calculating the distance of all cluster instances to the corresponding hyperplane the standard deviation  $\sigma_i$  can be derived. Assuming that the deviations of each cluster  $C_i$  fit to a Gaussian Distribution  $G_i$ , we can derive an equation to calculate the probability of a point  $p$  belonging to a distribution of a certain cluster  $C_i$ :

$$G_i(p, \sigma_i) = \frac{1}{\sigma_i \sqrt{2\pi}} e^{-\frac{(d(p, C_i))^2}{2\sigma_i^2}} \quad P(C_i | p, \sigma_i) = \frac{G_i(p, \sigma_i)}{\sum_{j=1}^n G_j(p, \sigma_j)}$$

### 3 Semi-Supervised Correlation Clustering

As stated in the introduction, the goal of this work is to design an algorithm that provides a solution to a very specific problem. A domain expert has found correlations in a  $R$ -dimensional database  $D^R$  and specified certain instances which belong to these different correlations. The set of sets of the instances belonging to a correlation is called the background knowledge  $BK$ , while each single correlation cluster derived from the background knowledge is denoted as  $BK_i$ . The cardinality of  $BK_i$  is thought to be significantly smaller than the cardinality of  $D^R$ . While the dimensionality of  $BK$  is equal to  $R$ , the correlation dimensionality of each individual background knowledge cluster  $\lambda_{BK_i}$  has to be strictly smaller than  $R$ . Simply said the task is to utilize the background knowledge provided by a domain expert to decide which other unlabeled instances in  $D$  belong to each of the correlation clusters  $BK_i$ , while adhering to the restrictions set by the expert knowledge. To be more concise, the opinion of the domain expert is extrapolated and applied to  $D \setminus \{BK\}$ .

This task can be designated as semi-supervised, because it is an traditionally unsupervised task – correlation clustering – but includes preexisting supervised expert knowledge that influences the results. Additionally the resulting models can be simplified as described in subsection 2.1 and the domain expert can compare the models derived from the expert knowledge to the results. Thereby they receive feedback about how accurately the correlations can be extrapolated onto the unlabeled instances and how much the resulting model deviates from their provided background knowledge. Furthermore all subspace clustering algorithms are usually categorized as either top down or bottom up, because of the circular dependency inherent to the task [15]. In the case of semi-supervised correlation clustering the background knowledge already provides an a priori clustering. Thus the algorithm can work similarly to top-down algorithms using the background knowledge as a starting point. The task we are left with is finding the relevant subspace for each cluster and searching for other instances that are situated in these subspaces. In order to find these subspaces and predict additional points, this work contains a semi-supervised extension of the algorithm from subsection 2.1.

### 3.1 SIDEKICK

SIDEKICK follows four main steps. At the beginning we calculate the correlation hyperplanes for each background knowledge cluster using the algorithm from [2], which corresponds to `derive_model(X)`. Additionally in accordance with section 2.1 the standard deviation  $\sigma$  of distances between the cluster instances and the corresponding correlation hyperplane is calculated for each cluster. Each correlation cluster  $C_i$  is thereby clearly defined by its model, which is at this point defined as consisting of its eigenvectors – split into weak and strong (i.e. ) – and the standard deviation  $\sigma_i$  and mean vector  $\mu_i$ . The strong eigenvectors are needed to calculate the distance between an instance and the hyperplane. The standard deviation is needed to derive the normal distribution that is assumed to have produced the noise along the weak components. In summary:

1. Derive the underlying models for the each of the ground truth clusters
2. Predict the labels for all unlabeled instances
3. Assign either all or a subset of the predicted instances to their corresponding cluster
4. Deriving updated models for each resulting cluster and simplifying the hyperplane equation to highlight the underlying correlations

### 3.2 Unlimited SIDEKICK

The basic algorithm is called unlimited SIDEKICK since at step 3 all unlabeled instances are added to their corresponding cluster. This performs well with highly correlated clusters. Generally if a cluster is 100% correlated, i.e. it only possesses variance along the strong components SIDEKICK is expected to classify unlabeled instances correctly, even when only using a background knowledge consisting very few instances. However, missclassifications at the intersections of different clusters can occur, but only in cases where the correct membership of an instance is indeterminable anyways. Generally the Winner-Takes-All principle is applied, i.e. every instance is always classified as a member of the most probable cluster. On the other hand the biggest short-coming of the unlimited SIDEKICK algorithm is that whenever there are instances that were generated by a process not belonging to a cluster that was included in the background knowledge, almost all of these outliers are added to the cluster with the highest variance along the weak components, i.e. the lowest density along the weak components. This effect can be explained by the properties of the Gaussian distributions that are used to assign instances to a cluster. A slight difference of  $\sigma$  between two clusters has a huge impact on the normalized probability for a single instance and thereby it will be classified as highly likely to belong to the first cluster, even though it most probably does not belong to either correlation. In conclusion the resulting problem is how to find a suitable subset of the predicted points for each cluster to be assigned.

### 3.3 $\phi$ -correlated SIDEKICK

Our approach to solve this problem is to work with the definition of strong components and  $\alpha$  from [2]. Then only using a new hyperparameter  $\phi$  and the sum of the strong eigenvalues  $\check{e}$  of a background knowledge cluster, we can approximate the standard deviation  $l$  along the weak components of the resulting cluster:

$$l = \sqrt{\left(\frac{1}{\phi} - 1\right) \cdot \sum_{\forall e_i \in \check{e}} e_i}$$

This cluster is then  $\phi$ -correlated, in accordance to the original definition of  $\alpha$ . Which is the minimum ratio of the total variance that can be explained by the strong components. Using the eigenvalues  $e$  of a cluster it holds:

$$\frac{\sum_{i=1}^{\lambda} e_i}{\sum_{i=1}^d e_i} \geq \alpha$$

Additionally, the number of strong components is equal to the correlation dimensionality  $\lambda$  of a cluster. That means, if the variance of the weak components becomes to high, they might become strong components. Therefore, the maximum variance  $\hat{e}$  that the weak eigenvectors can explain while still resulting in a  $\phi$ -correlated cluster is therefore equal to

$$\hat{e} = \frac{\check{e}}{\phi} - \check{e}$$

According to the Three Sigma Rule [17] about 68% of all instances belonging to a normal distribution are situated within a distance of one standard deviation from the mean and 99.7% at triple the standard deviation. That means if we only include the subset of instances for which the distances are lower than three times the square root of this variance, we can find about 99.7% of the instances that would be included in a normal distribution of a cluster that is  $\phi$ -correlated. Which would mean that step 3 of SIDEKICK now involves only the subset of the predicted instances that are part of a certain correlation with a strength of  $\phi$ . Therefore, as long as the direction of the hyperplane is close to the background knowledge, a domain expert can now specify the exact strength of correlation they are searching for.

### 3.4 Runtime Complexity

If we denote  $n$  as the total number of instances,  $c$  as the total number of clusters and  $bk$  as the amount of instances used as background knowledge the complexity of SIDEKICK for all clusters can be computed as  $\mathcal{O}(c \cdot n)$ , if  $bk$  is significantly smaller than  $n$  or  $\mathcal{O}(c \cdot bk_2)$  if not. We came to this conclusion by simplifying the overall runtime complexity per cluster, that can be computed as the sum of the complexities of:

- Performing PCA on the background knowledge; Using Power Iteration [16] this would be  $\mathcal{O}(bk^2)$
- Computing the standard deviation of the distances between the background knowledge objects and their corresponding clusters; This is  $\mathcal{O}(bk)$
- Computing the probability of each object belonging to a cluster model; This is  $\mathcal{O}(2(n - bk))$  for each unlabeled object, since we need to make two computations per object (c.f. Section 2.1)

In summary the overall complexity can be denoted as:

$$\sum_i^c \overbrace{(\mathcal{O}(bk_i^2))}^{\text{PCA}} + \overbrace{\mathcal{O}(bk_i)}^{\text{Compute standard deviation}} + \overbrace{\mathcal{O}(n - bk_i)}^{\text{Probability for single cluster}} + \overbrace{\mathcal{O}(n - bk_i)}^{\text{Normalized Probability over all clusters}}$$

The different variants only add a single step with a complexity of  $\mathcal{O}(n)$  and therefore do not have a significant effect on the overall complexity.

## 4 Experiments and Discussion

In this section we evaluate SIDEKICK using two synthetic data sets. One data set contains 3000 three dimensional instances segmented into six linear correlated clusters, five with a correlation dimensionality of one and a single cluster with a correlation dimensionality of two. What makes this data set especially challenging is the fact that the linear correlated clusters exhibit different densities. The second data set is equal to the first one, except for a set of 1250 randomly generated outlier objects that were added to it. Using these two datasets we compared SIDEKICK against five established correlation clustering algorithms from the related work section (Section 2). Suitable Hyperparameter settings for each of the five algorithms have been determined through either a grid-based or a sequential scan (ERiC) for both data sets. The experiments were conducted using the ELKI [18] data mining framework. Furthermore, we have provided all the hyperparameter settings in Tables 1 and 2 and made the source code for SIDEKICK and the test data sets publicly available<sup>1</sup> to ensure reproducibility. Table 1 illustrates that the correlation clustering algorithm 4C achieved the best results with an Adjusted Rand Index (ARI) of 78,93%. When we used the same hyperparameter settings on the data set that contains outliers, 4C remained the best performing correlation clustering algorithm. Its ARI decreased from 0.7893 to 0.7074 while other competing methods achieve a by far lower ARI score as seen in Table 2.

Table 1: ARI results and hyperparameter settings of competitive methods on the synthetic data set

Dataset	Algorithm	ARI	Hyperparameter Settings
without noise	CASH	0.5998	minpts: 370, maxlvl: 20, jitter: 2.5
without noise	4C	<b>0.7893</b>	$\varepsilon$ : 8.0, minpts: 15
without noise	COPAC	0.5691	$\varepsilon$ : 8.0, minpts: 15, kNN: 81
without noise	ORCLUS	0.7351	k: 6, l:2
without noise	ERiC	0.2260	k: 6

Table 2: ARI results and hyperparameter settings of competitive methods on the synthetic data set with noise

Dataset	Algorithm	ARI	Hyperparameter Settings
with noise	CASH	0.5348	minpts: 400, maxlvl: 30, jitter: 4.5
with noise	4C	<b>0.7074</b>	$\varepsilon$ : 8.0, minpts: 15
with noise	COPAC	0.3786	$\varepsilon$ : 8.0, minpts: 15, kNN: 89
with noise	ORCLUS	0.2465	k: 6, l:2
with noise	ERiC	0.146	k: 6

<sup>1</sup> <https://github.com/huenemoerder/SIDEKICK>

Table 3: ARI results and hyperparameter settings of SIDEKICK on the synthetic data set with and without noise

Setting	Algorithm	Average ARI	Max. ARI	Min. ARI	Variance ARI
Without noise, 0.99 BK	Unlimited	0.9569	1.0000	0.5634	0.0064
With noise, 0.99 BK	$\phi$	0.6957	0.9644	0.2941	0.0248
With noise, 0.96 BK	$\phi$	0.9381	0.9747	0.8568	0.0007

To evaluate SIDEKICK we started by using only unlimited SIDEKICK to cluster the data set without outliers, since all other variants were specifically designed to deal with outliers. We chose the background knowledge randomly from each cluster, mimicking a domain scientists expertise. To avoid the influence of the random choice, we repeated the experiment one hundred times using different random seeds. Our method achieved an average ARI of 95%. To cluster the the data set that contains outliers, we used the  $\phi$  algorithm, setting  $\phi$  individually for each cluster. We started by using only 1% background knowledge per cluster, which equals to 3 instances per cluster. This yielded an average ARI of 69% with a variance of 2% which means it was on average as good as 4C. When we increased the amount of background knowledge to 4%, which means that we used 20 instances as domain expert knowledge per cluster, the average ARI rose to 93%. This is superior to any of the state-of-the-art correlation clustering methods in our experiments. Even in the worst case, we got an ARI of 85%, which is still above any of the competing methods.

Finally we want to highlight, that it was not our intention to show whether SIDEKICK is better than any of its competitors, since these competitors do not use any background knowledge. Rather the core message we want to convey is, that even a small amount of background knowledge is sufficient to boost the performance of solving a correlation clustering task significantly. Adding a further hyperparameter like  $\phi$  can increase the robustness against noise and outliers.

## 5 Conclusion and Future Work

In conclusion the algorithm discussed in this work demonstrates the prospects of a semi-supervised approach to correlation clustering. As we have seen and discussed in the experiments using just a small amount of background knowledge can drastically improve the results of solving a correlation clustering task. Therefore any correlation clustering algorithm could theoretically benefit from such background knowledge.

Generally, SIDEKICK and its different variants should provide a useful toolkit for data exploration. In relation to the different variants of SIDEKICK itself we have learned that when working with a data set without outliers unlimited SIDEKICK is always the best choice. When working with a data set that contains many outliers choosing individual  $\phi$ 's for each cluster should be the best solution. The only complication that revealed itself is the dependence on correct background knowledge. But, firstly, this is intentional, because the goal of the algorithm was to trust the knowledge of the domain experts and only change it slightly at best. Secondly, this conclusion is somewhat diminished by the fact that during evaluation, the background knowledge for each cluster was sampled randomly from that cluster. Background knowledge provided by humans should usually be much closer to the truth and thereby provide better results even when using small amounts of instances as background knowledge.

## References

1. Achtert, E., Böhm, C., David, J., Kröger, P., Zimek, A.: Global correlation clustering based on the hough transform. *Statistical Analysis and Data Mining: The ASA Data Science Journal* **1**(3), 111–127 (2008)
2. Achtert, E., Böhm, C., Kriegel, H.P., Kröger, P., Zimek, A.: Deriving quantitative models for correlation clusters. In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. pp. 4–13. ACM (2006)
3. Achtert, E., Böhm, C., Kriegel, H.P., Kröger, P., Zimek, A.: Robust, complete, and efficient correlation clustering. In: *Proceedings of the 2007 SIAM International Conference on Data Mining*. pp. 413–418. SIAM (2007)
4. Achtert, E., Böhm, C., Kriegel, H.P., Zimek, A., et al.: On exploring complex relationships of correlation clusters. In: null. p. 7. IEEE (2007)
5. Achtert, E., Böhm, C., Kröger, P., Zimek, A.: Mining hierarchies of correlation clusters. In: *Scientific and Statistical Database Management, 2006. 18th International Conference on*. pp. 119–128. IEEE (2006)
6. Adadi, A., Berrada, M.: Peeking inside the black-box: A survey on explainable artificial intelligence (xai). *IEEE Access* **6**, 52138–52160 (2018)
7. Aggarwal, C.C., Yu, P.S.: Finding generalized projected clusters in high dimensional spaces, vol. 29. ACM (2000)
8. Akata, Z., Stumpf, S., Kieseberg, P., Holzinger, A.: Explainable ai: The new 42?
9. Basu, S., Davidson, I., Wagstaff, K.: *Constrained clustering: Advances in algorithms, theory, and applications*. CRC Press (2008)
10. Böhm, C., Kailing, K., Kröger, P., Zimek, A.: Computing clusters of correlation connected objects. In: *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*. pp. 455–466. ACM (2004)
11. Davidson, I., Ravi, S.: Clustering with constraints: Feasibility issues and the k-means algorithm. In: *Proceedings of the 2005 SIAM international conference on data mining*. pp. 138–149. SIAM (2005)
12. Gondek, D., Vaithyanathan, S., Garg, A.: Clustering with model-level constraints. In: *Proceedings of the 2005 SIAM International Conference on Data Mining*. pp. 126–137. SIAM (2005)
13. Holzinger, A., Kieseberg, P., Weippl, E., Tjoa, A.M.: Current advances, trends and challenges of machine learning and knowledge extraction: From machine learning to explainable ai. In: *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*. pp. 1–8. Springer (2018)
14. Kazempour, D., Seidl, T.: Insights into a running clockwork: On interactive process-aware clustering. In: *Proceedings of the 22nd International Conference on Extending Database Technology (EDBT)*, in press. pp. 000–000 (2019)
15. Kriegel, H.P., Kröger, P., Zimek, A.: Subspace clustering. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* **2**(4), 351–364 (2012)
16. Mises, R., Pollaczek-Geiringer, H.: *Praktische verfahren der gleichungsauflösung*. ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik **9**(2), 152–164 (1929)
17. Pukelsheim, F.: The three sigma rule. *The American Statistician* **48**(2), 88–91 (1994), <http://www.jstor.org/stable/2684253>
18. Schubert, E., Zimek, A.: ELKI: A large open-source library for data analysis - ELKI release 0.7.5 "heidelberg". *CoRR abs/1902.03616* (2019), <http://arxiv.org/abs/1902.03616>
19. Wagstaff, K., Cardie, C., Rogers, S., Schrödl, S., et al.: Constrained k-means clustering with background knowledge. In: *ICML*. vol. 1, pp. 577–584 (2001)

## Acknowledgement

This work has been funded by the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A. The authors of this work take full responsibilities for its content.



## Chapter 7

# Implicit Hough Transform Neural Networks for Subspace Clustering

This chapter consists of a preprint version of the following publication:

©2021 IEEE. Reprinted, with permission, from Julian Busch, Maximilian Hünemörder, Janis Held, Peer Kröger, and Thomas Seidl. “Implicit Hough Transform Neural Networks for Subspace Clustering”. In: *2021 International Conference on Data Mining, ICDM 2021 - Workshops, Auckland, New Zealand, December 7-10, 2021*. IEEE, 2021, pp. 441–448. DOI: 10.1109/ICDMW53433.2021.00060. URL: <https://doi.org/10.1109/IC%5C-DM%5C-W53%5C-43%5C-3.2021.00060>

**Statement of Originality:** The concept of this paper originated in discussions between J. Busch, D. Kazempour, and **M. Hünemörder**. This idea was then used as the basis for a bachelor thesis by J. Held and supervised by J. Busch and **M. Hünemörder**. The code for the experiments was mainly written by J. Busch, based on the code of J. Held. **M. Hünemörder** added code to allow running older subspace clustering algorithms from ELKI and ran experiments on the time performance of the algorithm. The manuscript was mainly written by J. Busch and **M. Hünemörder**, loosely based on work by J. Held. P. Kröger and T. Seidl reviewed and edited the paper.

- **Conception:** Busch (Lead), **Hünemörder** (Lead), Kazempour (Support)
- **Planning:** Busch (Lead), **Hünemörder** (Lead), Held (Support)
- **Execution:** Busch (Lead), Held (Lead), **Hünemörder** (Support)
- **Manuscript:** Busch (Lead), **Hünemörder** (Lead), Held (Support), Kröger (Support), Seidl (Support)

# Implicit Hough Transform Neural Networks for Subspace Clustering

Julian Busch  
LMU Munich  
Munich, Germany  
busch@dbs.ifi.lmu.de

Maximilian Hünemörder  
CAU Kiel  
Kiel, Germany  
mah@informatik.uni-kiel.de

Janis Held  
LMU Munich  
Munich, Germany  
j.held@campus.lmu.de

Peer Kröger  
CAU Kiel  
Kiel, Germany  
pkr@informatik.uni-kiel.de

Thomas Seidl  
LMU Munich  
Munich, Germany  
seidl@dbs.ifi.lmu.de

**Abstract**—Subspace clustering constitutes a fundamental task in data mining and unsupervised machine learning with myriad applications. We present a novel approach to subspace clustering that detects affine hyperplanes in a given arbitrary-dimensional dataset by explicitly parametrizing them and optimizing their parameters using gradient updates w.r.t. a differentiable loss function. The explicit parametrization allows our model to avoid the exponential search space incurred by models relying on an explicit Hough transform to detect subspaces by searching for high-density points in parameter space. Compared to other existing approaches, our method is highly scalable, can be trained very efficiently on a GPU, is applicable to out-of-sample data, and is amenable to anytime scenarios since training can be stopped at any time and convergence is usually fast. The model can further be viewed as a linear neural network layer and trained end-to-end with an autoencoder to detect arbitrary non-linear correlations. We provide empirical results on a wide array of synthetic datasets with different characteristics following a rigorous evaluation protocol. Our results demonstrate the advantageous properties of our model and additionally reveal that it is particularly robust to jitter and noise present in the data.

**Index Terms**—Subspace Clustering, Neural Networks

## I. INTRODUCTION

Clustering constitutes a fundamental task in data mining and unsupervised learning. Applications range from data pre-processing for downstream tasks to grouping similar data objects, such as measurement points, text documents, or images. Clusters in real-world data often do not appear in the full-dimensional data space but are rather situated in individual lower-dimensional subspaces. Subspace clustering algorithms [1], [2] aim at detecting such clusters.

We present a novel approach to finding such subspace clusters in arbitrary-dimensional datasets by explicitly parametrizing affine hyperplanes and optimizing their parameters using gradient updates w.r.t. a differentiable loss function. This explicit parametrization avoids the exponential search space

faced by traditional Hough transform-based subspace clustering methods. Such methods explicitly transform the feature space to a parameter space and search this space for high-density regions. In addition to scalability problems, the search procedure is error-prone. On the other hand, our method fits hyperplanes directly in the data space, instead of first translating the points to parameter space and subsequently translating the detected hyperplanes back to the original space. By explicitly shifting hyperplanes in data space, our model also implicitly moves through parameter space without materializing it. This connection between the data and the parameter space is visualized in Figure 1.

The explicit parametrization and differentiable loss function lead to an interpretation of our model as a linear neural network layer with a specific activation function that can be trained using gradient-based optimization. Consequently, our method is highly scalable and can be trained highly efficiently on a GPU. Training time is linear, and space on the GPU can even be reduced to a constant size by performing update steps only on small mini-batches of the whole dataset. Further, the training procedure can be stopped at any time, and intermediate results can be reported, such that our model is naturally amenable to anytime scenarios. Our model could further be combined and trained end-to-end together with an autoencoder to detect non-linear clusters and is able to cluster out-of-sample points by simply assigning them to their closest hyperplanes.

## II. RELATED WORK

Various subspace clustering algorithms have been proposed in different research communities. In the data mining community, different variations of the subspace clustering problem have been considered, e.g., based on whether the subspaces are required to be axis-aligned. We refer here to subspace clustering in its more general form, where each cluster corresponds to an affine subspace. This setting is also often referred to as *arbitrarily-oriented subspace clustering* or *correlation clustering* in the data mining literature [1]. This includes the

This work has been partially funded by the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A. The authors of this work take full responsibilities for its content.

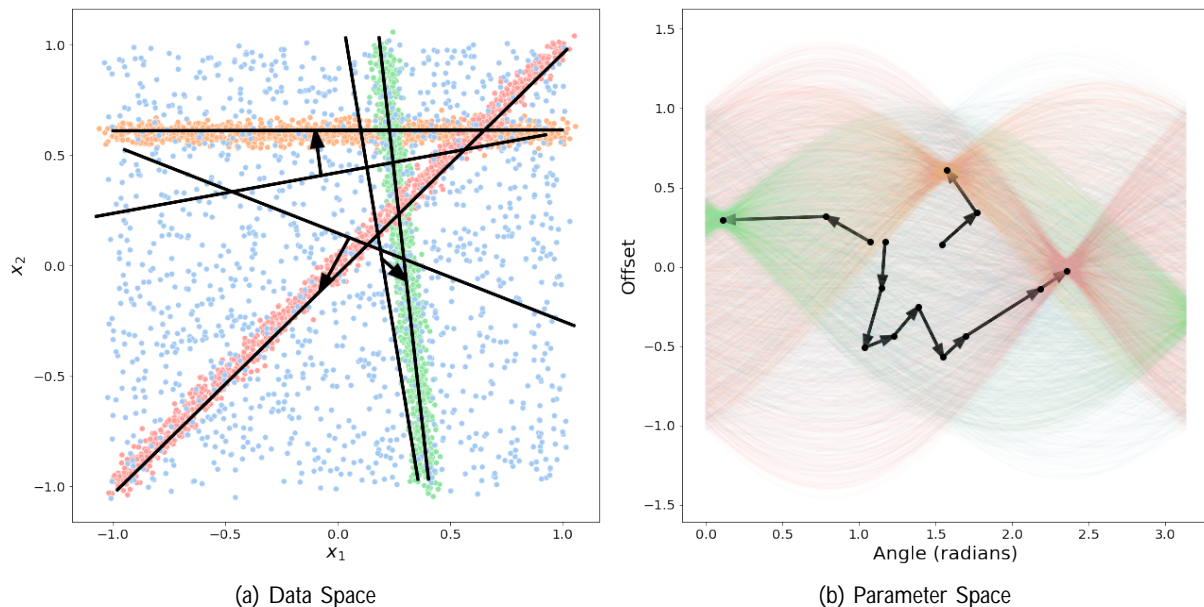


Fig. 1: Our methods iteratively shifts hyperplanes in the data space. Each hyperplane corresponds to a point in parameter space. While our method implicitly moves along a trajectory in parameter space, other Hough transform based methods rely on detection of dense regions.

algorithm *CASH* [3], which detects affine subspaces using a Hough transform. Data points are mapped into a parameter space, in which dense regions are detected using a grid-based search. Dense regions correspond to hyperplanes in the original space that contain many of the data points. In order to identify clusters of lower dimensionality, the algorithm is applied recursively on the already detected hyperplanes. However, the search procedure is error-prone and does not scale well. In comparison, our method performs robust and scalable optimization in the data space. Also very closely related to our approach, *k-Planes* [4] fits hyperplanes using *k*-Means-like optimization. Points are iteratively assigned to their respective closest hyperplanes, and hyperplanes are fit to the points assigned to them. In comparison, our method performs gradient-based optimization, which turns out to be less sensitive to initialization, more robust, and avoids eigenvector computations for fitting hyperplanes. *ORCLUS* [5] also relies on *k*-Means-style optimization, but starts with a higher number of clusters and incrementally merges these seed clusters and reduces their dimensionalities until *k* clusters are left. *4C* [6] follows a density-based clustering approach [7] to link smaller subspace clusters based on the similarities of their bases. This makes it unnecessary to specify the number of clusters in advance. Similarly, *COPAC* [8] relies on local correlations to group points with similar local correlations in a bottom-up fashion. Some further algorithms explore different more specific directions, such as hierarchical subspace clustering [9], [10], detecting a single optimal subspace for clustering [11], or detecting non-linear correlations in datasets [12]. While our algorithm already provides explicit subspace representations, another work [13] focuses on deriving such representations for

methods that are not explicitly parameterized. This work has also been extended to employ a small set of existing labels for semi-supervised clustering [14]. A semi-supervised extension of our method constitutes an interesting potential direction for future work.

In addition to the data mining community, subspace clustering algorithms have been investigated in the machine learning, and computer vision communities [2]. Typical applications considered in this context include different image clustering tasks, such as face or digit clustering and motion segmentation. *Self-expressive* algorithms [2] have gained particular attention recently. These algorithms rely on the idea that points in the same subspace can be represented as linear combinations of each other. The corresponding coefficients are learned and collected in a quadratic coefficient matrix that can be converted to a subspace-affinity matrix, which finally can be clustered using spectral clustering. Most of these algorithms, including a pioneering algorithm, *SSC* [15], consider linear subspace clusters, i.e., subspaces that contain the origin. Affine subspaces can be detected by adding an additional constraint to the optimization problem. A major downside of these algorithms is that they do not scale well. While more scalable variants have been proposed [16], [17], it is not clear how the affinity constraint could be handled during optimization, which potentially limits applicability. On the other hand, these algorithms can be easily combined with other neural network models to detect non-linear clusters [17], [18]. Similarly, our proposed algorithm is highly scalable and efficient and can potentially be combined with an autoencoder to detect non-linear clusters.

### III. IMPLICIT HOUGH TRANSFORM NEURAL NETWORKS

Given a data matrix  $X \in \mathbb{R}^{n \times d}$ , where each row corresponds to a point in a  $d$ -dimensional ambient space, we aim at partitioning the  $n$  data points into  $K$  subspace clusters. Thereby, each cluster is represented by a lower-dimensional subspace of the ambient space. While we assume that different subspaces may have different numbers of dimensions, we model each cluster as an affine hyperplane of dimension  $d-1$ . This allows our model to be agnostic of the actual subspace dimensions. Intuitively, our algorithm fits hyperplanes by iteratively shifting them through the data space, where they are attracted by the data points. Training can be performed with linear time and constant space complexity and turns out to be very robust.

Formally, an affine hyperplane can be represented by a normal vector  $w \in \mathbb{R}^d$ , and an offset  $b \in \mathbb{R}$ . The hyperplane then corresponds to the  $(d-1)$ -dimensional space orthogonal to  $w$ . The offset  $b$  allows for shifting the hyperplane away from the origin. Given a set of  $K$  hyperplanes, the parameters can be collected in a weight matrix  $W \in \mathbb{R}^{d \times K}$  and an offset vector  $b \in \mathbb{R}^K$ .

To quantify the error incurred by representing a given dataset by a set of hyperplanes, our model requires a differentiable loss function and, in turn, a notion of distance between points and hyperplanes. The distance between a point  $x_i$  and a hyperplane  $(w_j, b_j)$  can be measured in terms of the signed orthogonal projection distance

$$\text{pdist}_{ij} = x_i^T \frac{w_j}{\|w_j\|_2} - b_j. \quad (1)$$

The sign of this expression indicates on which side of the hyperplane  $x_i$  is located. A value of zero indicates that the point is located directly on the hyperplane. To obtain a proper distance function, one could simply consider the absolute value. To additionally obtain a similarity function, we apply an activation function

$$\text{sim}_{ij} = \frac{1}{1 + \left(\frac{\text{pdist}_{ij}}{\lambda}\right)^2}. \quad (2)$$

The square leads to stronger localization, such that points closer to a hyperplane exert a stronger attraction. Compared to, e.g., a radial basis function, the tail of this activation function is longer, such that more distant points can still attract a hyperplane. The length of the tail can be controlled by a hyperparameter  $\lambda$ . Pairwise similarities between all  $n$  points and  $K$  hyperplanes can be efficiently computed in matrix form:

$$S = \frac{1}{1 + \left(\frac{X\tilde{W}-b}{\sigma\lambda}\right)^2} \in \mathbb{R}^{n \times K}, \quad (3)$$

where  $\tilde{W}$  results from  $W$  by  $L_2$ -normalizing all columns. Compared to Equation 2, we additionally normalize the raw projection distances by their standard deviation  $\sigma$ . This allows our model to adapt to the structure of the dataset and to obtain more flexibility compared to a fixed parameter  $\lambda$ .

The similarity function is scaled to  $(0, 1]$ , where a maximum value indicates that the respective point lies directly on the hyperplane. The cluster assignment for  $x_i$  is given as

$$y_i = \underset{j=1, \dots, K}{\text{argmax}} s_{ij}. \quad (4)$$

A suitable loss function to be minimized by our model should be differentiable and minimal when each point fits perfectly into at least one hyperplane. For a point  $x_i$ , the latter property can be expressed by taking the product of the distances  $1 - s_{ij}$  to all hyperplanes  $j = 1, \dots, K$ . If  $x_i$  lies exactly on one hyperplane, the loss is minimal. Further,  $x_i$  can still lie on or close to multiple hyperplanes without incurring a higher loss. This property is desirable since subspace clusters can often intersect. Accordingly, we consider the following loss function w.r.t. a dataset  $X$ :

$$\mathcal{L}(X) = \frac{1}{n} \sum_{i=1}^n \prod_{j=1}^K (1 - s_{ij}). \quad (5)$$

This function is differentiable and can be minimized with standard gradient descent algorithms. We employ the *Adam* [19] optimizer, which is a common choice for training neural network models and which we observed to perform better than simple stochastic gradient descent. In fact, our model can be interpreted as a single linear neural network layer, where  $S$  constitutes the layer output for a given input  $X$  and  $W$  and  $b$  correspond to the layer's trainable parameters. This further allows our model to be combined and trained end-to-end with an autoencoder model to detect clusters in a learned feature space.

Additionally, we can perform gradient updates with small mini-batches sampled from the whole dataset, such that the model only operates on a small subset of the whole dataset at any given time. This leads to a constant-size memory footprint and allows our model to potentially scale to arbitrarily-sized datasets. The only requirement is that mini-batches are still representative of the structure of the whole dataset, which is usually the case if they are sufficiently large and sampled uniformly at random. As a further advantage, our model can be trained very efficiently on a GPU.

Finally, we wish to point out that similar activation and loss functions as above were used by an existing approach [20] to line detection in images. However, the distance function considered by the authors does not rely on distance normalization by the standard deviation, and the loss function uses an additional exponent. Further, optimization is performed using a custom training algorithm, and applications are limited to line detection based on 2-dimensional image coordinates.

### IV. EXPERIMENTS

We evaluate our model, which we will refer to as *HoughNet* in the following, on a wide array of different datasets with different characteristics and compare it against 5 established subspace clustering algorithms, each based on different assumptions and relying on different techniques. Most closely

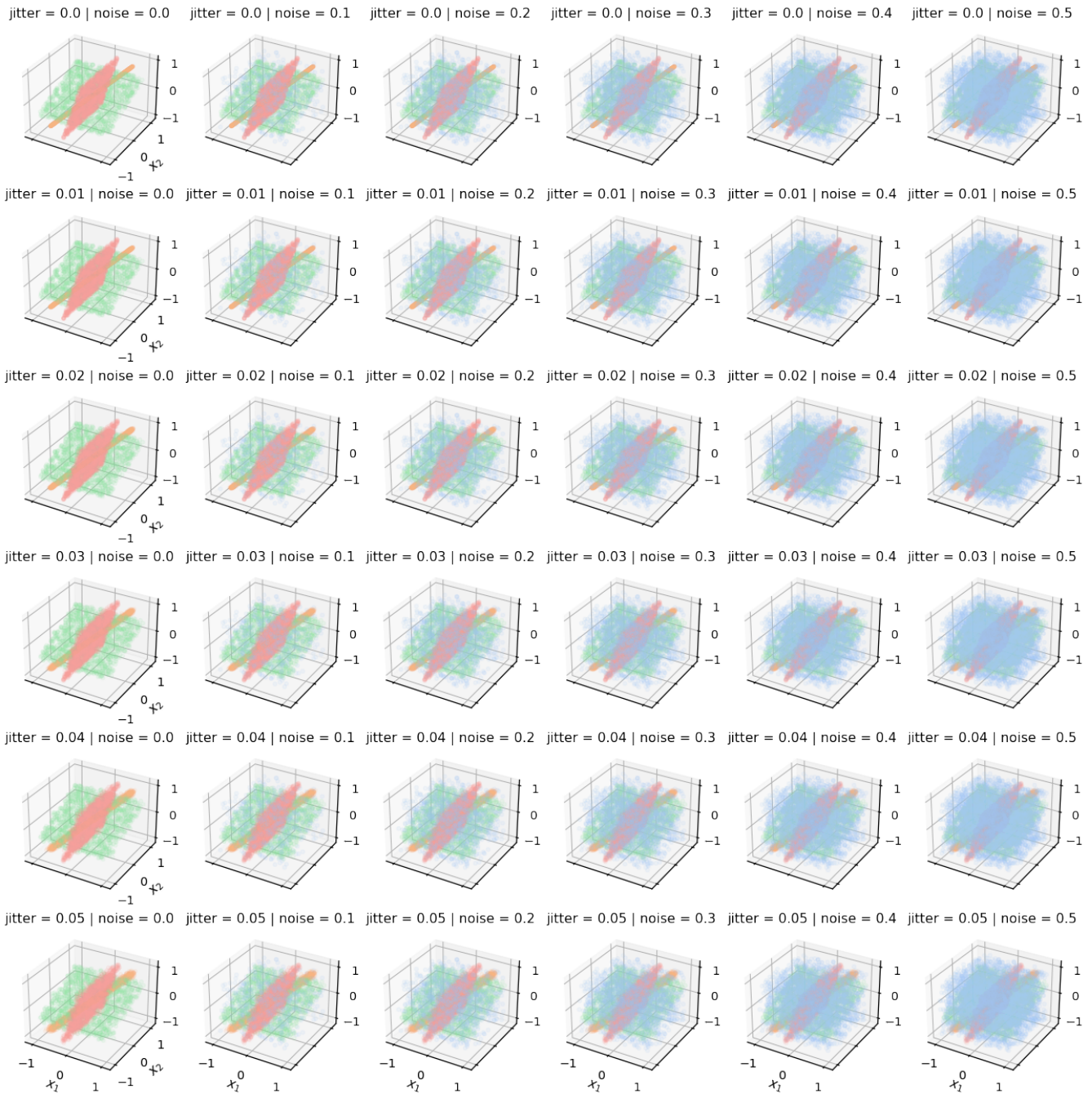


Fig. 2: 2P1L dataset with varying amounts of jitter and noise.

related to our approach are *CASH* [3] and *k-Planes* [4]. We further compare against *ORCLUS* [5], *4C* [6], and *SSC* [15]. Our model was implemented using *PyTorch* [21] and was trained on GPU. All remaining models were trained on CPU. *SSC* was implemented using the *CVXPY* library [22]. For *CASH*, *ORCLUS*, and *4C*, we rely on the *ELKI* framework [23]. All experiments were performed on a machine with 32 CPU cores, 128GB RAM and a single NVIDIA GeForce RTX 2080 Ti GPU with 12GB GPU memory. In the spirit of

reproducible research, we make our code publicly available <sup>1</sup>.

#### A. Datasets

To gain deeper insights into the capabilities *HoughNet* in different environments, we construct different synthetic datasets with different characteristics. The first base dataset, 2P1L, consists of two 2-dim. and one 1-dim. subspace in a 3-dim. ambient space. Each cluster contains 1000 points,

<sup>1</sup><https://github.com/buschju/houghnet>

and the data is normalized to the interval  $[-1, 1]^3$ . Different cluster dimensions and intersections between clusters already pose challenges to subspace clustering algorithms. Since real-world data usually does not fit exactly into subspaces, we add different types of errors that make the clustering task more challenging. First, we add jitter to make the clusters more blurry and increase the size of their intersections. In total, six increasing levels are considered. Secondly, we add increasing amounts of uniform noise, considering six different levels ranging from 0% to 50%. In the most extreme case, half of the dataset consists of random noise. The resulting grid of 36 datasets is visualized in Figure 2.

To investigate the ability of *HoughNet* and the competing methods to detect subspace clusters in higher-dimensional spaces, we additionally construct three high-dimensional datasets. In particular, we consider a 100-dim. ambient space and plant three randomly sampled subspace clusters with different dimensionalities. We refer to this base dataset as *HighD* and consider three different settings: (1) Three low-dim. clusters, (2) three medium-dimensional clusters, and (3) three high-dimensional clusters. The cluster dimensionalities are indicated in Table I. As for the first dataset, we sample 1000 points per cluster and normalize the data to  $[-1, 1]^{100}$ . To make the clustering task more challenging, we add random jitter of up to 0.03 and 30% noise.

In total, we consider 39 datasets, each with different characteristics. We restrict ourselves to synthetic datasets to get a more solid understanding of the capabilities of different algorithms in a controlled environment and leave a more extensive evaluation on further synthetic and real-world datasets for future work.

### B. Experimental Setup

To ensure a meaningful and fair comparison, we follow a rigorous evaluation protocol. Clustering performance is determined w.r.t. the ground truth labels and measured in terms of accuracy (ACC), adjusted Rand index (ARI), and normalized mutual information (NMI). We only evaluate whether actual cluster points are assigned correctly and ignore noise points during evaluation. If an algorithm predicts a noise label for some points, we collect these points in an additional noise cluster. For each algorithm, the optimal hyper-parameters are determined using a grid search for each dataset individually. All details can be found in the provided code. Some data-specific hyper-parameters are fixed for all algorithms: The maximum cluster dimensionality is set to  $d - 1$ , the maximum jitter is set to 0.05, and the correct number of clusters is supplied to all algorithms except *4C*. For *HoughNet*, we use a fixed batch size of 100 and a fixed learning rate of 0.1 and only optimize  $\lambda$  over  $\{0.1, 0.2, \dots, 0.9\}$ . *HoughNet*, *ORCLUS*, and *k-Planes* are executed 10 times with different random initializations, and the mean and standard deviation over all runs are reported for all scores. Since the remaining algorithms are deterministic, we can only report a single score.

### C. Clustering Performance

Clustering performance on 2P1L is reported in Figure 3. It can be observed that *HoughNet* consistently provides the highest clustering performance, especially in settings with higher noise levels. Clustering results are highly stable for lower noise levels. With increasing noise, variance in the clustering performance increases. Performance slowly degrades as jitter is increased. *CASH* performs notably worse, even when no jitter or noise is present in the data. Performance degrades further with increasing noise and especially jitter. This indicates that optimization in data space as performed by *HoughNet* is much more stable and more robust to noise and especially jitter, compared to dense region detection in parameter space. Similarly, as for *HoughNet*, the performance of *k-Planes* slowly degrades as jitter is increased. However, *k-Planes* is much more vulnerable to noise. Since *k-Planes* uses the same clustering model as *HoughNet*, this indicates that the *k*-Means-like optimization procedure is much less stable and robust to noise than gradient-based minimization of the differentiable loss function employed by *HoughNet*. Base performance of *ORCLUS* is very low, but the algorithm is very robust to jitter and also relatively robust to noise. *4C* provides a similar base performance as *CASH* but is very robust to noise as long as jitter is low. With increasing jitter, performance degrades quickly. *SSC* provides the lowest base performance among all competitors but surprisingly seems to provide more stable results for higher values of jitter. A possible reason for this behavior is that *SSC* struggles to connect smaller sub-clusters with spectral clustering and that these clusters can be linked in some cases with jitter or noise points.

In summary, *HoughNet* provides comparatively high clustering performance in all considered settings and, compared to competing algorithms, is highly robust to jitter and noise, and provides stable performance over different random initializations.

### D. Clustering in High-Dimensional Spaces

In many applications, subspace clusters are hidden in a higher-dimensional ambient space. To evaluate clustering performance in higher-dimensional settings, we further compare all algorithms on the three variants of the *HighD* dataset presented in Section IV-A. Rather surprisingly, *SSC* is able to recover the subspace clusters perfectly in all settings. On the other hand, it needs to run for over 12 hours in one setting, while all other algorithms, except *CASH*, provide results within seconds. Results for *CASH* could not be reported since the algorithm did not terminate after more than 24 hours, even after drastically reducing the search depth and increasing the minimum number of points per cluster. Among the algorithms apart from *SSC*, *HoughNet* provides the best performance by a considerable margin. While it is not able to match the performance of *SSC*, it can still provide reasonable clustering performance and can scale to large datasets that can not be processed within reasonable resource limits with *SSC*.

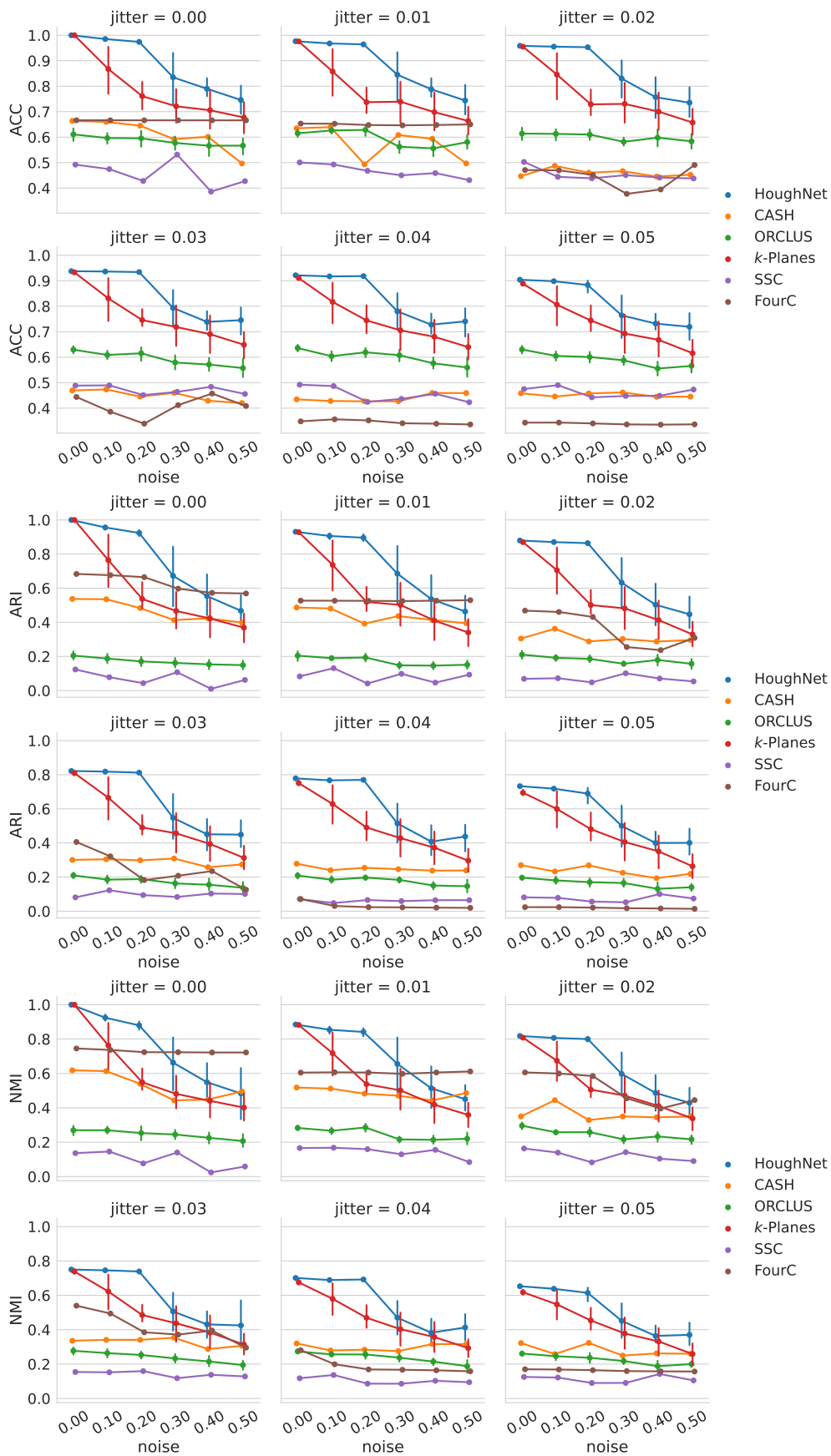


Fig. 3: Clustering results on the 2P1L dataset under varying jitter and noise.

### E. Convergence and Anytime Clustering

While *HoughNet* provides high clustering performance, we additionally wish to examine its scalability and speed of convergence. To this end, we construct two additional datasets based on 2P1L and HighD with jitter 0.03 and 20% noise, where we sample 10,000 points per cluster. For HighD, we generate 3 clusters with dimensions 6, 8, and 10. We compare against *k-Planes*, since both algorithms rely on iterative optimization to fit hyperplane clusters. Again, we run both algorithms with 10 different random initializations and report the mean performance over all runs. In Figure 4, it can be observed that *HoughNet* almost converges already after a single training epoch, i.e., after iterating over the whole dataset a single time. This happens within less than a second. Subsequent training epochs can improve performance only slightly, and 10 epochs take only a few seconds. We attribute the quick convergence to the fact that the model performs multiple update steps on different mini-batches during each epoch. These mini-batches allow our model to observe and update based on different varieties of the data within a single training epoch. Training can, in principle, be stopped at any time, even during an epoch. *HoughNet* is thus potentially very well suited for anytime subspace clustering as well. *k-Planes* on the other hand, requires much more time per iteration since it needs to compute an eigenvector for each cluster. Convergence is also much slower. On the HighD dataset, performance even drops initially.

### V. CONCLUSION

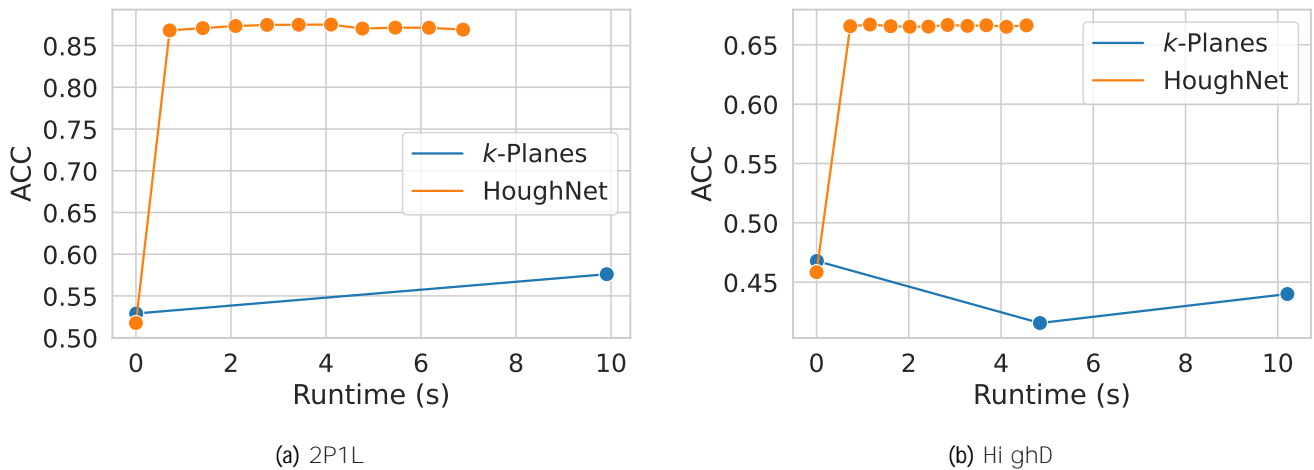
We presented a novel subspace clustering algorithm that fits affine hyperplanes to a given arbitrary-dimensional dataset using gradient updates from a differentiable loss function. This procedure can be interpreted as an implicit Hough transform, where hyperplanes are fitted directly in the data space rather than performing a dense region search in parameter space. Compared to existing approaches, our model is highly robust to jitter and noise, is highly scalable, can be efficiently trained on a GPU, converges fast and is amenable to anytime settings, and performs reasonably well even in high-dimensional spaces. These properties were empirically observed on 39 different synthetic datasets following a rigorous evaluation protocol. In future work, we plan to improve performance on high-dimensional datasets by reducing the dimensionalities of the hyperplanes to better adapt to lower-dimensional subspaces. We further plan to extend our model with a convolutional autoencoder to evaluate it on image clustering benchmarks and investigate its capability to cluster out-of-sample data.

### REFERENCES

- [1] H.-P. Kriegel, P. Kröger, and A. Zimek, "Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering," *Acm transactions on knowledge discovery from data (tkdd)*, vol. 3, no. 1, pp. 1–58, 2009.
- [2] R. Vidal, "Subspace clustering," *IEEE Signal Processing Magazine*, vol. 28, no. 2, pp. 52–68, 2011.
- [3] E. Achtert, C. Böhm, J. David, P. Kröger, and A. Zimek, "Robust clustering in arbitrarily oriented subspaces," in *Proceedings of the 2008 SIAM International Conference on Data Mining*. SIAM, 2008, pp. 763–774.
- [4] P. S. Bradley, O. L. Mangasarian, and P. Pardalos, "k-plane clustering," 1999.
- [5] C. C. Aggarwal and P. S. Yu, "Finding generalized projected clusters in high dimensional spaces," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 70–81.
- [6] C. Böhm, K. Kailing, P. Kröger, and A. Zimek, "Computing clusters of correlation connected objects," ser. SIGMOD '04. New York, NY, USA: Association for Computing Machinery, 2004, p. 455–466. [Online]. Available: <https://doi.org/10.1145/1007568.1007620>
- [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 *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, ser. KDD'96. AAAI Press, 1996, p. 226–231.
- [8] E. Achtert, C. Böhm, H. P. Kriegel, P. Kröger, and A. Zimek, "Robust, Complete, and Efficient Correlation Clustering," *Proceedings of the Seventh SIAM International Conference on Data Mining*, pp. 413–418, 2007.
- [9] E. Achtert, C. Böhm, P. Kröger, and A. Zimek, "Mining hierarchies of correlation clusters," in *18th International Conference on Scientific and Statistical Database Management (SSDBM'06)*, 2006, pp. 119–128.
- [10] E. Achtert, C. Böhm, H.-P. Kriegel, P. Kröger, and A. Zimek, "On exploring complex relationships of correlation clusters," in *19th International Conference on Scientific and Statistical Database Management (SSDBM 2007)*. IEEE, 2007, pp. 7–7.
- [11] S. Goebel, X. He, C. Plant, and C. Böhm, "Finding the optimal subspace for clustering," in *2014 IEEE International Conference on Data Mining*, 2014, pp. 130–139.
- [12] A. Tung, X. Xu, and B. Ooi, "Curler: Finding and visualizing nonlinear correlated clusters." 01 2005, pp. 467–478.
- [13] E. Achtert, H. Peter Kriegel, and A. Zimek, "Deriving quantitative models for correlation clusters," in *in Proc. KDD, 2006*, pp. 4–13.
- [14] M. A. X. Hünemörder, D. Kazempour, P. Kröger, and T. Seidl, "Sidekick: Linear correlation clustering with supervised background knowledge," in *Similarity Search and Applications*, G. Amato, C. Gennaro, V. Oría, and M. Radovanović, Eds. Cham: Springer International Publishing, 2019, pp. 221–230.
- [15] E. Elhamifar and R. Vidal, "Sparse subspace clustering: Algorithm, theory, and applications," 2013.
- [16] C. You, D. Robinson, and R. Vidal, "Scalable sparse subspace clustering by orthogonal matching pursuit," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3918–3927.
- [17] J. Busch, E. Faerman, M. Schubert, and T. Seidl, "Learning self-expression metrics for scalable and inductive subspace clustering," *NeurIPS 2020 Workshop: Self-Supervised Learning - Theory and Practice*, 2020.
- [18] P. Ji, T. Zhang, H. Li, M. Salzmann, and I. Reid, "Deep subspace clustering networks," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, 2017, pp. 23–32.
- [19] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations (ICLR)*, 2015.
- [20] J. Basak, "Learning hough transform: A neural network model," *Neural Computation*, vol. 13, no. 3, pp. 651–676, 2001. [Online]. Available: <https://doi.org/10.1162/089976601300014501>
- [21] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga et al., "Pytorch: An imperative style, high-performance deep learning library," *Advances in Neural Information Processing Systems*, vol. 32, pp. 8026–8037, 2019.
- [22] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *Journal of Machine Learning Research*, vol. 17, no. 83, pp. 1–5, 2016.
- [23] E. Schubert and A. Zimek, "Elki: A large open-source library for data analysis - elki release 0.7.5 "heidelberg"," *ArXiv*, vol. abs/1902.03616, 2019.

Cluster Dimensions	Model	ACC	ARI	NMI	Runtime (s)
[6, 8, 10]	HoughNet	90.53 ± 14.08	86.38 ± 19.61	90.54 ± 13.05	1.76 ± 0.09
	ORCLUS	56.71 ± 15.24	37.43 ± 24.70	45.68 ± 30.27	55.14 ± 2.59
	<i>k</i> -Planes	47.43 ± 8.21	11.75 ± 9.83	10.95 ± 8.53	7.15 ± 1.76
	SSC	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	13,659.92 ± 0.00
	4C	33.43 ± 0.00	0.00 ± 0.00	0.40 ± 0.00	10.69 ± 0.00
[48, 50, 52]	HoughNet	96.05 ± 3.83	89.31 ± 9.62	88.21 ± 9.16	4.20 ± 0.27
	ORCLUS	66.67 ± 0.00	55.09 ± 3.67	69.22 ± 7.02	61.38 ± 2.32
	<i>k</i> -Planes	36.34 ± 1.58	0.46 ± 0.43	0.47 ± 0.39	5.93 ± 1.15
	SSC	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	11,040.02 ± 0.00
	4C	33.33 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	10.49 ± 0.00
[90, 92, 94]	HoughNet	82.48 ± 7.39	61.15 ± 13.31	68.38 ± 11.18	8.54 ± 0.22
	ORCLUS	66.67 ± 0.00	57.13 ± 0.00	73.37 ± 0.00	66.40 ± 2.89
	<i>k</i> -Planes	34.47 ± 0.50	0.01 ± 0.05	0.07 ± 0.04	5.55 ± 1.20
	SSC	100.00 ± 0.00	100.00 ± 0.00	100.00 ± 0.00	45,694.31 ± 0.00
	4C	33.33 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	10.59 ± 0.00

TABLE I: Clustering performance on the Hi ghD dataset with different cluster dimensionalities.

Fig. 4: Convergence of *HoughNet* and *k*-Planes on different datasets.



## Chapter 8

# OAB - An Open Anomaly Benchmark Framework for Unsupervised and Semisupervised Anomaly Detection on Image and Tabular Data Sets

This chapter consists of a preprint version of the following publication:

©2021 IEEE. Reprinted, with permission, from Andreas Lohrer, Jan Deller, Maximilian Hünemörder, and Peer Kröger. “OAB - An Open Anomaly Benchmark Framework for Unsupervised and Semisupervised Anomaly Detection on Image and Tabular Data Sets”. In: *2021 International Conference on Data Mining Workshops (ICDMW)*. 2021, pp. 991–1000. DOI: 10.1109/ICDMW-W5-34-33.2021.00129

**Statement of Originality:** This idea originated in conversations between **M. Hünemörder** and A. Lohrer. The proof of principle and experiments were designed, implemented, and conducted as part of J. Deller’s bachelor thesis, supervised by A. Lohrer. The manuscript was adapted and extended from the bachelor thesis in a collaboration of A. Lohrer, J. Deller and **M. Hünemörder** with guidance from P. Kröger.

- **Conception:** **Hünemörder** (Lead), Lohrer (Lead), Deller (Support)
- **Planning:** Lohrer (Lead), Deller (Lead), **Hünemörder** (Support)
- **Execution:** Deller (Lead), Lohrer (Support)
- **Manuscript:** Lohrer (Lead), Deller (Lead), **Hünemörder** (Support), Kröger (Support)

# OAB - An Open Anomaly Benchmark Framework for Unsupervised and Semisupervised Anomaly Detection on Image and Tabular Data Sets

Andreas Lohrer †  
Information Systems and Data Mining  
Kiel University  
24118 Kiel, Germany  
Email: alo@informatik.uni-kiel.de  
ORCID: 0000-0001-7834-301X

Jan Deller †  
Kiel University  
24118 Kiel, Germany  
Email: jandeller@t-online.de  
† first authors

Maximilian Hünemörder  
Information Systems and Data Mining  
Kiel University  
24118 Kiel, Germany  
Email: mah@informatik.uni-kiel.de

Peer Kröger  
Information Systems and Data Mining  
Kiel University  
24118 Kiel, Germany  
Email: pkr@informatik.uni-kiel.de  
ORCID: 0000-0001-5646-3299

**Abstract**—We introduce OAB, an Open Anomaly Benchmark Framework for unsupervised and semisupervised anomaly detection on image and tabular data sets, ensuring simple reproducibility for existing benchmark results as well as a reliable comparability and low-effort extensibility when new anomaly detection algorithms or new data sets are added. While making established methods of the most popular benchmarks easily accessible, OAB generalizes the task of un- and semisupervised anomaly benchmarking and offers besides commonly used benchmark data sets also semantically meaningful real-world anomaly data sets as well as a broad range of traditional and state-of-the-art anomaly detection algorithms. The benefit of OAB for the research community has been demonstrated by reproducing and extending existing benchmarks to new algorithms with very low effort allowing researchers to focus on the actual algorithm research.

**Index Terms**—Unsupervised and Semisupervised Anomaly Detection, Reproducibility, Benchmark, Evaluation, Datasets

## I. INTRODUCTION

The increasing digitalization of processes in various domains like e.g. industrial automation, healthcare, mobility and others causes a large-scale volume of structured and unstructured data accommodating valuable knowledge and potential for process optimizations. One of the most interesting and similarly most domain beneficial kinds of knowledge discovery is the detection of abnormal patterns, also known as anomalies or outliers. “An outlier is an observation which deviates so much from the other observations as to arouse suspicions that it was generated by a different mechanism.” [1] The research field of anomaly detection demonstrates its relevance by addressing many applications like e.g. fraud detection, network security

issues, quality monitoring, etc. and thus it is not unusual that still numerous unsupervised and semisupervised anomaly detection algorithms [2], [3] get continuously developed and published. This advantageous situation of having such a versatile range of algorithms available becomes challenging every time when the question arises which algorithm performs better. There are still publications with only sparse or even no possibility for well reproducible results [4], missing an accurate description of hyperparameters, preprocessing steps, accessible data sets, applied sampling strategies, state of the art algorithms or meaningful standard evaluation metrics. Furthermore, in numerous anomaly detection benchmarks and papers (e.g. [5]–[9], etc.) anomaly data sets are synthetically generated by downsampling individual classes of actual classification data sets as anomalies instead of using real-world anomaly data sets containing semantically meaningful anomalies such as e.g. aerospace hardware parts on martian surface images [10]. Existing anomaly benchmarks [5], [6], [11] address all these requirements with different focus and scope but when researchers evaluate new algorithms they often cannot reproduce the results of competing algorithms. So they tend to conduct their experiments still individually instead of using existing benchmarks leading to not actually comparable results which makes truly well performing approaches hard to identify in the research field. Especially for anomaly detection algorithms developed for image and tabular data there is so far no standardized benchmark simultaneously ensuring (1) reproducibility, (2) comparability and (3) low-effort extensibility to new algorithms and data sets.

Thus we introduce with this work-in-progress the approach

of OAB - An **O**pen **A**nomaly **B**enchmark framework for image and tabular data sets supporting benchmarks for unsupervised and semisupervised anomaly detection algorithms and data sets. OAB covers a broad range of traditional and deep anomaly detection algorithms and provides next to most common anomaly benchmark data sets also real-world anomaly data sets. Moreover, OAB generalizes the task of un- and semisupervised anomaly benchmarks and sets an equally strong focus to the reproducibility and comparability of its results in order to support the requirements of the *Machine Learning Reproducibility Checklist* [4]. Different to other comparable benchmarks the OAB framework allows low-effort extensions to new algorithms and data sets enabling researchers to keep the focus on the actual algorithm research instead of trying to reproduce results and conducting their own individual experiments. The main contributions of this work can be summarized as follows:

- Simple reproducibility, comparability and extensibility for anomaly benchmark results when adding new algorithms or new data sets.
- OAB supports unsupervised and semisupervised benchmarks for image and tabular data sets by covering related traditional and deep anomaly detection algorithms.
- OAB offers reproducible sampling and contamination strategies and provides commonly used anomaly benchmark data sets as well as various real-world anomaly data sets.
- A well documented open Python library OAB with hands-on examples enabling the research community for its practical use.

The remainder is structured as follows. Section II describes related anomaly benchmarks. In Section III, we describe our framework for an Open Anomaly Benchmark and evaluate its performance w.r.t. the stated contributions in Section IV. Section V concludes the paper and proposes further ideas for future work.

## II. RELATED WORK

In one of the first benchmarks for outlier detection the authors Campos et al. [5] present a study of unsupervised anomaly detection, in which an extensive collection of 23 data sets gets featured primarily taken from the UCI repository [12]. These data sets are all tabular, considering that ALOI is originally an image data set from which only premade image features are used, and are mostly rather low-dimensional and small in respect to the amount of samples. Considering the problem that originally most UCI data sets are intended for classification and sometimes clustering, Campos et al. especially highlight a group of data sets that are semantically meaningful for outlier detection in comparison to the common procedure of downsampling just any classification data set. These semantically meaningful data sets include medical data, i.e. normal and sick patients, mail and webdata, i.e. filtering out spam and ads, stamps (forged or real), and trees (normal or diseased). The benchmark only investigates  $k$ -nearest neighbor based outlier detection algorithms, since these were popular

at the time and are often still used as baseline models in current research (often represented by the Local Outlier Factor (LOF) algorithm). An advantage of this restriction is that all of these methods mainly depend on the size of the neighborhood that got analyzed at the beginning by the hyperparameter  $k$ . These methods are therefore easily compared in respect to this parameter. Campos et al. chose common evaluation measures for outlier detection, e.g. the Area under the Curve of the Receiver-Operating Characteristic (AUC ROC or ROC AUC) and Average Precision (AP), but highlight also a version of Precision-at- $n$  ( $P@n$ ), which was adjusted for chance (Adjusted  $P@n$ ) similarly to AP (Adjusted  $AP@n$ ). In order to prepare the data sets, different preprocessing steps are discussed in detail and listed for each data set. Campos et al. utilize downsampling for classification data sets and in case of duplicate samples a data set with and without duplication removal is provided. Categorical attributes get transformed into numerical data using either 1-of- $n$  or Inverse Document Frequency (IDF). Furthermore, normalization gets considered as well as missing values if needed. Campos et al. published their implementations<sup>1</sup> by a collection of Python- and R-Scripts together with the used version of the Java-based KDD-Library ELKI [13].

The benchmark of Emmott et al. [6] also focuses like Campos et al. on unsupervised anomaly detection for tabular data. Therefore, 19 data sets are retrieved from the UCI repository [12]. In total 25685 child anomaly detection data sets are then sampled from these "mothersets" with variation in four dimensions. (1) Point difficulty measures how hard it is to distinguish normal points from anomalies, where the difficulty of an individual point is computed using Kernel Logistic Regression. (2) Semantic variation assesses how different the anomalies are from each other considering their clusteredness. (3) Contamination rate indicates what percentage of sampled data points are anomalous and (4) feature relevance/irrelevance adds additional irrelevant features to test how well an algorithm can handle these. Categorical features get discarded in advance from the mothersets, and all remaining features get normalized. Additionally, labels "normal" and "anomalous" are assigned to the data points.

A variety of 8 algorithms are tested on the resulting child data sets, including density-based algorithms like Ensemble Gaussian Mixture Model, model-based algorithms like One-Class Support Vector Machine, nearest-neighbors-based algorithms like LOF and projection-based ones like Isolation Forest. For the evaluation, the metrics ROC AUC and AP are used and their significance, and thus also the benchmark itself, gets evaluated by an additional hypothesis test.

All benchmarks are implemented in R and available<sup>2</sup> for download. As pointed out in the project's README, the documentation is unfortunately sparse. This published code allows to reproduce data preprocessing and sampling, the experiments and their results are not included however.

<sup>1</sup><https://www.dbs.ifi.lmu.de/research/outlier-evaluation/DAMI>

<sup>2</sup><http://ir.library.oregonstate.edu/xmlui/handle/1957/59114>

One of the most recent comprehensive benchmarks for anomaly detection has been published by Domingues et al. [11]. Like the previously introduced benchmarks the focus is also set to unsupervised anomaly detection on tabular data. The collection of 15 data sets in total is composed out of 12 public data sets from the UCI [12] and OpenML [14] repositories and out of 3 proprietary data sets from the traveling industry domain. In order to benchmark for production relevant models by unseen data Domingues et al. apply train-test-splits in the unsupervised setting with an equal proportion of outliers in the train and test set. In extensive benchmarks 14 anomaly detection algorithms from different groups like probabilistic-, neighbourhood-, isolation-, SVM- or neural-network-based algorithm groups are evaluated in detail. The performance of these algorithms gets evaluated on test set samples by the metrics ROC AUC and Area Under the Precision Recall Curve (AUPRC). Furthermore, Domingues et al. conduct algorithm complexity analyses related to the time for training and prediction, memory usage and noise stability in dependence to the total number of samples and features based on synthetically generated data sets. In the preprocessing steps of the benchmark the data sets get normalized and the categorical data is one-hot-encoded allowing the algorithms to utilize the complete information from the data sets. Furthermore supporting reproducibility, the hardware setup, hyperparameters as well as the implementation languages like R, Python and Matlab are listed for each algorithm, but the benchmark scripts itself are not published avoiding a simple extension to new algorithms and data sets.

Besides the aforementioned well known approaches there are also further related benchmarks in the unsupervised setting like Goldstein et al. [15] which focuses on traditional algorithms and Lu et al. [16] evaluating dependency-based anomaly detection algorithms. Although each of them demonstrates various strengths, none of them allows the evaluation of most recent anomaly detection approaches in the un- and semisupervised setting on tabular as well as on image data sets and ensures a simple reproducibility, comparability and extensibility when new algorithms or new data sets should be added.

### III. OPEN ANOMALY BENCHMARK (OAB)

This section describes the components of the Open Anomaly Benchmark framework covering requirements for image and tabular data sets, a selection of anomaly detection algorithm groups, most suitable evaluation metrics and methods ensuring reproducibility.

#### A. Benchmark Data Sets for Anomaly Detection

Ideally, actual anomaly data sets are used to benchmark anomaly detection algorithms, i.e., data sets with a semantic notion of *normal* and *anomalous* data points. However, real anomaly data sets are rare, especially in image detection, and might not have the characteristics authors are interested in when evaluating their new algorithm, e.g., with regards to number of observations or dimensionality. To remedy

this, classification and regression data sets are frequently transformed into anomaly detection data sets [6], [11]. In classification data sets, this assumes that a downsampled generating process of some class resembles an anomaly-generating process, which is not necessarily the case. However, as classification and regression data sets are frequently used to assess the performance of an anomaly detection algorithm, they are also provided by OAB and included in this paper. Note that OAB is not limited to working with the data sets provided by it, instead, also other data sets can easily be loaded making benchmarks with OAB simply extensible to own data sets.

#### B. Benchmark Algorithms for Anomaly Detection

This paper measures the performance of a variety of anomaly detection algorithms. OAB is in no way limited to working with these algorithms - it provides sampled data and an evaluation procedure for own algorithms to allow new approaches to be tested and replicated.

There are a variety of anomaly detection algorithms, distinguishing themselves in their assumptions about how normal data and anomalous data are different. We distinguish between traditional unsupervised, traditional semisupervised, and deep anomaly detection algorithms.

1) *Unsupervised algorithms*: The unsupervised algorithms receive a test data set as input and calculate anomaly scores of these test data points.

*Nearest neighbour-based approaches* (c.f. [6], [17]) assume that compared in its nearest neighbors, normal and anomalous data points behave differently in a specific way.

*Isolation-based approaches* aim to isolate anomalies and compute an anomaly score based on how difficult it is to isolate a data point [18].

*Reconstruction-based approaches* first reduce the dimensionality of a data point and then try to reconstruct it from this reduced-dimensional representation.

2) *Semisupervised algorithms*: The semisupervised algorithms are in contrast to unsupervised algorithms first trained on a clean training set. In the inference phase, they calculate an anomaly score for test points.

*One-class approaches* aim to learn a boundary around the normal training data points and assign anomaly scores based on this boundary.

*Isolation-based approaches* can also be used in the semisupervised setting. The rules to isolate data points are learned from the training set, and applied to data points from the test set [18].

*Reconstruction-based approaches* can be used similarly to isolation-based approaches, i.e., the dimensionality reduction and increase procedures are learned from the training data and applied to the test data.

3) *Deep learning algorithms*: The deep learning algorithms exist for both unsupervised and semisupervised anomaly detection and can be differentiated based on how directly the

deep part of the model is connected to anomaly detection. Among others, Pang et al. [2] identify the following two classes of algorithms:

*Deep learning for feature extraction* means that final layers for classification do not exist. The layers of such a neural network are only used to extract features which are fed into a traditional model.

*Learning feature representations of normality* comprise algorithms that are trained on some surrogate learning task that is not directly aiming for anomaly detection. Based on this surrogate task, a heuristic is used to calculate anomaly scores.

A core strength of OAB is that it is not limited to these algorithms. It allows easily testing new algorithms on the data sets provided and own data sets ensuring extensibility.

### C. Evaluation Metrics

As the evaluation metrics of an anomaly benchmark framework need to be meaningful for the related task of anomaly detection several requirements need to be fulfilled. In the unsupervised setting there are usually no external labels available for evaluation. Thus, Marques et al. introduced IREOS [19] as an internal evaluation metric for the unsupervised setting. However, since the work of Aggarwal [20] states that internal validity measures are only rarely used for outlier detection and that it is more reasonable to use external validity measures in this field, OAB follows this recommendation and selects evaluation measures requiring the availability of external labels. Since the expressiveness of an evaluation metric is dependent on the used data set as well as on the task which an anomaly detection algorithm should perform there is not only one metric which should be taken into consideration. Hence the OAB framework supports the following set of evaluation metrics.

1) *ROC AUC*: The Receiver Operating Characteristic (ROC) [21] is represented by a curve in a chart defined by the true positive rate (TPR) of correctly detected outliers on the y-axis and by the false positive rate (FPR) of incorrectly detected outliers on the x-axis. This usually monotonously increasing ROC curve covers the full range of possible outlier decision thresholds  $\tau$  with  $\tau \in [0; 1]$ . The Area Under that Curve (AUC), allows to represent this dependency by a single value between 0 and 1 describing the performance of the evaluated anomaly detection algorithm as ROC AUC. The maximum is a ROC AUC of 1.0 having a TPR of 1.0 and a FPR 0.0 from the beginning. The normalization of the TPR and FPR with the number of TP and FP respectively ensures ROC AUC still to be an expressive evaluation metric even when the ratio between normal and abnormal samples is imbalanced as it is usually the case for anomaly detection data sets. [5] In case the primary objective of the given anomaly detection task is to avoid false positives (false alarms) the ROC AUC would be a suitable evaluation measure.

2) *AUPRC*: As the name of the Precision-Recall Curve reveals it is defined by the Precision on the y-axis and the Recall on the x-axis for any possible decision threshold  $\tau$ .

As for ROC AUC the Area Under the Precision-Recall Curve (AUPRC) describes the performance of an anomaly detection algorithm by a single value between 0 and 1 having the same maximum, but differently to ROC AUC the AUPRC is not necessarily monotonously increasing. If an anomaly detection task has the goal to avoid false positives (false alarms) as well as false negatives (missed alerts) then the AUPRC would be a reasonable evaluation metric.

3) *P@n*: The evaluation metric Precision@n [22] is conceived for benchmarking anomaly detection algorithms which focus on the evaluation of just the top  $n$  (ranked) outlier scores. This can be reasonable especially for large data sets with a relative high percentage of anomalies or in cases in which not more than  $n$  samples can or should be handled. Since the definition of  $n$  is crucial for meaningful P@n-results the availability of external anomaly labels is required. One possibility is to set  $n$  according to the total number of known anomalies in the data set. Since the precision is calculated with this metric, P@n can be used for anomaly detection tasks having their focus on the avoidance of false alarms.

4) *AdjP@n*: The Adjusted-Precision@n [5] addresses benchmark settings in which the outlier scores and thus also the percentage of total outliers may vary. Therefore, the previously introduced P@n metric gets adjusted for chance by aligning different outlier scores by their expected outlier score value.

5) *AP*: The Average Precision [23] evaluation metric is a further possibility to calculate the AUPRC. There exist several variants but since OAB relies on the `average_precision_score` of `sklearn` the weighted mean of precisions at each threshold is used. Equally to AUPRC, if an anomaly detection task needs to avoid false positives (false alarms) as well as false negatives (missed alerts) then the AP would be a suitable choice.

6) *AdjAP*: The Adjusted-Average-Precision [5] considers similarly to AdjP@n benchmark settings with varying outlier scores for an adjustment by chance, whereas the metric is calculated analogously.

7) *Friedman and Nemenyi tests*: The Friedman test [24] can be used to investigate whether or not all algorithms perform equally with regards to a metric, while the Nemenyi test [25] allows for a significance analysis of pair-wise performance comparisons.

### D. Reproducibility

The core strengths of the OAB framework are reproducibility and comparability of its benchmarks. Therefore, OAB divides the task of anomaly benchmarking into a set of generic subtasks, which can as part of a benchmark recipe easily be replicated and generically applied to unsupervised and semisupervised anomaly benchmarks. These subtasks are represented by a variety of benchmarking steps introduced in this section. These benchmarking steps are closely related to the process steps for knowledge discovery and data mining (KDD) introduced by Fayyad et al. [26] (cf. Fig. 1a)). Differently to [26] the step of Data Mining is in OAB further divided into the

steps Sampling, which also represents the splitting into train and test sets, and Anomaly Detection Algorithms, leading to a Performance Overview after Evaluation. Considering Fig. 1b), OAB provides reproducibility across the benchmarking steps, noting that for steps with grey background no user-related actions are required and steps with white parts allow the user to increase or decrease the Selection of data sets and algorithms.

1) *Sampling Strategy*: Similar to algorithms, sampling strategies also need to be differentiated between unsupervised and semisupervised. With the sampling strategies outlined below, we aim to achieve two goals in both settings: (1) As many data points as possible are included to ensure that the result is not achieved by chance, and (2) to get a more robust estimate, we sample multiple times from each data set where each sample is different from the others.

In the unsupervised setting, the sampling should return a set of data points and their respective labels to allow for evaluation. Among the set of data points, the number of anomalies is expected to be comparatively low, as normal data points dominate the set of data points. The main sampling parameters are the number of samples  $s_u$  and the contamination rate  $c$ .

*Target contamination rate*. The target contamination rate, i.e., the proportion of anomalous data points among all data points, is set to  $c = 0.05$  in our experiments, but a different target contamination rate can be set in OAB. The target contamination rate  $c$  is restricted to  $c \in (0, 1)$ .

*Number of samples*. The number of samples depends on the data set. In some data sets, the actual contamination rate is larger than the target contamination rate  $c$ , whereas in others, it is lower. Therefore, a procedure needs to be specified that calculates how many data points are sampled to ensure a contamination rate of 0.05. Emmott et al. [6] either subsample the normal data points if the original data's contamination rate is lower than the target contamination rate or subsample the anomalous data points in the other case. Our approach is similar to this and extends it in both providing a formalization and ensuring variance across samples. First, we calculate the maximum sample size restriction for normal and anomalous points  $s_n^{\max}$  and  $s_a^{\max}$  respectively. We denote the number of normal points in a data set by  $n_n$  and the number of anomalies by  $n_a$ .

$$s_n^{\max} = \left\lfloor \frac{n_n}{1-c} \right\rfloor \quad (1)$$

$$s_a^{\max} = \left\lfloor \frac{n_a}{c} \right\rfloor \quad (2)$$

If a data set consists of  $n_n = 950$  normal data points and the target contamination rate is  $c = 0.05$ , the sample size can at maximum be  $950/(1-0.05) = 1000$ . If it is larger than this, a contamination rate of 0.05 is not possible without duplicating normal data points. This is calculated with Equation 1. The same train of thought can be applied to anomalous data points, leading to Equation 2. The tighter condition, i.e., the minimum of both values, is the maximum sample size. To ensure variability both among normal and anomalous labels,

this maximum sample size is scaled with a factor  $f \in (0, 1)$  which we set to 0.9 to arrive at the actual sampling size in the unsupervised case  $s_u$ :

$$s_u = \left\lfloor f * \min \left\{ \frac{n_a}{c}, \frac{n_n}{1-c} \right\} \right\rfloor \quad (3)$$

*Number of sampling steps and random seed*. Finally, we sample a total of 10 times from a data set, and compute the average and standard deviation for each metric across these samples. For the first sample, the random seed is set to 42, and increased by 1 in each subsequent sampling step.

TABLE I  
PARAMETERS USED WHEN SAMPLING IN THE UNSUPERVISED SETTING.

Target contamination rate $c$	0.05
Downscaling factor $f$	0.9
Sampling size $s_u$	$\left\lfloor f * \min \left\{ \frac{n_a}{c}, \frac{n_n}{1-c} \right\} \right\rfloor$
Number of sampling steps	10
Initial random seed	42

These sampling parameters are summarised in Table I. OAB allows for sampling either by specifying the number of data points to sample and the contamination rate or by specifying the contamination rate and a scaling factor. In the latter case, the sampling size is automatically computed using Equation 3.

In the semisupervised setting, a training set as well as a test set and the corresponding labels for the test set are provided by the sampling procedure. The training set is usually clean, i.e., it consists only of normal data points. In the test set, the contamination rate does not have to be as small as in the semisupervised case, as the model is already trained when assessing its performance on the test set and the data points from the test set do not affect the anomaly scores of each other. For the toothbrush data set of MVTEC AD for example, the test set consists of 11 normal points and 29 anomalies [27], [28]. The parameters for sampling are therefore not specified by the number of samples and the contamination rate, but instead by the percentage  $train$  of normal data points used for training and the maximum contamination rate  $c_{test}^{\max}$  in the test set.

*Percentage of normal data points used for training*. The parameter  $train$  specifies the percentage of normal data points which are used for training. It is constrained to  $train \in (0.5, 1)$  as there should be more observations in the training set than in the test set, and  $1 - train$  is the part of the normal data points used for testing.  $train$  needs to balance two considerations. On the one hand, a larger training set allows the algorithm to see more normal data points which in turn allows it to better learn what normal data points look like. On the other hand, the training set should not be too large, as in this case the number of normal data points in the test set can become very small. If this is the case and the test set contains mainly anomalous points, it becomes more difficult to assess an algorithm's strength in distinguishing normal from anomalous points. A training set size of  $train = 0.7$  of the normal points is therefore chosen.

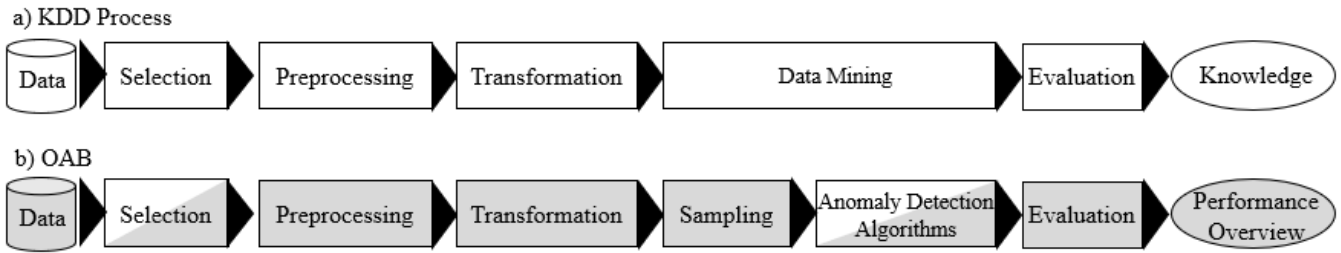


Fig. 1. Comparison of a) KDD process steps [26] and b) reproducible benchmarking steps of OAB (grey) and user-related steps (white).

*Maximum contamination rate.* The test set consists of both normal and anomalous data points. All remaining normal data points are used in the test set, and theoretically, all anomalous data points can be used as well. Note however that in some data sets, especially original classification data sets with multiple classes, this can lead to a large contamination rate in excess of 0.9. If the ratio of normal points in the test set is very small, the performance metric loses robustness. For this reason, the contamination rate in the test set is restricted by  $c_{test}^{max} \in (0, 1)$ , which is set to 0.5 in this benchmark.

TABLE II

PARAMETERS USED WHEN SAMPLING IN THE SEMISUPERVISED SETTING.

$train$	0.7
$c_{test}^{max}$	0.5
Number of samples	10
Initial random seed	42

Apart from the parameters specified here, the random seed and number of samples are the same as in the unsupervised setting. The parameters are summed up in Table II.

Note that in contrast to unsupervised sampling, semisupervised sampling does not need a scaling factor as the variability is ensured by splitting the normal data into different training and test sets in each sampling. In addition to the sampling described above, OAB also supports sampling using explicit training and test set sizes and allows for a contamination in the training set, which can be of use to assess an algorithm's robustness towards anomalies in the training set. In this benchmark however, the sampling above is used.

If data sets already have a native train-test split like MVTEC AD [27], [28], this split can also be used for sampling using OAB. This native split is also used in the experiments section of this paper.

2) *Preprocessing, Transformation and Parameters:* To ensure reproducibility, describing which data set is used and where it was downloaded from does not suffice. If preprocessing and transformation steps are applied, they have to be well-documented. In addition to that, also the training- and hyperparameters need to be tracked. Therefore, this section describes these steps in detail.

*Regression data sets.* While classification data sets can easily be transformed into anomaly detection data sets by specifying which labels are considered to be normal and which

anomalous, regression data sets need a different approach: The inter-quartile range (IQR) is calculated and instances  $x$  are considered as outliers in case of exceeding the range of  $[(Q1 - d * IQR); (Q3 + d * IQR)]$  where  $d$  is a data set specific factor (mostly  $d=1.5$ ).

*Missing values.* Algorithms considered here are not capable of working with data points with missing features. When missing values do occur, we follow Campos et al. [5] and delete an attribute if 10% or more of the instances do not have a value for this attribute. Otherwise only the related instances get removed.

*Duplicate data points.* In line with the preprocessing performed by Campos et al. [5], duplicate data points are removed.

*Categorical features.* While most features are numeric, some categorical features also exist. Campos et al. [5] propose two techniques to deal with categorical features, namely one-hot encoding and using the inverse document frequency of the attribute value as encoding. Emmott et al. [6] ignore categorical features. In this benchmark, categorical features are one-hot encoded.

*Parameters.* The OAB recipe also tracks training- and hyperparameters for reproducible algorithm runs. These are e.g. the number of epochs, learning rate, batch size, layer sizes, etc.

*Normalization.* Campos et al. [5] propose to use two data sets: One with attribute-wise linear normalization to values between 0 and 1 (including) and a second one without normalization. Domingues et al. [11] on the other hand standardize their features, i.e., scale them to mean 0 and unit standard deviation. For tabular data, attribute-wise scaling with `RobustScaler` of `sklearn` on all attributes is used. For image data, the typical machine learning scaling to values between 0 and 1 is used, i.e., all pixel values are multiplied by  $1/255$ .

All aspects described above are part of OAB and thus reproducible for each experiment.

#### IV. EXPERIMENTAL SETUP AND RESULTS

In this section, the performance of the OAB framework gets investigated. Therefore, a selection of tabular and image data sets is loaded, the data sets are sampled and the selected anomaly detection algorithms get benchmarked on these samples. The results can easily be replicated using the Google-Colabs from our github.

### A. Data sets

As described in Section III-A, OAB supports both real anomaly data sets as well classification and regression data sets transformed into anomaly data sets. The data sets used to obtain the benchmark results are presented in this section. They are naturally divided into tabular and image data sets. Real anomaly data sets, i.e., data sets that contain semantically meaningful anomalies, are marked with \*. In addition to these data sets, a variety of other data sets is also built in to OAB.<sup>3</sup> The following tabular data sets are used in this study. A brief description of their characteristics can be found in Table III.<sup>4</sup>

**Spambase\*** [12]. Emails divided into non-spam (normal) and spam (anomalous)

**Wilt\*** [12]. Segments of images divided into other land area (normal) and diseased trees (anomalous)

**NASA\*** [29]. NASA software for receiving and processing ground data divided into non-failing (normal) and with defects (anomalous)

**Anthyroid\*** [30]. Medical data, divided into healthy (normal) and two kinds of hypothyroidism (anomalous)

**Page-blocks** [12]. Blocks of a page layout divided into text blocks (normal) and non-text blocks (anomalous)

**Ionosphere** [30]. Radar returns from the ionosphere, classified into good (normal) or bad (anomalous)

**Boston** [31]. Regression data set on housing data and prices for areas in Boston, transformed as described above

Image data sets consist of either colored or black-and-white images. Their characteristics are shown in Table IV.

**MVTec AD\*** [27]. MVTec AD consists of a set of real-world data sets with object and texture images. From this collection, the data sets transistor, screw, pill, carpet and hazelnut (hazel) are used.

**MNIST\_0** (mnist) [32]. Hand-written digits classification data set, with instances for each digit from 0 to 9. Here, we transform it into an anomaly detection data set by setting the label 0 as normal label and all others as anomalous, which also defines the suffix of the data set name.<sup>5</sup>

**CIFAR-10\_0** (cifar10) [33]. Classification data set with 10 classes of images. Here, we set label 0 as normal label and all others as anomalous.

### B. Algorithms

In Section III-B, a variety of different categories of algorithms was laid out. At least one algorithm from each category is presented here and used in the experiments. Unless noted otherwise, PyOD's [34] implementation is used. For each setting, we provide a table indicating which algorithms are used in this setting and which hyperparameters they are used with to ensure reproducibility. If hyperparameters are not mentioned, the defaults of PyOD v0.9.3 are used.

<sup>3</sup>A list of all built-in data sets is provided in the supplementary github.

<sup>4</sup>For details please refer to the supplementary material on github.

<sup>5</sup>OAB also supports iterating through all labels as normal label to obtain a single score for all anomaly data sets composable from MNIST.

TABLE III

TABULAR DATA SETS.  $n_{\text{NORMAL}}$  IS THE NUMBER OF NORMAL DATA POINTS, AND  $n_{\text{ANOMALY}}$  THE NUMBER OF ANOMALIES IN THE DATA SET. FEATURES INDICATES THE NUMBER OF FEATURES (OR DIMENSIONALITY) OF EACH DATA POINT.

Name	$n_{\text{normal}}$	$n_{\text{anomaly}}$	Features
spambase	2528	1679 (39.91%)	57
wilt	4562	257 (5.33%)	5
NASA	877	315 (26.43%)	21
annthyroid	6528	534 (7.56%)	6
page-blocks	4883	510 (9.46%)	10
ionosphere	225	125 (35.71%)	33
boston	475	31 (6.13%)	13

TABLE IV

IMAGE DATA SETS. THE FINAL FIVE DATA SETS ARE FROM MVTEC AD.

Name	$n_{\text{normal}}$	$n_{\text{anomaly}}$	Features
MNIST_0	6903	63097 (90.15%)	(28, 28)
CIFAR10_0	6000	54000 (90%)	(32, 32, 3)
transistor	273	40 (12.78%)	(256, 256, 3)
screw	361	119 (24.79%)	(256, 256)
pill	293	141 (32.49%)	(256, 256, 3)
carpet	308	89 (22.41%)	(256, 256, 3)
hazelnut	431	70 (13.97%)	(256, 256, 3)

For tabular data, Table V covers the unsupervised setting and Table VI covers the semisupervised setting. For image data, please refer to Tables VII and VIII respectively. In the following the core concepts of each benchmarked anomaly detection algorithm gets briefly described.

**Nearest neighbor-based approaches: kNN.** The vanilla k-nearest neighbors (kNN) algorithm uses the distance to the k-th neighbour of a data point as anomaly score. The underlying assumption is that normal data points can be found in dense neighborhoods, which is not the case for anomalous points [17]. The choice for k is inspired by the choice of Domingues et al. [11] for similar algorithms.

**Nearest neighbor-based approaches: LOF.** As the name Local Outlier Factor [35] indicates, the neighbourhood of a point is investigated to judge if a data point is anomalous instead of observing all data points globally. If a data point is in a less dense neighborhood than its neighbors, it is assumed to have a higher likeliness of being an anomaly. The choice for k is taken from the choice of Domingues et al. [11], but a lower bound is chosen as some data sets have few observations.

**Nearest neighbor-based approaches: ABOD.** Angle-Based outlier Detection was proposed by Kriegel et al. [36] and assumes that the variation in angles from a data point to other data points is larger for normal data points, as they can be found towards the center of a dense cluster, and smaller for anomalies. As for kNN and LOF, the hyperparameter k is inspired by the choice of Domingues et al. [11].

**Isolation-based approaches: Isolation Forest.** Isolation Forests [18] consist of isolation trees which are trees built at random. A data point traverses the trees and it is assumed that anomalies are isolated, i.e., reach a leaf node, faster than normal data points.

**One-class approaches: OCSVM.** One-Class Support Vector Machines [37] map data points into a feature space and learn a hyperplane that separates the normal training data points from the origin as tightly as possible. Test data points are assigned anomaly scores based on their position relative to the hyperplane in the feature space.

**Reconstruction-based approaches: PCA.** Principal component analysis [38], [39] is a linear model that can be used to reduce dimensionality. From this reduced dimensional-representation, the original representation is recomputed using the eigenvectors. The respective mappings can either be learned on a clean training set (semisupervised) or on a test set in which normal data points dominate (unsupervised). The reconstruction error is transformed into an anomaly score.

**Deep learning for feature extraction: AE + Traditional algorithm.** Autoencoders (AE) [40] can be used as a means to reduce feature dimensionality. The bottleneck representation can be fed into a traditional algorithm. For image data, convolutional AEs (CAEs) are usually used. CAEs as well as the combination of two algorithms are self-implemented.

**Learning feature representations of normality: AE.** AEs can be used to reduce dimensionality, but analogous to PCA, they can also be used as anomaly detection algorithm in both the unsupervised and semisupervised setting. As in PCA, the anomaly scores are calculated based on the reconstruction error of a data point. Alternatives are Variational AEs (VAEs) [41] and CAEs.

TABLE V

HYPERPARAMETERS FOR ALGORITHMS USED IN UNSUPERVISED TABULAR DATA ANOMALY DETECTION.  $n$  IS THE NUMBER OF DATA POINTS IN A SAMPLE.

KNN	n_neighbors	$\max(n, 0.05, 10)$
LOF	n_neighbors	$\max(n, 0.1, 10)$
ABOD	n_neighbors	$\max(n, 0.01, 10)$
IForest	random_state	42
AE	random_state	42
	hidden_layers	[6, 3, 3, 6]
AE+LOF	AE: random_state	42
	AE: hidden_layers	[6, 3, 3, 6]
	LOF: n_neighbors	$\max(n, 0.1, 10)$

TABLE VI

HYPERPARAMETERS FOR ALGORITHMS USED IN SEMISUPERVISED TABULAR DATA ANOMALY DETECTION.

OCSVM	-	-
IForest	random_state	42
PCA	n_components	0.9
	pca_solver	'full'
AE	random_state	42
	hidden_layers	[6, 3, 3, 6]
VAE	random_state	42
	encoder_neurons	[6, 3]
	decoder_neurons	[3, 6]

TABLE VII

HYPERPARAMETERS FOR ALGORITHMS USED IN UNSUPERVISED IMAGE DATA ANOMALY DETECTION.  $n$  IS THE NUMBER OF DATA POINTS IN A SAMPLE. THE CAE PARAMETERS ARE ALWAYS THE SAME AS THOSE IN THE LAST ROW.

CAE+kNN	kNN: n_neighbors	$\max(n, 0.05, 10)$
CAE+LOF	LOF: n_neighbors	$\max(n, 0.1, 10)$
CAE+ABOD	ABOD: n_neighbors	$\max(n, 0.01, 10)$
CAE+IForest	IForest: random_state	42
CAE	random_state	42
	epochs	50
	latent_dim	100

TABLE VIII

HYPERPARAMETERS FOR ALGORITHMS USED IN SEMISUPERVISED IMAGE DATA ANOMALY DETECTION. THE CAE PARAMETERS ARE ALWAYS THE SAME AS THOSE IN THE LAST ROW.

CAE+OCSVM	-	-
CAE+IForest	IForest: random_state	42
CAE	random_state	42
	epochs	50
	latent_dim	100

### C. Results

We ran the algorithms mentioned above on the data sets presented in Section IV-A. The sampling parameters are those described in Tables I and II in Section III-D1 and the MVTEC AD data sets were sampled with their native train-test split in the semisupervised setting. Results for tabular data sets can be found in Tables IX and X for unsupervised and semisupervised anomaly detection. The results for image data sets can be found in Tables XII and XI respectively.

### D. Reproduce results from other benchmarks

To further validate OAB, we aimed at reproducing results from other papers. As the preprocessing is in large parts comparable to Campos et al. [5], we focussed on their results. Because they report results for each sampled data set individually and with varying hyperparameters, we were specifically interested in algorithms which had the same or very similar hyperparameters for each sampled data set. We found that this is the case for LOF with n\_neighbors=100 on the PageBlocks data set<sup>6</sup> with a contamination rate of 0.05 in its unscaled variant without duplicates. We further reproduced their sampling procedure by setting the number of sampled points to 5139. As shown in the corresponding Google-Colab accompanying this paper on github, we were able to reproduce their results. The ROC AUC score was the same up to 3 digits after the decimal, and other scores matched up to 2 digits after the decimal.

## V. CONCLUSION

In summary, we introduced in this paper OAB, an Open Anomaly Benchmark framework for unsupervised and semisupervised anomaly detection on image and tabular data sets.

<sup>6</sup><https://www.dbs.ifi.lmu.de/research/outlier-evaluation/DAMI/semantic/PageBlocks/>

TABLE IX  
TABULAR DATA IN UNSUPERVISED SETTING, ROC AUC WITH STANDARD DEVIATIONS.

	spambase	wilt	NASA	annthyroid	page-blocks	ionosphere	boston	Average
kNN	0.633 ± 0.025	0.612 ± 0.005	0.653 ± 0.034	0.947 ± 0.002	0.935 ± 0.004	0.950 ± 0.030	0.733 ± 0.008	0.780
LOF	0.588 ± 0.025	0.573 ± 0.006	0.652 ± 0.032	<b>0.950 ± 0.002</b>	0.927 ± 0.004	0.907 ± 0.040	0.715 ± 0.010	0.758
IForest	<b>0.778 ± 0.014</b>	0.414 ± 0.027	0.671 ± 0.031	0.826 ± 0.008	0.914 ± 0.004	0.947 ± 0.026	<b>0.811 ± 0.018</b>	0.766
ABOD	0.721 ± 0.016	<b>0.699 ± 0.005</b>	<b>0.698 ± 0.029</b>	0.927 ± 0.005	<b>0.954 ± 0.004</b>	<b>0.970 ± 0.022</b>	0.632 ± 0.042	<b>0.800</b>
AE	0.753 ± 0.019	0.334 ± 0.008	0.593 ± 0.040	0.688 ± 0.011	0.915 ± 0.005	0.917 ± 0.035	0.796 ± 0.012	0.714
AELOF	0.504 ± 0.005	0.500 ± 0.000	0.506 ± 0.008	0.503 ± 0.004	0.524 ± 0.032	0.503 ± 0.013	0.470 ± 0.004	0.501
Average	0.663	0.522	0.629	0.807	0.861	0.866	0.693	

TABLE X  
TABULAR DATA IN SEMISUPERVISED SETTING, ROC AUC WITH STANDARD DEVIATIONS.

	spambase	wilt	NASA	annthyroid	page-blocks	ionosphere	boston	Average
OCSVM	0.658 ± 0.009	<b>0.463 ± 0.008</b>	0.618 ± 0.014	<b>0.953 ± 0.002</b>	<b>0.944 ± 0.003</b>	0.902 ± 0.021	0.738 ± 0.023	0.754
IForest	<b>0.821 ± 0.011</b>	0.452 ± 0.026	<b>0.663 ± 0.013</b>	0.904 ± 0.011	0.929 ± 0.004	<b>0.923 ± 0.027</b>	<b>0.838 ± 0.014</b>	<b>0.790</b>
PCA	0.810 ± 0.007	0.327 ± 0.040	0.587 ± 0.015	0.792 ± 0.013	0.929 ± 0.004	0.903 ± 0.024	0.812 ± 0.017	0.737
AE	0.809 ± 0.007	0.351 ± 0.006	0.565 ± 0.019	0.822 ± 0.012	0.934 ± 0.004	0.914 ± 0.021	0.819 ± 0.012	0.745
VAE	0.810 ± 0.007	0.341 ± 0.006	0.585 ± 0.017	0.823 ± 0.012	0.933 ± 0.004	0.902 ± 0.022	0.814 ± 0.015	0.744
Average	0.782	0.387	0.604	0.859	0.934	0.909	0.804	

TABLE XI  
IMAGE DATA IN SEMISUPERVISED SETTING, ROC AUC WITH STANDARD DEVIATIONS. NOTE THAT FOR MVTEC AD DATA SETS, THE ORIGINAL TRAIN-TEST SPLIT IS USED. THEREFORE, DATA IS NOT SAMPLED MULTIPLE TIMES AND THERE IS NO STANDARD DEVIATION.

	mnist	cifar10	transistor	screw	pill	carpet	hazelnut	Avg
CAEOCSVM	<b>0.991 ± 0.001</b>	0.639 ± 0.009	<b>0.721</b>	<b>0.720</b>	0.502	<b>0.614</b>	0.632	<b>0.688</b>
CAEIForest	0.980 ± 0.003	0.641 ± 0.014	0.718	0.269	<b>0.524</b>	0.519	0.661	0.616
CAE	0.943 ± 0.005	<b>0.714 ± 0.008</b>	0.675	0.001	0.509	0.492	<b>0.841</b>	0.597
Avg	0.971	0.665	0.705	0.330	0.512	0.542	0.711	

TABLE XII  
IMAGE DATA IN UNSUPERVISED SETTING, ROC AUC SCORES.

	mnist	cifar10	transistor	screw	pill	carpet	hazelnut	Avg
CAEKNN	0.992	0.639	0.649	0.486	0.686	0.556	0.554	0.652
CAELOF	<b>0.995</b>	0.632	0.657	0.526	<b>0.790</b>	0.541	0.548	0.670
CAEABOD	0.935	0.657	<b>0.686</b>	<b>0.567</b>	0.676	<b>0.569</b>	<b>0.671</b>	<b>0.680</b>
CAEIForest	0.967	0.645	0.638	0.507	0.575	0.521	0.545	0.628
CAE	0.952	<b>0.737</b>	0.635	0.360	0.643	0.381	0.401	0.587
Avg	0.968	0.662	0.653	0.489	0.674	0.514	0.544	

With this work-in-progress we did neither aim to find the best performing anomaly detection algorithm nor replace well established benchmarking methods or try to cover each algorithm or data set ever benchmarked in this context. Instead, we demonstrated that OAB allows to standardize benchmarking related steps like preprocessing, sampling, train-test-splitting as well as the actual evaluation while simultaneously ensuring reproducibility, comparability and low-effort extensibility for new anomaly detection algorithms and data sets. Thereby, OAB sets the foundation towards an automatized fulfillment of the requirements of the *Machine Learning Reproducibility Checklist* [4].

Furthermore, the OAB framework provides a selection of semantically meaningful real-world anomaly data sets and covers different methods for preprocessing, sampling and contamination. Addressing the openness of the framework, OAB is open for further contributions by sharing the code

as open-source project<sup>7</sup> but it is also open in relation to offering existing benchmarks in detail (data sets, algorithms, benchmark setups, etc.). The practical use of OAB is supported by well documented Google-Colabs shared in the github repository for hands-on experience.

Since a reasonable benchmark should not only consider the performance of an algorithm on a set of arbitrarily selected data sets for specific evaluation metrics, OAB would benefit from future work evaluating algorithms related to their dependency on the choice of normalization method and on the data set characteristics like already demonstrated by Kandanaarachchi et al. [42]. Moreover, as there are no external labels available in the actual unsupervised setting, the sampling of OAB could be further improved by sampling from different embedding clusters to ensure a point difficulty balance in case of downsampling or train-test-splitting.

#### ACKNOWLEDGMENT

This work has been funded by the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A. The authors of this work take full responsibility for its content.

<sup>7</sup><https://github.com/ISDM-CAU-Kiel/oab>

## REFERENCES

- [1] D. M. Hawkins, "Identification of outliers," in *Monographs on Applied Probability and Statistics*, 1980.
- [2] G. Pang, C. Shen, L. Cao, and A. V. D. Hengel, "Deep learning for anomaly detection: A review," *ACM Comput. Surv.*, vol. 54, 2021.
- [3] L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Muller, "A unifying review of deep and shallow anomaly detection," *Proceedings of the IEEE*, vol. 109, p. 756–795, 2021.
- [4] J. Pineau, P. Vincent-Lamarre, K. Sinha, V. Lariviere, A. Beygelzimer, F. d'Alche Buc, E. Fox, and H. Larochelle, "Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program)," *Journal of Machine Learning Research*, vol. 22, pp. 1–20, 2021. [Online]. Available: <http://jmlr.org/papers/v22/20-303.html>
- [5] G. O. Campos, A. Zimek, J. Sander, R. J. Campello, B. Micenkova, E. Schubert, I. Assent, and M. E. Houle, "On the evaluation of unsupervised outlier detection: Measures, datasets, and an empirical study," *Data Min. Knowl. Discov.*, vol. 30, p. 891–927, Jul. 2016.
- [6] A. Emmott, S. Das, T. G. Dietterich, A. Fern, and W.-K. Wong, "A meta-analysis of the anomaly detection problem," *arXiv: Artificial Intelligence*, 2015.
- [7] F. Keller, E. Muller, and K. Bohm, "Hics: High contrast subspaces for density-based outlier ranking," in *2012 IEEE 28th International Conference on Data Engineering*, 2012, pp. 1037–1048.
- [8] A. Zimek, M. Gaudet, R. J. G. B. Campello, and J. Sander, "Subsampling for efficient and effective unsupervised outlier detection ensembles," *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2013.
- [9] J. Yang, N. Zhong, Y. Yao, and J. Wang, "Local peculiarity factor and its application in outlier detection," in *KDD*, 2008.
- [10] A. McEwen, "Views of msl hardware 12 days after landing," 2012. [Online]. Available: [https://www.uahirise.org/ESP\\_028401\\_1755](https://www.uahirise.org/ESP_028401_1755)
- [11] R. Domingues, M. Filippone, P. Michiardi, and J. Zouaoui, "A comparative evaluation of outlier detection algorithms: Experiments and analyses," *Pattern Recognit.*, vol. 74, pp. 406–421, Sep. 2018.
- [12] D. Dua and C. Graff, "UCI machine learning repository," 2017. [Online]. Available: <http://archive.ics.uci.edu/ml>
- [13] E. Schubert and A. Zimek, "ELKI: A large open-source library for data analysis - ELKI release 0.7.5 "heidelberg"," *CoRR*, vol. abs/1902.03616, 2019.
- [14] J. Vanschoren, J. N. van Rijn, B. Bischl, and L. Torgo, "Openml," *ACM SIGKDD Explorations Newsletter*, vol. 15, p. 49–60, 2014.
- [15] M. Goldstein and S. Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data," *PLoS ONE*, vol. 11, 2016.
- [16] S. Lu, L. Liu, J. Li, T. Le, and J. Liu, "Dependency-based anomaly detection: Framework, methods and benchmark," *ArXiv*, vol. abs/2011.06716, 2020.
- [17] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Comput. Surv.*, vol. 41, pp. 15:1–15:58, 07 2009.
- [18] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Trans. Knowl. Discov. Data*, vol. 6, Mar. 2012.
- [19] H. Marques, R. J. G. B. Campello, A. Zimek, and J. Sander, "On the internal evaluation of unsupervised outlier detection," *Proceedings of the 27th International Conference on Scientific and Statistical Database Management*, 2015.
- [20] C. C. Aggarwal, *Outlier Analysis*, 2nd ed. Springer Publishing Company, Incorporated, 2016.
- [21] J. Hanley and B. McNeil, "The meaning and use of the area under a receiver operating characteristic (roc) curve," *Radiology*, vol. 143 1, pp. 29–36, 1982.
- [22] N. Craswell, *Precision at n*. Boston, MA: Springer US, 2009, pp. 2127–2128.
- [23] E. Zhang and Y. Zhang, *Average Precision*. Boston, MA: Springer US, 2009, pp. 192–193.
- [24] M. Friedman, "A Comparison of Alternative Tests of Significance for the Problem of  $m$  Rankings," *The Annals of Mathematical Statistics*, vol. 11, pp. 86–92, 1940.
- [25] P. Nemenyi, *Distribution-free Multiple Comparisons*. Princeton University, 1963.
- [26] U. M. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "Knowledge discovery and data mining: Towards a unifying framework," in *KDD*, 1996.
- [27] P. Bergmann, M. Fauser, D. Sattlegger, and C. Steger, "Mvtec ad – a comprehensive real-world dataset for unsupervised anomaly detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.
- [28] P. Bergmann, K. Batzner, M. Fauser, D. Sattlegger, and C. Steger, "The mvtec anomaly detection dataset: A comprehensive real-world dataset for unsupervised anomaly detection," *Int. J. Comput. Vis.*, vol. 129, pp. 1038–1059, 2021.
- [29] J. Sayyad Shirabad and T. Menzies, "The promise repository of software engineering databases," <https://www.openml.org/d/1067>, 2005.
- [30] S. Rayana, "Odds library," 2016. [Online]. Available: <http://odds.cs.stonybrook.edu>
- [31] D. Harrison and D. L. Rubinfeld, "Hedonic housing prices and the demand for clean air," *Journal of Environmental Economics and Management*, vol. 5, pp. 81–102, 1978.
- [32] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, pp. 141–142, 2012.
- [33] A. Krizhevsky, "Learning multiple layers of features from tiny images," pp. 32–33, 2009. [Online]. Available: <https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf>
- [34] Y. Zhao, Z. Nasrullah, and Z. Li, "Pyod: A python toolbox for scalable outlier detection," *Journal of Machine Learning Research*, vol. 20, pp. 1–7, 2019.
- [35] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers," *SIGMOD Rec.*, vol. 29, p. 93–104, 2000.
- [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. New York, NY, USA: Association for Computing Machinery, 2008, p. 444–452.
- [37] B. Schölkopf, J. Platt, J. Shawe-Taylor, A. Smola, and R. Williamson, "Estimating support of a high-dimensional distribution," *Neural Computation*, vol. 13, pp. 1443–1471, 07 2001.
- [38] K. Pearson, "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, pp. 559–572, 1901.
- [39] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *Journal of Educational Psychology*, vol. 24, pp. 498–520, 1933.
- [40] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
- [41] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," 2014.
- [42] S. Kandanaarachchi, M. A. Muñoz, R. Hyndman, and K. Smith-Miles, "On normalization and algorithm selection for unsupervised outlier detection," *Data Mining and Knowledge Discovery*, vol. 34, pp. 309–354, 2019.



## Chapter 9

# Over-optimistic evaluation and reporting of novel cluster algorithms: an illustrative study

This chapter consists of the following publication:

[80] Theresa Ullmann, Anna Beer, Maximilian Hünemörder, Thomas Seidl, and Anne-Laure Boulesteix. “Over-optimistic evaluation and reporting of novel cluster algorithms: an illustrative study”. In: *Advances in Data Analysis and Classification* (2022), pp. 1–28. DOI: 10.1007/s11634-022-00496-5

Nothing was changed compared to the final version which was published under a Creative Commons Attribution 4.0 International License (see <https://creativecommons.org/licenses/by/4.0/>)

**Statement of Originality:** T. Ullmann conceptualized the paper together with A. Beer, **M. Hünemörder**, and A.-L. Boulesteix. T. Ullmann was the lead for designing the article’s methodology, with A. Beer and **M. Hünemörder** providing support. The analysis software was written by **M. Hünemörder**, with support from T. Ullmann and A. Beer. T. Ullmann performed validation of the software to check the reproducibility of results. The majority of the original draft was written by T. Ullmann, with the exception of the subsections on “Information visualization”, “Robustness”, and “Adversarial Attacks”, which were written by A. Beer. The other authors (**M. Hünemörder**, T. Seidl, and A.-L. Boulesteix) made several comments regarding the original draft that were included in the manuscript. The review and editing of the manuscript were led by T. Ullmann and supported by all other authors. All authors read and approved the final version of the article.

- **Conception:** Ullmann (Lead), Beer (Support), **Hünemörder** (Support), Boulesteix (Support)
- **Planning:** All Authors
- **Execution:** **Hünemörder** (Lead), Ullmann (Support), Beer (Support)
- **Manuscript:** Ullmann (Lead), Beer (Support), **Hünemörder** (Support), Boulesteix (Support), Seidl (Support)



## Over-optimistic evaluation and reporting of novel cluster algorithms: an illustrative study

Theresa Ullmann<sup>1</sup> · Anna Beer<sup>2</sup> · Maximilian Hünemörder<sup>3</sup> · Thomas Seidl<sup>2</sup> · Anne-Laure Boulesteix<sup>1</sup>

Received: 12 August 2021 / Revised: 18 December 2021 / Accepted: 14 February 2022  
© The Author(s) 2022

### Abstract

When researchers publish new cluster algorithms, they usually demonstrate the strengths of their novel approaches by comparing the algorithms' performance with existing competitors. However, such studies are likely to be optimistically biased towards the new algorithms, as the authors have a vested interest in presenting their method as favorably as possible in order to increase their chances of getting published. Therefore, the superior performance of newly introduced cluster algorithms is over-optimistic and might not be confirmed in independent benchmark studies performed by neutral and unbiased authors. This problem is known among many researchers, but so far, the different mechanisms leading to over-optimism in cluster algorithm evaluation have never been systematically studied and discussed. Researchers are thus often not aware of the full extent of the problem. We present an illustrative study to illuminate the mechanisms by which authors—consciously or unconsciously—paint their cluster algorithm's performance in an over-optimistic light. Using the recently published cluster algorithm Rock as an example, we demonstrate how optimization of the used datasets or data characteristics, of the algorithm's parameters and of the choice of the competing cluster algorithms leads to Rock's performance appearing better than it actually is. Our study is thus a cautionary tale that illustrates how easy it can be for researchers to claim apparent "superiority" of a new cluster algorithm. This illuminates the vital importance of strategies for avoiding the problems of over-optimism (such as, e.g., neutral benchmark studies), which we also discuss in the article.

---

Theresa Ullmann  
tullmann@ibe.med.uni-muenchen.de

<sup>1</sup> Institute for Medical Information Processing, Biometry, and Epidemiology, LMU Munich, Marchioninstr. 15, 81377 München, Germany

<sup>2</sup> Institute for Informatics, LMU Munich, München, Germany

<sup>3</sup> Department of Computer Science, CAU Kiel, Kiel, Germany

**Keywords** Over-optimism · Clustering · Cluster analysis · Evaluation

**Mathematics Subject Classification** 62H30 · 68W40

## 1 Introduction

Cluster analysis refers to grouping similar objects in data, while separating dissimilar ones. While there already are a huge number of cluster algorithms (see e.g., Xu and Wunsch (2010) for an overview), researchers continue to propose novel algorithms every year. Researchers who introduce a new cluster algorithm typically publish it together with a demonstration of the strengths of their approach and its superiority over alternative methods.

However, the results of such studies should be regarded with caution. *Publication bias* (Boulesteix et al. 2015) constitutes a considerable external incentive for researchers to demonstrate the superiority of their new approach: journals and conferences are much more likely to accept a paper about a novel computational method if this method shows good performance and is “better” than pre-existing approaches. This may tempt researchers to present their method’s performance in an *over-optimistic* fashion, a mechanism that is also called the “self-assessment trap” (Norel et al. 2011). Such scenarios can not only appear in the research field of clustering but can also be found in all types of *methodological research*, i.e., the development and evaluation of data analytic techniques and algorithms (Boulesteix et al. 2020).

Over-optimization is not necessarily performed in a malicious or even intentional manner, but it is problematic because the new method may turn out to have a worse performance than initially claimed when it is later investigated in a neutral comparison study, i.e., a study whose authors do not have a vested interest in one of the competing methods, see Boulesteix et al. (2013). In other words, the good performance result is not replicable (Boulesteix et al. 2020). Anecdotal evidence for this lack of replicability is presented by Buchka et al. (2021) for a specific data analysis problem related to the pre-processing of a special type of high-throughput molecular data. The over-optimistic presentation of computational methods may lead to the usage of flawed methods in applications, which could ultimately hinder research progress or even lead to questionable results in applied research.

But how exactly may researchers present their new methods in an over-optimistic fashion? For *supervised* classification, an illustrative case has already been presented in the field of bioinformatics by Jelizarow et al. (2010). They considered a “promising” novel classification method, which in reality was not superior to other classifiers. Yet the authors were able to demonstrate that different mechanisms allow over-optimistic presentation of this new method’s performance, namely choosing specific datasets, optimizing the method’s settings and characteristics to these datasets while burying the other in the file drawer, and choosing suboptimal competing classifiers.

However, to the best of our knowledge, such a study has not yet been conducted for cluster analysis, i.e., the unsupervised scenario. While over-optimistic (selective) reporting is well understood in the context of statistical testing and supervised learning, where its impact can be easily measured, it is much less so in the field of cluster analysis,

which is characterized by the difficulty to properly evaluate methods. We thus aim at filling this gap by demonstrating how a novel cluster algorithm's performance can be presented in an (overly) favorable light.

The problem of over-optimism is in fact as important in unsupervised clustering as it is in supervised classification, and is probably even exacerbated because the performance evaluation of cluster algorithms has not been studied as systematically as the evaluation of supervised classifiers in the methodological literature. Guidance for proper benchmarking of cluster algorithms has only recently emerged (Van Mechelen et al. 2018). Even though the “true” cluster labels are unknown in clustering applications, researchers typically use datasets with known labels to evaluate their novel cluster algorithms. To some extent, the performance evaluation of cluster algorithms thus appears similar to the evaluation of classifiers. Yet for cluster analysis, the role of test data is not as clear-cut as in supervised classification (Ullmann et al. 2021), which entails that researchers are less aware that “overfitting” can not only happen in supervised classification, but also in cluster analysis. Moreover, optimizing hyperparameters such as the number of clusters based on the “ground truth”, as is frequently done in cluster algorithm evaluation, does not take into account that other researchers who eventually want to use the algorithm in applications do not know the “true” cluster labels of their datasets, and will thus likely obtain worse results than the performances reported in the original evaluation of the novel algorithm. To evaluate their new method, researchers might also use performance evaluation measures which do not require a fixed “ground truth”, such as internal validation indices which measure internal properties of the data (e.g., homogeneity and/or separateness of the clusters). However, over-optimism can still be an issue when using these indices.

In the present study, we use the “Rock” algorithm (Beer et al. 2019) as an illustrative example. Beer et al. (2019) agreed to the usage of their algorithm in our paper. Rock was originally introduced as a “promising” new algorithm and was presented as being able to outperform competitors. In subsequent studies, it turned out that Rock does not generally perform better than its competitors. In the present paper, we show that Rock outperforms competing algorithms in very specific scenarios and that these scenarios can be obtained by three different mechanisms: (1.) optimization of datasets and data characteristics, (2.) optimization of parameters of the Rock algorithm and (3.) the choice of the competing clustering approaches. We demonstrate that if the optimized scenarios are selectively reported and the settings in which Rock performs worse are omitted, the algorithm then appears to outperform its competitors—as a result of an over-optimistic presentation.

Rock is used only as an example—demonstrating the specific characteristics of the Rock algorithm is not the main interest of our work. Rather, we use Rock to illustrate more general mechanisms of over-optimization. We suspect that many studies which introduce new cluster algorithms are affected by these mechanisms. However, given that over-optimization can happen quite subtly and/or unintentionally, we do not cite any published papers here which probably presented their results in an over-optimistic fashion. Neither do we try to quantify the actual optimistic bias that currently exists in the literature on cluster algorithms. Rather, our study is intended as a cautionary tale to raise awareness of the over-optimism problem, and to illuminate the importance of using strategies to avoid over-optimism (e.g., avoiding selective reporting, using

independent test data and conducting neutral benchmark studies, as discussed in detail in Sect. 6).

We first give an overview of related work in Sect. 2. Section 3 explains how we performed optimization of Rock's performance. The corresponding results are presented in Sect. 4 and further discussed in Sect. 5. Possible solutions for the problem of over-optimism are outlined in Sect. 6. We conclude the paper in Sect. 7.

## 2 Related work

In this section we discuss studies that are related to our work. After presenting studies which directly look at the over-optimistic bias of new computational methods, we address aspects in the field of data mining that are connected to over-optimistic presentation of cluster algorithms.

### 2.1 Previous work about over-optimistic bias of new computational methods

There appears to be a lack of literature about over-optimism in the introduction of new cluster algorithms. For computational methods other than clustering, there exist some studies, to our knowledge mostly in the field of bioinformatics.

As mentioned above, a study similar to ours was previously reported by Jelizarow et al. (2010), but for supervised classification. Moreover, while this study illustrated over-optimism with a classification method for gene expression data and used real cancer gene expression datasets for this purpose, our example is not application specific. For performance evaluation we choose simulated and real datasets which are frequently used for the evaluation of cluster algorithms in computational research (e.g., the synthetic "Two Moons" dataset, the Iris dataset etc., see Sect. 3).

Broadly speaking, the three categories of optimization mechanisms that we analyze are similar to the categories previously considered in Jelizarow et al. (2010), i.e., optimization of the data, optimization of the algorithm's characteristics, and the choice of competing approaches. However, the use of simulated data allows us to systematically consider data characteristics such as noise or dimensionality, which was not done for the real datasets used in Jelizarow et al. (2010).

In a similar application context, Yousefi et al. (2010) also addressed over-optimism when reporting the performance of newly proposed classifiers. They focused on classification on high-dimensional data with low sample size, such as gene expression data. The authors specifically considered the optimization of the datasets, i.e., they analyzed the optimistic bias that results from reporting only the datasets with the best (or second best) performance of the new classifier. They estimated this bias in a simulation study, by repeatedly sampling sets of datasets, and recording the best (or second best) performing dataset of each set. The aim of their study thus was to *quantify* the optimistic bias with specific focus on the choice of datasets, whereas we model different over-optimization mechanisms of a (hypothetical) researcher in an illustrative way. The results of Yousefi et al. (2010) show that in the high-dimensional data setting, there is indeed a large optimistic bias when reporting only the best or second best performing dataset.

Finally, again in the context of bioinformatics, a recent study aimed to estimate the optimistic bias in the reported performance of new computational methods to preprocess a special type of raw high-throughput molecular data (Buchka et al. 2021). The approach was to perform a literature search and compare the reported performance of newly introduced methods against their performance in later neutral comparison studies. As expected, novel methods were ranked better than competitors in most of the papers introducing them, but outperformed competitors at a lesser rate in neutral studies. Yet the new methods still outperformed more than 50% of their paired competitors in neutral studies, showing that while there is optimistic bias, there is also some level of genuine scientific progress.

Outside of bioinformatics, Ferrari Dacrema et al. (2021) assessed optimistic bias in research about recommender systems. Recommender algorithms can be used, for example, to propose new movies to a media streaming user based on previously watched movies. Many new recommendation algorithms based on deep learning were published in recent years, which usually claimed superiority over previous approaches. Ferrari Dacrema et al. (2021) repeated the evaluations of the original authors, but with additional baseline algorithms. Their analysis showed that most of the new methods did not actually outperform simple and long-known baseline algorithms, provided strong-performing baselines were chosen and their hyperparameters were tuned as carefully as those of the new algorithms. This highlights that not including strong competitors or not treating the competing methods fairly might lead to optimistic bias.

## 2.2 Information visualization

Over-optimistic presentation of results can also be obtained by visualization methods, i.e., not only by a biased selection of *which* data to show, but also by *how* the selected data is shown. Studies on *information visualization* address the latter aspect. For example, visualization methods with a high lie factor (the ratio between “size of effect shown in graphic” and “size of effect in data”, see Tufte (1983)), or misleading labeling and scaling of axes, could be used by a researcher to let their algorithm appear in a more favorable light.

We do not focus on such mechanisms in our study, and instead illustrate that over-optimistic reporting of results is also possible if all rules regarding “correct” information visualization are observed.

## 2.3 Robustness

Robust clustering algorithms yield a similar quality of results for similar input. Thus, it is unlikely that there are experimental setups which yield notably better results than similar experiments and could thus be selectively presented in an over-optimistic fashion. We do not systematically evaluate the robustness of any of the tested cluster algorithms in Sect. 4, but rather show how the lack of robustness can be exploited in order to over-optimistically present the results of the exemplary algorithm. Out of the diverse types of robustness, we focus on the lack of robustness regarding different properties of the data as well as hyperparameter settings. For example, we consider

robustness w.r.t. noise. “Noise” can mean either background noise, i.e., uniformly distributed points across the data space which do not belong to the original distribution, or jitter, i.e., small deviations or perturbations in the original distribution. We regard only the latter in our experiments.

That robustness is crucial for clustering algorithms was already stated by Davé and Krishnapuram (1997). In recent literature on cluster algorithms, the robustness regarding different properties of the data is often presented, e.g., the size of the dataset, number of clusters, dimensionality, and structure of the data. Usually there is a base case for which one property at a time is changed to regard the effects on the clustering result. However, it is often left unclear how and why this base case was obtained, and how the settings which are not regarded in the respective experiment are chosen.

Even though the robustness regarding the choice of hyperparameters seems similarly important, authors often refer to “expert knowledge” for finding the “best” setting, and omit a robustness analysis. This can lead to enormous disagreements in the evaluation of an algorithm, see, e.g., the controversy about DBSCAN (Ester et al. 1996; Gan and Tao 2015; Schubert et al. 2017). Even easily interpretable hyperparameters, such as the number of clusters  $k$  (e.g., for  $k$ -Means, Lloyd 1982), which at first sight do not seem to require a robustness analysis, might show better performance w.r.t. the evaluation measure when set at a value different from the “ground truth”.

To summarize, robustness regarding different aspects is not only important to guarantee a predictable quality of clustering for users, but also reduces the potential for over-optimism.

## 2.4 Adversarial attacks

An adversarial attacker may corrupt the results of an algorithm by only performing small changes or additions in a dataset, leading to a wrong but more favorable outcome for the attacker (Goodfellow et al. 2018). Even though adversarial attacks are most often regarded in context of supervised machine learning, they can also influence results of unsupervised machine learning: recently, Chhabra et al. (2020) showed that adversarial attacks are also possible for clustering, even without knowing important details of the cluster algorithm. Algorithms which tend to return results of highly varying quality, also for only small perturbations in the data, are easy victims not only for adversarial attacks, but also for over-optimism. However, where adversarial attackers aim at changing only certain results, over-optimistic researchers would try to change the impression of an algorithm’s overall quality. By knowing the details of their novel algorithm as well as deciding on all hyperparameters and competitive methods, the influence over-optimistic researchers can have on the presentation of their results is massive, especially compared to an adversarial attacker.

## 3 Over-optimization methods

In this section we outline the concept and the experimental design of our study. We first explain the three different categories of over-optimization mechanisms that we

illustrate in our study. We then detail our concrete implementation, e.g., the clustering algorithms, datasets, evaluation measure and optimization method.

### 3.1 Three categories of over-optimization

Imagine a researcher who wishes to present his/her cluster algorithm in a favorable light. We model the work process of this researcher as an “optimization task”: the characteristics of the study in which the new algorithm is compared to existing ones are optimized such that the researcher’s algorithm scores well, in particular better than the best performing competing algorithm. This optimization can refer to (1.) finding datasets or data characteristics for which the new algorithm works particularly well, (2.) finding optimal parameters of the algorithm (and vice versa, neglecting the search for optimal parameters for the competitors) or (3.) choosing specific competing algorithms.

*Optimizing datasets or data characteristics.* A new cluster algorithm might perform well for specific types of datasets, but not for other types. Researchers might decide to report only the best-performing types of datasets. Additionally, for synthetic datasets, there is potential for over-optimism when varying specific characteristics (e.g., the amount of noise, the sample size, or the number of dimensions), and reporting only the optimal settings. Moreover, simulated datasets depend on the random seed, such that in turn, the performance of the cluster algorithm might also vary over different random seeds. Researchers might actively look for a “good” random seed or simply stumble across a particular “good” random seed by chance, neglecting to try other random seeds to check for robustness.

*Optimizing the algorithm’s parameters or characteristics.* Hyperparameters of the cluster algorithm, or characteristics of the algorithm designed during the development phase, could be varied by researchers to look for the best result. Hyperparameter optimization (HPO) is per se a legitimate procedure in performance evaluation. However, there is less awareness for proper evaluation of cluster algorithms combined with HPO, compared to the more extensive methodological literature on correct evaluation of supervised classifiers with HPO (Boulesteix et al. 2008; Bischl et al. 2021). In cluster analysis, over-optimism in relation to HPO may result from (1.) optimizing hyperparameters based on the “true” cluster labels known to the researchers, and (2.) not splitting the data into training and test sets. Both aspects will be discussed in more detail in Sects. 4 and 5. Moreover, over-optimism might also result when researchers neglect to set optimal parameters for the *competing* algorithms, e.g., when choosing suboptimal hyperparameter defaults for the competitors while finetuning their own algorithm.

*Optimizing the choice of competing algorithms.* Finally, researchers might pick specific competing clustering methods that let their own algorithm appear in a better light. They could neglect to look for the best state-of-the-art competitor, instead opting for less optimal comparison algorithms. Even if the researchers are aware of state-of-the-art competitors, they might not include them because the codes are not openly available, or implemented in a programming language which they are not familiar with. Researchers could also think of different groups of competing cluster algorithms, and

then pick the group that is most favorable for comparison with their own algorithm. A new density-based cluster algorithm could for example be compared either with a group of other density-based algorithms, or with a group of some well-known, not necessarily density-based cluster algorithms. While both choices could in principle be sensible, it is over-optimistic if researchers either deliberately exclude a class of competitors a priori because they expect their novel algorithm to perform worse than this class, or if they choose the competitor group a posteriori after having seen the results (Jelizarow et al. 2010).

Apart from these three categories of optimization, there are some further optimization possibilities (e.g., optimizing the evaluation measure) that we do not analyze here in detail, but briefly discuss in Sect. 5.

We assume that usually, researchers do not consciously perform the three classes of optimization tasks in a malicious and systematic manner. Nevertheless, in the course of a longer research process during which researchers try different datasets, algorithm parameters/configurations and competing algorithms, researchers might optimize the settings in an unsystematic and (probably) unintentional manner. Even if researchers start their analysis with the best intentions, they might post-hoc rationalize their (over-optimistic) choices as perfectly reasonable decisions, given that “[h]umans are remarkably good at self-deception” and scientists often “fool themselves” (Nuzzo 2015).

One might argue that the optimizations outlined above are not actually *over-optimizations* and that it is perfectly fine to look for scenarios in which a novel algorithm performs well. We would agree that it is not a priori wrong to search for and report such scenarios, as a new cluster algorithm can never be expected to outperform every other cluster algorithm in every situation. However, it should also be transparently reported how the presented “successful” scenarios were obtained, and how the algorithm performs in other settings. Over-optimism ultimately appears when performance results are *selectively* reported. We will illustrate this with our results in Section 4.

### 3.2 Experimental setup

We now present the exemplary cluster algorithm and its settings, the competing algorithms, the datasets and the evaluation measure. Our fully reproducible code is available at <https://github.com/thullmann/overoptimism-clust-algo>.

In accordance with the authors, we used the already published algorithm Rock (Beer et al. 2019) as a novel and promising algorithm. Rock is an iterative approach similar to Mean Shift (Fukunaga and Hostetler 1975), but based on the  $k$  nearest neighbors (kNN) instead of the bandwidth. In each step, points “roam” to the mean of their respective  $k$  nearest neighbors. Points with a similar final position are assigned to a common cluster. The algorithm involves the hyperparameter  $t_{max}$ , which gives the maximum number of iterations. As the maximum meaningful value for  $k$  is fixed ( $k > \frac{n}{2}$  would lead to an assignment of all points to the same cluster), and the increase of  $k$  in every step is linear,  $t_{max}$  also determines the number  $k$  of nearest neighbors regarded in each iteration. The larger  $t_{max}$  is chosen, the closer values for  $k$  are in

consecutive steps. Lower values for  $t_{max}$  thus lead to larger gaps between consecutive values for  $k$ , which may cause volatile merges of different clusters. On the other hand, higher values for  $t_{max}$  lead to more iterations, which increases runtime.

As typical for short papers, only a limited number of experiments is presented in Beer et al. (2019), illustrating that the underlying idea is promising. The results for Rock looked good compared to  $k$ -Means (Lloyd 1982), DBSCAN (Ester et al. 1996) and Mean Shift, which are typical competitors in the field and representatives for algorithms finding different types of clusters. As examples for competing algorithms, we thus chose  $k$ -means, DBSCAN, Mean Shift and additionally Spectral Clustering (Ng et al. 2001).

As the clustering performance measure we use the Adjusted Mutual Information Score (AMI, Vinh et al. 2010), a version of the Mutual Information (MI) Score adjusted for chance agreement of random partitions. For each dataset and cluster algorithm, the known “true” clustering (as given either by the simulation design for the synthetic datasets or by additional label information for the real datasets) was compared via the AMI with the clustering found by the algorithm. The higher the AMI, the more similar the two clusterings are. The AMI attains its maximum value of 1 if the two clusterings are identical, and equals 0 if the MI between the two clusterings is equal to the MI value expected for two random partitions. We give the detailed mathematical definition of the AMI in the appendix A.

While we only use the AMI in our illustration for the sake of conciseness, a similar analysis could be performed for alternative indices which measure the agreement of the calculated clusterings with the “ground truth”, or even for internal validation indices which evaluate clusterings based on internal properties of the data alone and do not require the “ground truth” (see also the discussion in Sect. 5.2).

The choice of exemplary datasets is linked to the three different optimization tasks outlined in Sect. 3.1. We thus give the datasets for each task in turn and explain how the optimization was performed. Note that we performed the three optimization tasks sequentially, building on the results of each previous task. Of course, in reality, a researcher will likely not perform the optimizations in such a perfectly sequential matter, and might jump between different tasks of optimization or try to optimize different aspects simultaneously. Again, our sequential procedure merely serves illustrative purposes.

For some specific details of the implementation, we refer to the appendix A.

*Optimizing datasets and data characteristics.* For this part of the analysis, we chose three commonly used different synthetic datasets from scikit-learn (Pedregosa et al. 2011), see Fig. 2: Two Moons<sup>1</sup>, Blobs<sup>2</sup> (for details on this dataset, see the appendix A), and Rings<sup>3</sup>.

<sup>1</sup> [https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make\\_moons.html](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_moons.html), visited: 05/31/2021.

<sup>2</sup> [https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make\\_blobs.html](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs.html), visited: 05/31/2021.

<sup>3</sup> [https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make\\_circles.html](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_circles.html), visited: 05/31/2021.

First, we performed optimization by varying the following data characteristics: a) for Two Moons, the sample size and the jitter values (where “jitter” denotes small random perturbations to the original data points in the clusters), b) for Blobs, the sample size, the number of dimensions and the number of generated clusters (“blobs”), and c) for Rings, the sample size and the jitter values. The goal of the optimization was to find the parameter configuration (e.g., for Two Moons, the configuration  $(n, j)$  of sample size and jitter value) that yields the largest performance difference between Rock and the best of the competitors – which is not necessarily the parameter configuration that yields the best *absolute* performance of Rock.

That is, for each of the three types of synthetic datasets in turn, we performed the following formal optimization task:

$$\operatorname{argmax}_{D \in \mathcal{D}} \left\{ \frac{1}{10} \sum_{i=1}^{10} \left( \operatorname{AMI} \left( \operatorname{Rock}(D^i), y_{D^i} \right) - \max_{C \in \mathcal{C}} \operatorname{AMI} \left( C(D^i), y_{D^i} \right) \right) \right\} \quad (1)$$

where  $D \in \mathcal{D}$  denotes the different variants of the dataset. For example, for the Two Moons data, each dataset  $D$  is a version of Two Moons with a specific jitter value and sample size. Each  $D$  has a cluster label ground truth  $y_D$ . For each  $D \in \mathcal{D}$ , ten different versions of  $D$ , namely  $D^i, i = 1, \dots, 10$  resulting from ten different random seeds were generated. Put differently, we performed ten simulation iterations per setting, i.e., we sampled ten datasets from each data distribution with a specific data parameter setting. The AMI difference is then averaged over these ten versions. This is supposed to reduce the influence of the random seed. Only at a later point in the analysis did we look at the effect of picking specific random seeds (see below).  $\operatorname{Rock}(D^i)$  denotes the application of Rock to the data  $D^i$ , returning a partition of the objects. Analogously, the competing algorithms  $C \in \mathcal{C}$  return a partition of  $D^i$ , with  $\mathcal{C} = \{k\text{-means, DBSCAN, Mean Shift, Spectral Clustering}\}$ .

For each of the three types of datasets in turn, we performed the optimization task (1) by using the Tree-structured Parzen Estimator (TPE, Bergstra et al. 2011), as implemented in the Optuna framework (Akiba et al. 2019) in Python<sup>4</sup>. TPE is a Bayesian optimization (BO) method. BO approaches sequentially propose new parameter configurations based on a library of previous evaluations of the objective function (for more details on BO methods and the TPE, see the appendix A). The TPE is often used for hyperparameter optimization of machine learning models, but in our case, we use it to optimize the *data* parameters. The TPE optimization can be considered as a very simplified model of the researcher’s optimization procedure. Of course, a researcher’s behavior does not exactly correspond to the mathematical procedure of the TPE. However, if researchers perform *intentional* (over-)optimization, then they might indeed use an optimization method such as the TPE to find the best data settings. The Bayesian optimization mimics the researcher’s (unintentional) over-optimization in the following sense: as mentioned above, a researcher developing a new cluster algorithm might sequentially look for data settings in which the new algorithm performs well, taking

<sup>4</sup> <https://optuna.readthedocs.io/en/stable/reference/generated/optuna.samplers.TPESampler.html>, visited: 05/31/2021.

into account performance information from previously tried data parameters. This is the reason why we chose the TPE over a simple grid search or random search, because the latter do not use previously obtained performance information. To make the TPE process more “realistic”, we supplied a grid of limited discrete values to the TPE, given that a researcher presumably would not try arbitrary real numbers. We performed this experiment with only 100 optimization steps for each of the three types of datasets, in order to fairly represent a researcher trying different data parameters by hand.

After determining the optimal values for the data parameters (which we will later report in Table 1 in Sect. 4.1), we analyzed the performance of Rock for non-optimal parameter values. That is, for each dataset and single data parameter in turn, the parameter was varied over a list of values, while the other data parameters were kept fixed at their optimal values. For example, for the Two Moons dataset we tried different jitter values and plotted the corresponding performance as measured by the mean AMI over ten random seeds against the jitter, keeping the sample size at the optimal value determined by the TPE. These analyses show the effects of selectively reporting only the best data parameters versus the performance of the algorithm over a broader range of each data parameter.

In the experiments given so far, we always considered the AMI averaged over ten random seeds. In the final step of the analysis for this section, we specifically study the influence of individual random seeds. We take the Two Moons dataset as an example, with a data parameter setting which is not optimal for Rock, but for which DBSCAN performs very well. We generate 100 datasets with these characteristics by setting 100 different random seeds, to check whether there exist particular seeds for which Rock does perform well, leading to over-optimization potential.

For all experiments described so far, we applied reasonable parameter choices (defaults or heuristics) for the cluster algorithms. For Rock we chose  $t_{max} = 15$ , as done for all experiments in the original paper (Beer et al. 2019), and for the competing algorithms see the appendix A.

*Optimizing the algorithm’s parameters or characteristics.* For this example we varied Rock’s hyperparameter  $t_{max}$  (maximum number of iterations). As  $t_{max}$  is discrete with a reasonable range of  $\{1, \dots, 30\}$ , a researcher could easily try every value by hand. Thus we did not perform optimization with the TPE, but with a full grid search, i.e., we calculated the AMI performance of Rock for each value of  $t_{max}$  and for each dataset. For this illustration, we considered the *absolute* performance of Rock, given researchers would also strive to maximize the absolute performance of their novel algorithm.

As exemplary datasets, we again considered Two Moons, Blobs and Rings, and additionally four real datasets frequently used for performance evaluation: Digits, Wine, Iris and Breast Cancer as provided by scikit-learn<sup>5</sup> (see also the UCI Machine Learning Repository, Dua and Graff 2017). The data parameter settings for the three synthetic datasets (sample size, amount of jitter etc.) corresponded to the optimal settings from the TPE optimization of (1). We used a single random seed to generate the illustrative synthetic datasets.

<sup>5</sup> [https://scikit-learn.org/stable/datasets/toy\\_dataset.html](https://scikit-learn.org/stable/datasets/toy_dataset.html), visited: 05/31/2021.

In a next step, using the Two Moons dataset as an example, we compared the AMI performances of Rock and DBSCAN over ten random seeds, first without, then with hyperparameter optimization for Rock and DBSCAN. We used the TPE for HPO of DBSCAN. Here, the TPE was not intended to model a researcher’s behavior, but was used as a classical HPO method. The comparison illustrates the effect of neglecting parameter optimization for competing algorithms.

*Optimizing the choice of competing algorithms.* We did not perform new experiments here. Rather, we looked at the results from the two previous optimization tasks to derive the potential for optimization of the choice of competing cluster algorithms.

## 4 Results

We present our results for the three optimization tasks outlined above, starting with the optimization of datasets and data characteristics.

### 4.1 Optimizing datasets and data characteristics

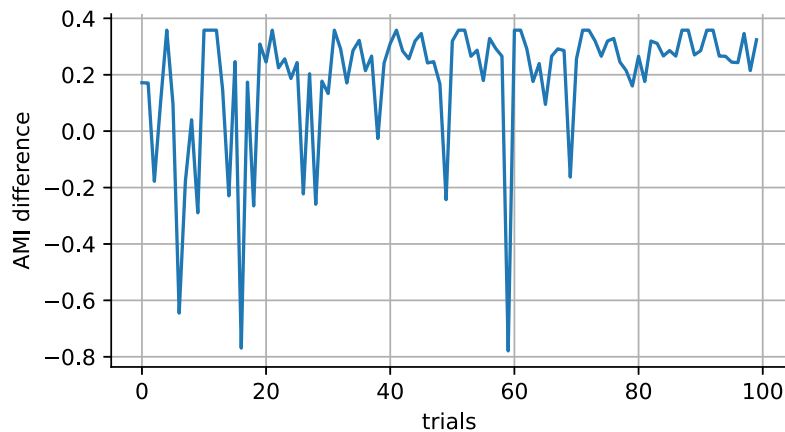
In this subsection we examine how strongly the choice of the “best” properties of a dataset, along with the type of dataset, can influence the performance estimation of Rock.

#### 4.1.1 Optimization of the data parameters with TPE

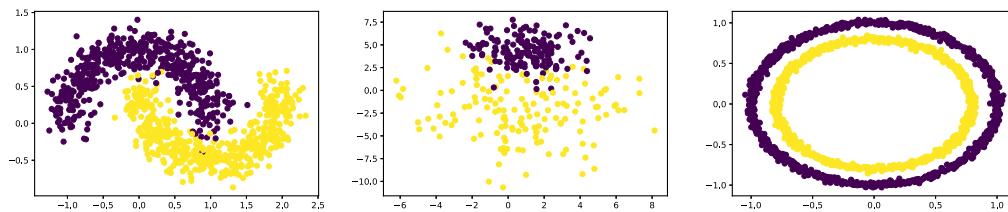
Table 1 reports the optimal data parameters for the three synthetic datasets as determined by the TPE optimization. The search space for each parameter is given in parentheses and consists of discrete values. The column “AMI diff.” shows the difference of the AMI obtained by Rock to the AMI obtained by the best competitor (averaged over ten random seeds). Recall that the AMI difference was used as the optimization criterion by the TPE to find the “optimal” parameter configuration. The column “Abs. AMI” denotes the absolute performance of Rock as measured by the AMI averaged over ten random seeds. The standard deviation over the seeds is also displayed.

**Table 1** Optimal data parameters as determined by the TPE optimization

Dataset	Sample size	Jitter	# of dim.	# of clusters	AMI diff.	Abs. AMI
Two Moons	<b>1000</b>	<b>0.15</b>	2	2	+0.3581	0.7881
	([1, 16] · 100)	([1, 20] · 0.01)	(default)	(default)		±0.1583
Blobs	<b>300</b>	–	<b>3</b>	<b>2</b>	+0.0475	0.8881
	([1, 16] · 100)		([2,20])	([2,10])		±0.1573
Rings	<b>1600</b>	<b>0.02</b>	2	2	+0.1789	0.1789
	([1, 16] · 100)	([1, 20] · 0.01)	(default)	(default)		±0.0026



**Fig. 1** Optimization progression for the Two Moons dataset, with the AMI difference averaged over ten random seeds



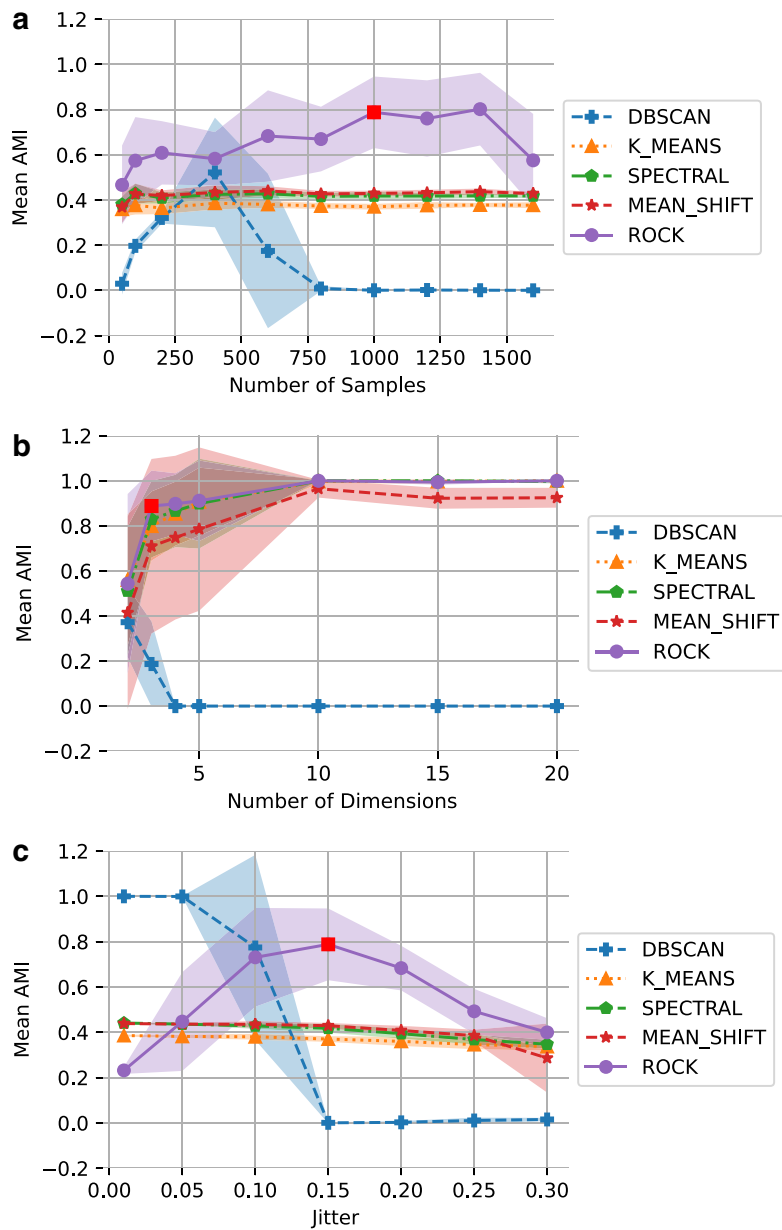
**Fig. 2** Example datasets (Two Moons, Blobs, Rings) with the optimal data parameters. For the Blobs example we only show the first and second dimensions

For the example of the Two Moons dataset, Fig. 1 shows a graphical representation of the TPE process over 100 optimization steps. The final “optimal” result is given by the best trial out of the 100 trials. The datasets with the optimal settings are pictured in Fig. 2, using a single illustrative seed of 0.

Judging from the results in Table 1, Rock appears to show better performance than its competitors. A researcher could use the results to claim Rock’s “superiority”. However, the *absolute* performance of Rock for the Rings dataset is not very good with a mean AMI of only 0.1789. Rock is only the best algorithm here because the competing methods completely fail to detect the clustering. A researcher who tries to optimize the data types might thus decide to let the Rings dataset disappear in the “file drawer”, particularly if he/she must omit some results due to page limits, and only present the Two Moons and Blobs datasets, for which Rock performs well, both in absolute and in relative (compared to competitors) terms. But would this presentation for Two Moons and Blobs be over-optimistic? To obtain a more realistic picture of Rock’s abilities, we analyze the results when the data parameters are not set at the optimal values, but varied over a grid.

#### 4.1.2 Varying the data parameters

We consider the influence of the sample size, the number of dimensions and the amount of jitter. For each data parameter, we pick one data type for illustrative purposes (either Two Moons or Blobs). The data parameters that are not currently considered are set



**Fig. 3** **a** Varying the sample size for the Two Moons dataset (jitter = 0.15), **b** varying the number of dimensions for the Blobs dataset (sample size = 300, number of blobs = 2), **c** varying the jitter amount for the Two Moons dataset (sample size = 1000)

to their optimal values from Table 1. Figure 3a–c show the performance of Rock and its competitors measured by the AMI over ten random seeds, depending on the varied data parameters. The border around each line shows the standard deviation over the seeds. Red squares indicate the optimal setting from Table 1.

*Sample size.* Here we consider the Two Moons dataset in Fig. 3a. We tried the following sample sizes: 50, 100, 200, 400, 600, 800, 1000, 1200, 1400, 1600. The jitter value is set at its optimal value 0.15 from Table 1. Rock indeed appears to perform better

here than its competitors over a broader range of numbers of samples, not just for the optimal setting. However, at smaller sample sizes, the difference to  $k$ -means, spectral clustering and Mean Shift is less impressive than at Rock's optimal setting of  $n = 1000$ .

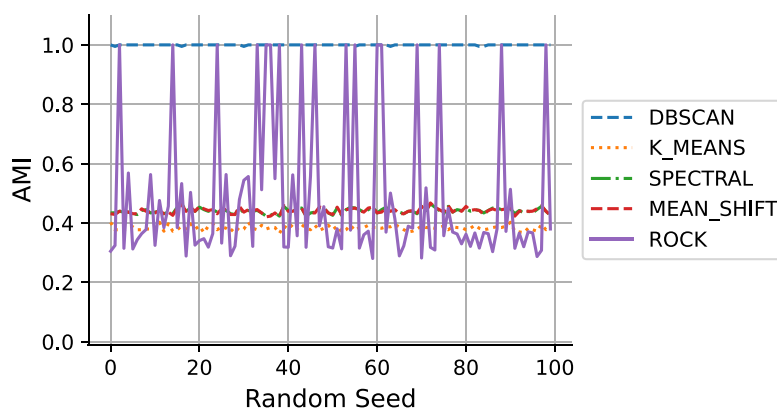
*Dimensionality.* The Blobs dataset is analyzed in Fig. 3b, varying the number of dimensions over  $\{2, 3, 4, 5, 10, 15, 20\}$ . The sample size is set at 300 and the number of generated blobs is 2, according to Table 1. Rock performs better than competitors mainly for small dimensions. Once the number of dimensions exceeds 5, Rock cannot outperform  $k$ -means and Spectral Clustering.

*Jitter.* The amount of jitter is varied for the Two Moons dataset, see Fig. 3c. We tried the following jitter amounts: 0.01, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30. The sample size is set to the optimal value of 1000 according to Table 1. Rock performs better than its competitors for the jitter set at 0.15 and above. However, for lower jitter values, Rock cannot outperform DBSCAN. Moreover, for jitter values of 0.25 and 0.30, the difference from Rock to  $k$ -means, spectral clustering and Mean Shift is quite low and not as impressive as at the optimal setting of 0.15.

To summarize, the performance of Rock is not robust with respect to variation of the data parameters, which leads to potential for over-optimization. While Rock is indeed better than its competitors for certain ranges of the data parameters, there are also settings for which Rock either does not perform better than the competitors, or the performance advantage is small. Thus the apparent "superiority" of Rock is generally less impressive than indicated by the results found from the TPE optimization in Table 1.

#### 4.1.3 Influence of the random seed

For the analyses mentioned so far, the mean AMI over ten random seeds was considered. However, it is also possible that a researcher chooses a particular random seed for which Rock performs well. As seen in Fig. 3c, Rock is outperformed by DBSCAN on the Two Moons dataset for a jitter value of 0.05 and 1000 samples. This statement is based on the AMI averaged over 10 random seeds. But could there also be particular



**Fig. 4** Performance of the cluster algorithms on the Two Moons dataset (sample size = 1000, jitter = 0.05) over 100 random seeds

random seeds for which Rock does perform well? In Fig. 4, we display the behavior of Rock and its competitors over 100 different random seeds. Since Rock performs as well as DBSCAN for some particular seeds, there is potential for an over-optimizing researcher to pick such a seed. While deliberately trying multiple seeds and presenting only the best one can be considered as malicious behavior, it is also possible that the seed set by the researcher is by chance a “good one”, and that the researcher does not consider a dependence of the performance on the random seed. To avoid such unintentional over-optimism, it is advisable to account for sampling variability and average over multiple random seeds, even when the cluster algorithm itself is deterministic. While the practice of sampling multiple datasets from a data distribution is well-known in statistics, this is sometimes neglected when evaluating data mining tasks like clustering.

#### 4.2 Optimizing the algorithm’s parameters

We analyze how the hyperparameter  $t_{max}$  of Rock can be optimized. In contrast to the previous sections, we now consider the absolute performance of Rock, given that a researcher would presumably not only try to outperform competitors, but also strive to obtain AMI values for Rock which are close to 1.

Additionally to Two Moons, Blobs and Rings, we consider the four real datasets mentioned in Sect. 3.2: Digits, Wine, Iris, Breast Cancer. For the Two Moons, Rings and Blobs datasets, we used the optimal data parameters from Table 1 and only generated a single illustrative dataset for each type by using 42 as a random seed. In accordance with typical evaluation of cluster algorithms, we do not split the datasets into training and test sets (see, however, the discussion in Sect. 6.2).

Figure 5 shows the performance of Rock as measured by the AMI, over  $t_{max}$  ranging from 1 to 30.

It can be seen that for different datasets, different  $t_{max}$  values are optimal. An optimistic researcher could report (only) the best  $t_{max}$  and the corresponding performance for each dataset. Optimizing hyperparameters of a cluster algorithm based on the “ground truth” of datasets (here via the AMI) is frequently seen in the literature.

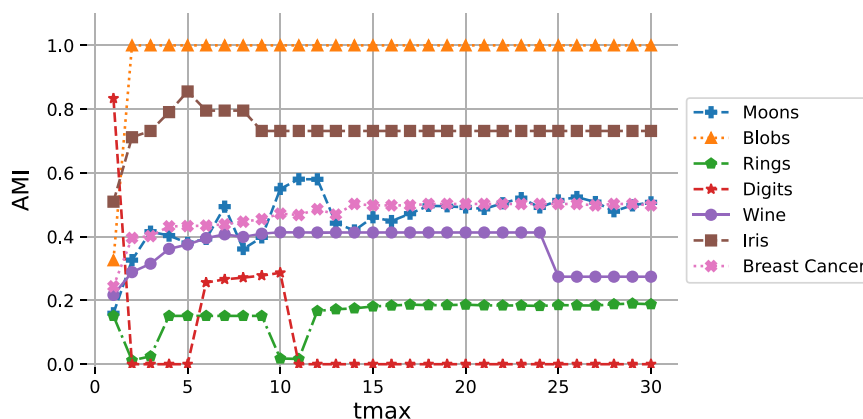


Fig. 5 Varying the hyperparameter  $t_{max}$  of Rock for different datasets

But as mentioned above, this could be over-optimistic with regards to the future performance of the algorithm: the evaluation of a novel algorithm is ultimately supposed to give hints about how well the algorithm will perform in future applications. But applied researchers usually do not know the “true” cluster labels of their datasets, as otherwise there would be no need for clustering. Thus the applied researchers cannot use a “ground truth” to determine a good  $t_{max}$  value for their specific datasets, and will thus obtain worse results for their datasets than the performances reported in the original paper which introduced the cluster algorithm. We will further discuss this issue in Sect. 5.1.

An alternative to reporting the best  $t_{max}$  for each dataset individually is to look for a  $t_{max}$  value that leads to good performance for multiple datasets. For example,  $t_{max} = 12$  yields reasonable performance values for Blobs, Two Moons and Iris. Thus, optimistic researchers might only report these three datasets with  $t_{max} = 12$  and claim that this choice of  $t_{max}$  will perform well for future datasets. However, such a statement would likely be over-optimistic as  $t_{max} = 12$  was chosen on only a few datasets, and considering the varied behavior of the different datasets for different  $t_{max}$  in Fig. 5.

Over-optimism can not only result from optimizing the hyperparameters of the novel algorithm, but also from simultaneously neglecting to optimize the hyperparameters of the *competing* algorithms. As an example, we compare Rock with DBSCAN on the Two Moons dataset, with the data parameters optimized for Rock from Table 1. Recall that in Sect. 4.1, we did not perform hyperparameter optimization, and instead used hyperparameter defaults or heuristics for the algorithms which could be reasonably justified (see also the appendix A): for Rock,  $t_{max} = 15$  as in the original paper of Beer et al. (2019), and for DBSCAN,  $minPts = 2 \cdot \#of\ dimensions$ , leading to  $minPts = 4$  for Two Moons, and  $eps = 0.2$ . The AMI for Rock for this case is  $0.7881 \pm 0.1583$  (mean and standard deviation over ten random seeds), see also Table 1. This mean value is different from the AMI value in Fig. 5 at  $t_{max} = 15$ , because a single seed was used for the latter. The AMI performance of DBSCAN was only  $0.0007 \pm 0.0024$ .

We then performed hyperparameter optimization for both cluster algorithms (with regards to the absolute AMI performance over ten random seeds). For Rock, we performed a simple grid search over  $t_{max} \in \{1, 2, \dots, 30\}$ . The optimal performance is at the previously used default  $t_{max} = 15$ , thus again yielding a mean AMI of  $0.7881 \pm 0.1583$ . This is not surprising, given that  $t_{max} = 15$  was used in Sect. 4.1 to optimize the data parameters of Two Moons such that Rock obtains superior performance (although the performance *difference* was used as the optimization criterion in that section). For DBSCAN, we performed hyperparameter optimization with the TPE, and obtained optimal parameters of  $minPts = 41$  and  $eps = 0.4$ , leading to a performance of  $0.8300 \pm 0.0244$ , which is a major improvement over the previous performance of DBSCAN. Thus DBSCAN outperforms Rock after hyperparameter optimization. This demonstrates that if researchers decide to perform hyperparameter optimization for the cluster algorithms to be compared, they should conduct the optimization not only for their own algorithm, but also equally carefully for all competing methods.

Returning to the topic of data type optimization (Sect. 4.1), Fig. 5 also shows the potential for picking specific datasets for which Rock performs reasonably well (e.g. Blobs, Iris, Two Moons) and discarding the ones with worse performance (Digits,

Rings). Again, *over-optimization* is marked by selective reporting: while no cluster algorithm can be expected to perform well on all types of data, it is still important to report data types for which a novel algorithm fails to detect clusters, to illuminate the limitations of the new method.

### 4.3 Optimizing the choice of competing algorithms

Here we revisit the results from Sect. 4.1 to analyze whether there is potential for picking specific competing cluster algorithms such that Rock appears better. For example, Fig. 3a–c show that Rock often performs better than DBSCAN, which was also due to neglecting hyperparameter optimization for DBSCAN, cf. Sect. 4.2. By picking suitable data parameter ranges, an over-optimistic researcher could praise the drastic performance improvement from Rock over DBSCAN. The same figures show that Rock is often better than Mean Shift. Thus, there is the potential for the following narrative: “Rock is an improvement of Mean Shift”. As the figures show, this claim would sweep some caveats under the carpet. For example, the other competitors, *k*-means and spectral clustering, are (almost) as good as Rock for the Blobs dataset in Fig. 3b.

## 5 Discussion

We have illustrated that selective presentation of performance results can lead to over-optimistic assessment of a novel cluster algorithm. Neglecting to show limitations of a new algorithm can lead to users applying it in inappropriate settings for the algorithm, which leads to unusable results. In this section, we discuss potential further aspects of over-optimism that we did not focus on, but would be interesting to study in future work.

### 5.1 Hyperparameter tuning and development of the algorithm

As explained in Sect. 4.2, the current standard of reporting the performance of a novel algorithm with hyperparameters optimized to the clustering “ground truth” (e.g., with a grid search) is likely over-optimistic. Using the ground truth of datasets for performance evaluation of a novel algorithm has a further drawback: as the number of datasets labeled by experts is limited, researchers using these datasets optimize their algorithm’s characteristics on these few labeled real world datasets, or alternatively use (unrealistic) synthetic datasets. Datasets such as Two Moons and Blobs are frequently used, but provide very limited information about how the cluster algorithm will perform in much more complex applied settings.

The optimization to a few datasets might not only concern the hyperparameters of the algorithm, but also the characteristics of the algorithm which are explored in the development phase. For example, Rock contains some “hidden hyperparameters” such as the growth rate of the number of neighbors considered in each iteration, or the weighting of the different nearest neighbours (Beer et al. 2019). These characteristics

are not intended to be changed by the user, but were decided on by the researchers during the development of the algorithm. However, if such characteristics are optimized according to the performance on just a few selected datasets, then this might result in an over-optimistic “overfitting” effect.

## 5.2 Evaluation measure

For all our experiments in this paper we used the Adjusted Mutual Information (AMI) as measure for the quality of clustering. Other partition similarity indices such as the Normalized Mutual Information (NMI, Strehl and Ghosh 2002), Adjusted Rand Index (ARI, Hubert and Arabie (1985)), Accuracy and F1-measure are often used in the field (see also Albatineh et al. (2006), for an overview). They all range in  $[-1, 1]$  resp.  $[0, 1]$  and describe how well the clustering results correspond to a ground truth, but have slightly different behaviors (Pfitzner et al. 2009). These indices are also called *external validation* indices, because they require an externally known partition (the ground truth) for evaluation. Yet evaluating a clustering based on the given “ground truth” might not always be the best choice. There could be interesting cluster structures in the data which differ from the given “true” labels, particularly because there is no unique definition of what a “good” clustering is (Hennig 2015). Moreover, as pointed out above, many real world datasets do not come with given labels. Thus researchers might also use internal validation indices (Halkidi et al. 2015) which do not require knowledge of the “true” labels, but evaluate a clustering based on internal properties of the data alone. Popular internal indices which measure within-cluster homogeneity and between-cluster heterogeneity/separateness include the Average Silhouette Width index (Kaufman and Rousseeuw 2009), the Caliński-Harabasz index (Caliński and Harabasz 1974), and the Davies-Bouldin index (Davies and Bouldin 1979). Such indices can also be used for performance evaluation of novel clustering algorithms, yet they might be susceptible to the over-optimism mechanisms outlined above. For example, researchers could optimize datasets and data characteristics with respect to an internal index, such that this index indicates a good performance for the new cluster algorithm, analogous to the optimization with the AMI discussed in Sect. 4.1.

The multitude of possible evaluation criteria—external or internal – gives rise to another potential source of over-optimism: Researchers could try different measures and pick the one that is most favorable to their novel algorithm. While researchers might be understandably uncertain about which evaluation measure to choose, they should not try different measures and then pick only the most favorable one *after* having seen the results. Researchers should carefully consider before starting the experimental evaluation which performance criterion is of particular interest in the considered context. If multiple measures are tried, then these should all be reported.

## 5.3 Preprocessing

Preprocessing the data can significantly influence the results of clustering. In our study, we scaled all the datasets. There are different normalizations that may be applied to the data, as well as methods to remove outliers or noise to improve the clustering results.

To avoid over-optimism, researchers should refrain from trying different preprocessing methods and reporting only the one most favorable to their new algorithm. Moreover, the same preprocessing steps should be applied to all datasets and *for all compared cluster algorithms*. Otherwise, if only the new algorithm is combined with suitable preprocessing, it might have an unfair advantage. A clear distinction should be made between preprocessing steps and steps belonging to the new cluster algorithm.

## 5.4 Theoretical evaluation

While we focus on the experimental evaluation of cluster algorithms with simulated or real-world datasets, it would also be interesting to study over-optimism in the context of theoretical analyses of algorithms. For example, researchers often make claims about their novel algorithms which they prove mathematically. But they could use very specific assumptions to yield the desired results. It might not always be easy for readers to judge how unrealistic these assumptions are, i.e., to which extent the assumptions restrict the use of the algorithm in real-world applications. Authors should thus always make their theoretical assumptions very clear, and thoroughly discuss how restrictive they are.

While theoretical analyses can, in principle, be affected by over-optimism, they are often a vital part of the evaluation of novel cluster algorithms. Theoretical results, if carefully deduced, can give a more complete picture of the algorithm's capabilities. Authors who thoroughly analyze their novel algorithm from a theoretical perspective might also use this background knowledge to choose a suitable and clearly defined experimental study design, such that unintentional over-optimization in the experimental part of the analysis could sometimes be partially avoided.

## 6 Possible solutions

As we have illustrated, there might be a strong over-optimistic bias when introducing a new cluster algorithm. How can such a bias be avoided or corrected? We discuss three options that all researchers can consider using in their research: (1.) avoiding selective reporting and analyzing robustness, (2.) evaluating the new method on independent data, and (3.) performing neutral benchmark studies. Moreover, we discuss (4.) how changing incentives in research culture and the publication system (that are beyond the control of individual researchers) might help to reduce over-optimism.

### 6.1 Avoiding selective reporting and analyzing the robustness of the algorithm

Our results have shown that over-optimistic presentation ultimately requires a certain amount of *selective reporting*, i.e., reporting only specific scenarios in which the new algorithm performs well. This might happen if many different scenarios are tried and only the "best" ones are reported, while the others are buried in the file drawer. Researchers might also omit the analysis of certain scenarios a priori, for example, when only considering data simulated according to a specific model. Such constraints

should be clearly explained, and the performance of the algorithm should not be oversold.

In the context of *model-based* cluster algorithms (see McLachlan et al. (2019) for an overview), selective reporting might be easier to detect. For example, if mainly datasets generated by the model of the newly developed algorithm are chosen, and/or the novel algorithm is compared with competing methods that were developed for the detection of clusters generated by other models, then the novel algorithm immediately has an advantage, which can be easily spotted. Nevertheless, there is still potential for an over-optimistic selection of datasets and comparative methods among all “reasonable” possibilities. Moreover, other potential sources of over-optimism discussed above, such as (hyper)parameter optimization, are also existent for model-based clustering. Readers and reviewers of articles about novel model-based cluster algorithms should keep this in mind, and the authors themselves must be careful to avoid over-optimistic choices.

Ideally, researchers should report scenarios in which their algorithm performed worse, to give a more realistic picture of the limitations of the novel approach. This may also require researchers to check the robustness of their algorithm (cf. Sect. 2.3): if the cluster algorithm is not robust with respect to certain data parameters, this should be honestly reported. Discussing the evaluation results for various parameter choices could also be beneficial as there is often not a single “best” choice and different parameters could be useful in different applications (Cerioli et al. 2018).

## 6.2 Validation on independent data

It is advisable to evaluate a new algorithm’s performance on fresh data that was *not* used for developing the algorithm and assessing its performance (Jelizarow et al. 2010). As we have demonstrated in Sects. 4.1 and 4.2, looking for specific data parameters or tweaking the algorithm’s hyperparameters might cause unintentional overfitting to the datasets used during the research process. As discussed in Sect. 5.1, overfitting to the used datasets could also concern the algorithm’s characteristics that were engineered in the development phase. The algorithm might not perform quite as well on new data, which would constitute a more realistic assessment of its performance.

More realistic performance values might also be obtained by taking inspiration from supervised classification and splitting the used datasets into “training” and “test” sets (Ullmann et al. 2021). Then hyperparameters such as  $t_{max}$  are optimized on the training set, and the chosen  $t_{max}$  is evaluated on the test set to assess performance. This could partially avoid “overfitting” of the hyperparameters to the data. However, a) this splitting procedure does not say anything about the performance on genuinely new data/data from different distributions, and b) when using the ground truth for optimization on the training set, this does not solve the problem that applied researchers who wish to use the new cluster algorithm in practice usually do not know the ground truth of their datasets, and thus cannot use the hyperparameter optimization procedure of the original authors. Therefore, it is advisable for authors who introduce a new algorithm to discuss and evaluate criteria for hyperparameter choice that do not require the ground truth, for example internal validation indices. Such indices could be used to

choose hyperparameters on the training set, and to evaluate the chosen hyperparameters on the test set to ensure that potential overfitting effects are detected.

### 6.3 Neutral benchmark studies

Awareness about the dangers of selective reporting and the importance of evaluation on fresh data might help to alleviate the problem of over-optimism. Academic teaching/training and illustrative studies such as ours can contribute to creating such awareness. Moreover, following guidelines for methodological computational research can help researchers avoid over-optimism (Boulesteix 2015). Ultimately, this will probably not solve the problem completely. Researchers are incentivized by the publication system to present their new algorithm favorably, which is unlikely to change in the short term (see 6.4). They are also more competent with respect to their own methods—and thus more likely to use them optimally than competing methods when conducting the evaluation. Thus, neutral benchmark studies are additionally required.

A neutral benchmark study is characterized by the comparison of existing algorithms (instead of the introduction of a new method), and neutrality of the authors, i.e., the authors do not have a vested interest in a particular method showing better performance than the others and are as a group approximately equally familiar with all considered methods, see Boulesteix et al. (2013, 2017) for an extensive discussion of these concepts. As mentioned in the introduction, neutral benchmark studies are less likely to suffer from over-optimism and usually offer a more realistic performance evaluation than studies presenting new methods.

In the field of clustering methodology, neutral benchmark studies are rarer than for supervised classification. Lately, however, there have been some advances: guidelines for performing benchmark studies for cluster algorithms were published in Van Mechelen et al. (2018). Following these guidelines, Hennig (2021) compared nine popular cluster algorithms, mainly with respect to various internal validation indices, but also regarding the recovery of the “true” clusterings. For an overview of previous cluster benchmark studies, see Van Mechelen et al. (2018) and Hennig (2021). In principle, the guidelines of Van Mechelen et al. (2018) could and should also be followed by non-neutral researchers who evaluate their new algorithm.

### 6.4 Changing incentives in the culture of research and the publication system

The three possible solutions presented so far are in principle accessible to individual researchers or teams of researchers. Ultimately, however, each researcher is subject to the constraints of the research and publication system. For example, researchers might hesitate to report limitations of their novel algorithm, because this could reduce their chances of getting published. Moreover, it can still be difficult to publish a neutral comparison study as many journals and conferences—stressing the importance of “novelty”—prefer studies introducing new methods (Boulesteix et al. 2018). In our view, changes in this attitude are necessary to further reduce over-optimism. Accepting neutral benchmark studies for publication should become more widespread. Furthermore, reporting limitations of novel algorithms should not be considered a “failure”

and instead an integral part of a healthy research culture. Journals and conferences should actively encourage authors to report scenarios in which their new algorithm does not perform optimally, or at least should not consider such reporting to be a cause for rejection. At the same time, editors and reviewers play an important role in filtering manuscripts in which authors do not carefully justify their experimental choices and only present very specific settings, which may be a hint that the results could potentially be over-optimistic. It should be taken into account, however, that even when a persuasive justification is given, the authors might still have arrived at these choices by (intentional or unintentional) over-optimization.

## 7 Conclusion

We have shown that studies which introduce new cluster algorithms might be affected by over-optimistic presentation of the results. For illustrative purposes, we have demonstrated different over-optimism mechanisms using the recently developed Rock algorithm as an example. While this is a specific example, we believe that these mechanisms might similarly apply to other novel clustering algorithms. We have also given some recommendations for avoiding over-optimism. It is our hope that going forwards, these guidelines will be taken into account. After all, overselling of novel methods does not contribute to genuine scientific progress.

**Acknowledgements** We thank Oliver Langselius and Anna Jacob for making valuable language corrections.

**Author Contributions** Conceptualization: TU (Lead), AB (Supporting), MH (Supporting), A-LB (Supporting)

Methodology: TU (Lead), AB (Supporting), MH (Supporting)

Software: MH (Lead), TU (Supporting), AB (Supporting) Validation: TU (Lead)

Writing—original draft preparation: TU (Lead), AB (Supporting), MH (Supporting), A-LB (Supporting), TS (Supporting)

Writing—review and editing: TU (Lead), AB (Supporting), MH (Supporting), A-LB (Supporting), TS (Supporting)

Funding acquisition: TS (Lead), A-LB (Supporting) Supervision: A-LB (Lead), TS (Supporting).

**Funding** Open Access funding enabled and organized by Projekt DEAL. This work has been funded by the German Federal Ministry of Education and Research (BMBF) under Grant No. 01IS18036A. The authors of this work take full responsibilities for its content.

**Availability of data and material** All used datasets are publicly available in scikit-learn for Python (Pedregosa et al. 2011).

## Declarations

**Conflict of interest** The authors declare that they have no conflict of interest.

**Code availability** Our fully reproducible code is available at: <https://github.com/thullmann/overoptimism-clust-algo>.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give

appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## A Appendix

In this appendix we give some details about the implementation outlined in Sect. 3.2. More information can be found in our fully reproducible code which is available at <https://github.com/anonresearcher461/over-optimism>. All experiment were performed with Python<sup>6</sup>, version 3.9.5.

### A.1 Adjusted mutual information (AMI)

Here we give the mathematical definition of the Adjusted Mutual Information Score (AMI, Vinh et al. 2010) which we use to compare the calculated clusterings with the “true” cluster labels. To define the AMI, we first discuss the entropy  $H$  of a single clustering and the Mutual Information (MI) of two clusterings. See Vinh et al. (2010) and Meila (2015) for more detailed explanations.

Let  $C$  and  $C'$  be two clusterings with  $k$  respectively  $l$  clusters. Let  $n_{ij}$ ,  $i = 1, \dots, k$ ,  $j = 1, \dots, l$  the number of data points which are in cluster  $i$  of  $C$  and cluster  $j$  of  $C'$ . Let  $n_{i\bullet}$  and  $n_{\bullet j}$  be the respective marginal sums, and  $n$  the overall number of data points.

The entropy  $H$  of clustering  $C$  is defined as

$$H(C) = - \sum_{i=1}^k \frac{n_{i\bullet}}{n} \log \left( \frac{n_{i\bullet}}{n} \right).$$

The entropy can be interpreted as the level of uncertainty associated with the clustering  $C$ . The Mutual Information (MI) of the clusterings  $C$ ,  $C'$  is defined as

$$MI(C, C') = \sum_{i=1}^k \sum_{j=1}^l \frac{n_{ij}}{n} \log \left( \frac{n_{ij}/n}{n_{i\bullet}n_{\bullet j}/n^2} \right).$$

The MI measures to which extent knowledge of the clustering  $C$  reduces uncertainty about the clustering  $C'$ . The MI is a symmetric measure, and it holds that

$$0 \leq MI(C, C') = MI(C', C) \leq \min(H(C), H(C')).$$

<sup>6</sup> <https://www.python.org>, visited: 05/31/21.

The MI can be normalized to ensure the measure ranges in  $[0, 1]$ , yielding the Normalized Mutual Information (NMI):

$$NMI(C, C') = \frac{MI(C, C')}{\text{avg}(H(C), H(C'))}.$$

Different choices for the “average”  $\text{avg}$  are possible, e.g., the arithmetic mean, the geometric mean, the minimum or maximum. We use the arithmetic mean (Kvalseth 1987), which is the scikit-learn default.<sup>7</sup>

Both the MI and NMI tend to increase with an increasing number of clusters, even if the information shared mutually between the clusterings does not actually increase. To account for this effect, the MI can be adjusted for chance: the MI of  $C, C'$  is compared with the expected MI for two random clusterings drawn from a permutation model (see Vinh et al. (2010) for details). The Adjusted Mutual Information Score (AMI) is thus calculated as follows:

$$AMI(C, C') = \frac{MI(C, C') - E[MI(C, C')]}{\text{avg}(H(C), H(C')) - E[MI(C, C')]} \quad (2)$$

The AMI attains its maximum value of 1 if the two clusterings are identical, and equals 0 if the MI between the two clusterings is equal to the MI value expected for two random partitions. Negative values occur if the agreement between  $C$  and  $C'$  is “worse” than chance.

## A.2 Scaling of the datasets

All datasets used in our study were scaled with the scikit-learn standard scaler<sup>8</sup>, by subtracting the mean and dividing by the standard deviation of each variable. That is, for each dataset  $D = (x_{ij})_{i=1, \dots, n, j=1, \dots, d}$ , with  $n$  samples and  $d$  dimensions, each entry  $x_{ij}$  is scaled according to

$$\frac{x_{ij} - \frac{1}{n} \sum_{i=1}^n x_{ij}}{\sqrt{\frac{1}{n} \sum_{i=1}^n (x_{ij} - \frac{1}{n} \sum_{i=1}^n x_{ij})^2}}$$

## A.3 Details about the blobs dataset

The Blobs dataset<sup>9</sup> consists of isotropic Gaussian clusters, i.e., each cluster  $k \in \{1, \dots, K\}$  (with  $K$  the number of generated clusters) corresponds to a Gaussian distribution with covariance matrix  $\sigma_k^2 I_d$ , where  $\sigma_k^2 \geq 0$  and  $I_d$  is the  $d$ -dimensional

<sup>7</sup> [https://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted\\_mutual\\_info\\_score.html](https://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_mutual_info_score.html), visited: 05/31/2021.

<sup>8</sup> <https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html>, visited: 05/31/2021.

<sup>9</sup> [https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make\\_blobs.html](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs.html), visited: 05/31/2021.

identity matrix. We chose a standard deviation of  $\sigma_k = 3 \frac{k}{K}$  for each cluster  $k$ . This generates different variances for the clusters, making some clusters more compact and thus easier to detect, and others more scattered and harder to find.

#### A.4 Bayesian optimization (BO) and the tree-structured parzen estimator (TPE)

BO approaches (see Shahriari et al. (2016) for an introduction) are popular for optimization problems of the type  $\operatorname{argmax}_{x \in \mathcal{X}} f(x)$ , where  $f : \mathcal{X} \mapsto \mathbb{R}$  is expensive to evaluate. In each step of a BO procedure,  $f$  is modelled with a *surrogate model*, based on a library of evaluations of  $f$  from previous steps:  $((x^{(1)}, f(x^{(1)})), \dots, (x^{(k-1)}, f(x^{(k-1)})))$ . The surrogate model is used to construct an acquisition function, which is cheaper to evaluate and easier to optimize than  $f$ , yielding the optimal argument  $x^{(k)}$ . Then  $(x^{(k)}, f(x^{(k)}))$  is added to the library, and the process is repeated by updating the surrogate model. The concrete surrogate model and the acquisition function of the TPE were chosen by Bergstra et al. (2011) such that optimization of the acquisition function ultimately leads to optimization of  $x \mapsto l(x)/g(x)$ , where  $l(x), g(x)$  are two Gaussian Mixture Models.  $l(x)$  is fitted to the observations  $(x^{(i)})_i$  that performed well so far, i.e., for which  $f(x^{(i)}) > y^*$  for some threshold value  $y^*$ .  $g(x)$  is fitted to the remaining observations. The threshold  $y^*$  is chosen as a quantile of the observed  $y^{(i)} = f(x^{(i)})$  values, such that  $p(y > y^*) = \gamma$  for a suitable  $\gamma \in (0, 1)$ . For more details on the TPE, see the original paper of Bergstra et al. (2011), the Optuna documentation<sup>10</sup>, and our reproducible code.

#### A.5 Default settings for the hyperparameters of the cluster algorithms

For the analysis in Sect. 4.1 (optimizing datasets and data characteristics), we used defaults or heuristics for the hyperparameters of the cluster algorithms which a researcher could justify as “reasonable choices”. For Rock, we chose  $t_{max} = 15$ , as in the original paper of Beer et al. (2019). For  $k$ -Means and Spectral Clustering we used the number of ground truth clusters for the parameter  $k$  and the default settings from scikit-learn. For DBSCAN, we followed Schubert et al. (2017) to set  $minPts = 2d$  with  $d$  being the number of dimensions. Moreover, we set  $eps = 0.2$ , which can be seen as a sensible value, given that the samples were scaled to unit variance. For estimation of the bandwidth for Mean Shift we use the scikit-learn function `estimate_bandwidth`<sup>11</sup>.

## References

Akiba T, Sano S, Yanase T, Ohta T, Koyama M (2019) Optuna: A next-generation hyperparameter optimization framework. In: Proceedings of the 25rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp 2623–2631

<sup>10</sup> <https://optuna.readthedocs.io/en/stable/reference/generated/optuna.samplers.TPESampler.html>, visited: 05/31/2021.

<sup>11</sup> [https://scikit-learn.org/stable/modules/generated/sklearn.cluster.estimate\\_bandwidth.html](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.estimate_bandwidth.html), visited: 05/31/2021.

- Albatineh AN, Niewiadomska-Bugaj M, Mihalko D (2006) On similarity indices and correction for chance agreement. *J Classif* 23(2):301–313
- Beer A, Kazempour D, Seidl T (2019) Rock-let the points roam to their clusters themselves. In: Proceedings of the 22nd International Conference on Extending Database Technology (EDBT), pp 630–633
- Bergstra J, Bardenet R, Bengio Y, Kégl B (2011) Algorithms for hyper-parameter optimization. *Adv Neural Inf Process Syst NIPS* 24:2546–2554
- Bischl B, Binder M, Lang M, Pielok T, Richter J, Coors S, Thomas J, Ullmann T, Becker M, Boulesteix AL, Deng D, Lindauer M (2021) Hyperparameter optimization: Foundations, algorithms, best practices and open challenges. arXiv preprint [arXiv:2107.05847](https://arxiv.org/abs/2107.05847)
- Boulesteix AL (2015) Ten simple rules for reducing overoptimistic reporting in methodological computational research. *PLoS Comput Biol* 11(4):e1004191
- Boulesteix AL, Strobl C, Augustin T, Daumer M (2008) Evaluating microarray-based classifiers: an overview. *Cancer Inf* 6:77–97
- Boulesteix AL, Lauer S, Eugster MJ (2013) A plea for neutral comparison studies in computational sciences. *PLoS ONE* 8(4):e61562
- Boulesteix AL, Stierle V, Hapfelmeier A (2015) Publication bias in methodological computational research. *Cancer Informatics* 14(S5):11–19
- Boulesteix AL, Wilson R, Hapfelmeier A (2017) Towards evidence-based computational statistics: lessons from clinical research on the role and design of real-data benchmark studies. *BMC Med Res Methodol* 17:138
- Boulesteix AL, Binder H, Abrahamowicz M, Sauerbrei W (2018) On the necessity and design of studies comparing statistical methods. *Biometr J* 60(1):216–218
- Boulesteix AL, Hoffmann S, Charlton A, Seibold H (2020) A replication crisis in methodological research? *Significance* 17(5):18–21
- Buchka S, Hapfelmeier A, Gardner PP, Wilson R, Boulesteix AL (2021) On the optimistic performance evaluation of newly introduced bioinformatic methods. *Genome Biol* 22:152
- Caliński T, Harabasz J (1974) A dendrite method for cluster analysis. *Commun Stat* 3(1):1–27
- Ceroli A, García-Escudero LA, Mayo-Isacar A, Riani M (2018) Finding the number of normal groups in model-based clustering via constrained likelihoods. *J Comput Graph Stat* 27(2):404–416
- Chhabra A, Roy A, Mohapatra P (2020) Suspicion-free adversarial attacks on clustering algorithms. *Proc AAAI Conf Artif Intell* 34:3625–3632
- Davé RN, Krishnapuram R (1997) Robust clustering methods: a unified view. *IEEE Trans Fuzzy Syst* 5(2):270–293
- Davies DL, Bouldin DW (1979) A cluster separation measure. *IEEE Trans Pattern Anal Mach Intell. PAMI-* 1(2):224–227
- Dua D, Graff C (2017) UCI machine learning repository. <http://archive.ics.uci.edu/ml>
- Ester M, Kriegl HP, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: *KDD'96: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pp 226–231
- Ferrari Dacrema M, Boglio S, Cremonesi P, Jannach D (2021) A troubling analysis of reproducibility and progress in recommender systems research. *ACM Trans Inf Syst* 39(2):1–49
- Fukunaga K, Hostetler L (1975) The estimation of the gradient of a density function, with applications in pattern recognition. *IEEE Trans Inf Theory* 21(1):32–40
- Gan J, Tao Y (2015) DBSCAN revisited: mis-claim, un-fixability, and approximation. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, pp 519–530
- Goodfellow I, McDaniel P, Papernot N (2018) Making machine learning robust against adversarial inputs. *Commun ACM* 61(7):56–66
- Halkidi M, Vazirgiannis M, Hennig C (2015) Method-independent indices for cluster validation and estimating the number of clusters. In: Hennig C, Meila M, Murtagh F, Rocci R (eds) *Handbook of cluster analysis*. Chapman and Hall/CRC, Boca Raton, pp 616–639
- Hennig C (2015) What are the true clusters? *Pattern Recogn Lett* 64:53–62
- Hennig C (2021) An empirical comparison and characterisation of nine popular clustering methods. *Adv Data Anal Classif*. <https://doi.org/10.1007/s11634-021-00478-z>
- Hubert L, Arabie P (1985) Comparing partitions. *J Classif* 2(1):193–218
- Jelizarow M, Guillemot V, Tenenhaus A, Strimmer K, Boulesteix AL (2010) Over-optimism in bioinformatics: an illustration. *Bioinformatics* 26(16):1990–1998

- Kaufman L, Rousseeuw PJ (2009) Finding groups in data: an introduction to cluster analysis. John Wiley & Sons, Hoboken, NJ
- Kvalseth TO (1987) Entropy and correlation: some comments. *IEEE Trans Syst Man Cybern* 17(3):517–519
- Lloyd S (1982) Least squares quantization in PCM. *IEEE Trans Inf Theory* 28(2):129–137
- McLachlan GJ, Lee SX, Rathnayake SI (2019) Finite mixture models. *Ann Rev Stat Appl* 6:355–378
- Meila M (2015) Criteria for comparing clusterings. In: Hennig C, Meila M, Murtagh F, Rocci R (eds) *Handbook of cluster analysis*. Chapman and Hall/CRC, London, pp 640–657
- Ng AY, Jordan MI, Weiss Y (2001) On spectral clustering: Analysis and an algorithm. In: *Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic*, pp 849–856
- Norel R, Rice JJ, Stolovitzky G (2011) The self-assessment trap: can we all be better than average? *Mol Syst Biol* 7(1):537
- Nuzzo R (2015) How scientists fool themselves-and how they can stop. *Nat News* 526:182–185
- Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, Blondel M, Prettenhofer P, Weiss R, Dubourg V, Vanderplas J, Passos A, Cournapeau D, Brucher M, Perrot M, Duchesnay E (2011) Scikit-learn: machine learning in Python. *J Mach Learn Res* 12:2825–2830
- Pfitzer D, Leibbrandt R, Powers D (2009) Characterization and evaluation of similarity measures for pairs of clusterings. *Knowl Inf Syst* 19(3):361–394
- Schubert E, Sander J, Ester M, Kriegel HP, Xu X (2017) DBSCAN revisited, revisited: why and how you should (still) use DBSCAN. *ACM Trans Database Syst* 42(3):1–21
- Shahriari B, Swersky K, Wang Z, Adams RP, de Freitas N (2016) Taking the human out of the loop: a review of Bayesian optimization. *Proc IEEE* 104(1):148–175
- Strehl A, Ghosh J (2002) Cluster ensembles—a knowledge reuse framework for combining multiple partitions. *J Mach Learn Res* 3:583–617
- Tufte E (1983) *The visual display of quantitative information*. Graphics Press, Cheshire, CT
- Ullmann T, Hennig C, Boulesteix AL (2021) Validation of cluster analysis results on validation data: a systematic framework. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* e1444
- Van Mechelen I, Boulesteix AL, Dangl R, Dean N, Guyon I, Hennig C, Leisch F, Steinley D (2018) Benchmarking in cluster analysis: a white paper. *arXiv preprint [arXiv:1809.10496](https://arxiv.org/abs/1809.10496)*
- Vinh NX, Epps J, Bailey J (2010) Information theoretic measures for clusterings comparison: variants, properties, normalization and correction for chance. *J Mach Learn Res* 11:2837–2854
- Xu R, Wunsch DC (2010) Clustering algorithms in biomedical research: a review. *IEEE Rev Biomed Eng* 3:120–154
- Yousefi MR, Hua J, Sima C, Dougherty ER (2010) Reporting bias when using real data sets to analyze classification performance. *Bioinformatics* 26(1):68–76

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



## Chapter 10

# Towards a Learned Index Structure for Approximate Nearest Neighbor Search Query Processing

This chapter consists of a preprint version of the following publication:

[38] Maximilian Hünemörder, Peer Kröger, and Matthias Renz. “Towards a Learned Index Structure for Approximate Nearest Neighbor Search Query Processing”. In: *Similarity Search and Applications - 14th International Conference, SISAP 2021, Dortmund, Germany, September 29 - October 1, 2021, Proceedings*. Ed. by Nora Reyes, Richard Connor, Nils M. Kriege, Daniyal Kazempour, Ilaria Bartolini, Erich Schubert, and Jian-Jia Chen. Vol. 13058. Lecture Notes in Computer Science. Springer, 2021, pp. 95–103. DOI: 10.1007/978-3-030-89657-7\_8. URL: [https://doi.org/10.1007/978-3-030-89657-7\\_8](https://doi.org/10.1007/978-3-030-89657-7_8)

It was reproduced with permission from Springer Nature.

**Statement of Originality:** The idea behind this paper was thought of by P. Kröger and discussed with **M. Hünemörder** and M. Renz. **M. Hünemörder** wrote software and ran experiments and wrote the manuscript supported by P. Kröger and M. Renz.

- **Conception:** Kröger (Lead), **Hünemörder** (Lead)
- **Planning:** All authors
- **Execution:** **Hünemörder**
- **Manuscript:** **Hünemörder** (Lead), Kröger (Support), Renz (Support)

## Towards a Learned Index Structure for Approximate Nearest Neighbor Search Query Processing

Maximilian Hünemörder, Peer Kröger, and Matthias Renz

Institute for Computer Science  
Christian-Albrechts-Universität zu Kiel, Germany  
{mah, pkr, mr}@informatik.uni-kiel.de

**Abstract.** In this short paper, we outline the idea of applying the concept of a learned index structure to approximate nearest neighbor query processing. We discuss different data partitioning approaches and show how the task of identifying the disc pages of potential hits for a given query can be solved by a predictive machine learning model. In a preliminary experimental case study we evaluate and discuss the general applicability of different partitioning approaches as well as of different predictive models.

### 1 Introduction

Nearest neighbor (NN) search is prevalent in many applications such as image retrieval, recommender systems, and data mining. In order to process a NN query efficiently appropriate data structures (usually called index structures) that enable identifying the result of a query by examining only a sub set of the entire data set are typically used. Additional speed-up can be gained by approximate nearest neighbor (ANN) search that trades accuracy for query time which is acceptable in many applications.

In this short paper, we examine the applicability of a new emerging paradigm, so-called learned index structures (LIS), for ANN query processing. The idea of LIS has been coined in [1] where the authors show that an index for 1D search keys (e.g. a B+-tree) is essentially similar to a regression model: the index induces an ordering of the keys and stores the data objects according to this ordering on disc pages (blocks). The corresponding learning task is, given the keys (observations) as training data, to train a predictive model (function) that determines the physical page address for each key. Processing a query is then simply applying the predictive model to the query key, i.e., predicting the addresses of the blocks (pages) on disc where the results of the query are located. While this approach works pretty well for primary key search, such as exact match queries and range queries on 1D data, we present one of the first works towards extending LIS to multi-dimensional spatial queries such as (A)NN queries.

This work aims at exploring the general applicability of LIS for multi-dimensional indexing with a focus on ANN queries. We discuss the two basic challenges any index structure has to solve (see also Figure 1). First, the database needs to be partitioned in order to store the objects in a clustered way on disc pages. We propose a new partitioning that adapts to the real data distribution and is based on a specific  $k$ -Means clustering here, but any other partitioning scheme is possible, e.g. by simply taking the

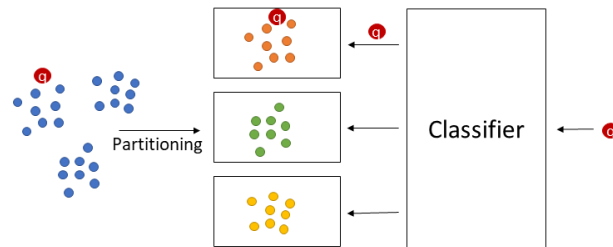


Fig. 1: A sketch of a spatial LIS: the the data (left) is partitioned and these partitions are mapped onto disc pages. A predictive model (classifier) learns this mapping. Given a query object  $q$ , the model predicts the disc page containing the potential NN of  $q$ .

leaf nodes of any hierarchical index structure. Second, the relationship between observations (values of the data objects) and their corresponding disc page IDs are learned using a predictive model. An ANN query can be supported by applying the learned prediction function to the query object. Since the predictive model may be not 100% accurate, the predicted disc page may not contain the true nearest neighbor(s) and therefore only result in an approximation. We will discuss implications, potential extensions, etc. on this aspect in detail. This way, a LIS could offer a good compromise between existing indexing paradigms: it could combine

1. a data-centric partitioning which is usually done by hierarchical index structures such as search trees that typically suffer from higher query costs due to the traversal of the search tree,
2. a fast prediction of disc page IDs which can be generally achieved by hash functions that often suffer from data-agnostic partitioning which may lead to a large number of collisions (disc page overflows) and, as a consequence to higher query times.

The remainder is organized as follows. Section 2 discusses preliminaries and related work. We sketch an LIS for multi-dimensional ANN query processing in 3. A preliminary empirical evaluation is presented in 4, and 5 offers a summary and a discussion of directions for future research.

## 2 Background

### 2.1 ANN Query Processing: Preliminaries and Related Work

Given a query  $q$ , an number  $k \in \mathbb{N}$  and a distance measure  $dist$ , a  $k$ NN query around  $q$  on a data set  $\mathcal{D}$ ,  $\text{NN}_k(q)$ , retrieves the  $k$  objects having the smallest distance to  $q$  among all objects in  $\mathcal{D}$  (ties need to be resolved). Without loss of generality, we set the query parameter  $k = 1$  and omit it in the following. Sequentially scanning all data objects to retrieve the NNs involves loading all pages of the entire data file from disk. Since

this is usually not acceptable performance-wise, many approaches for speeding up NN search using indexing techniques have been explored in recent years. A further way to achieve speed-ups is to trade performance for accuracy of the results using approximate algorithms that may report false hits. These ANN algorithms usually implement one of the following index paradigms:

*Hierarchical indexes* are typically based on balanced search trees [2–4] that recursively split the data space by some heuristics until a minimum number of objects remain in a partition. All nodes of the search tree are usually mapped to pages on disk. Searching theoretically requires  $O(\log f)$  random page accesses on average for  $f$  data pages but the performance typically degrade with increasing data complexity.

*Hashing* such as locality sensitive hashing (LSH) and variants [5–8] applies one or more hash functions to map data objects into buckets (and store these buckets as pages on disc). If the number of objects in a bucket exceeds the maximum capacity of a page (e.g. due to an unbalanced partitioning), the objects are stored in any order on so-called "overflow pages" increasing the number of page accesses necessary to answer a given query. However, in the best case, query processing requires  $O(1)$  page accesses.

*Vector quantization and compression* techniques (e.g. [9–11]) aim at reducing the data set size by encoding the data as a compact approximated representation such that (approximate) similarity among data objects is preserved.

A significant comparison of the different methods under varying realistic conditions is a generally challenging task. Thus, a benchmarking tool for ANN algorithms have been proposed in [12]. However, we do not aim for benchmarking LIS with other approaches here but rather explore the general applicability of LIS to ANN queries.

## 2.2 Learned Index Structures

The term LIS has been introduced by [1] where the authors show how to represent an index structures as a learning task. This pioneering work proposes a LIS for indexing 1D keys and supporting exact match and range queries. In recent years, the term LIS has been also used for methods that utilize machine learning techniques to support any aspect of query processing, e.g. [13] where  $k$ NN distance approximations are learned in order to support reverse NN queries, or [14] where the authors propose a new approach to generate permutations for permutation based indexing using deep neural networks. The most similar approach to ours can be found in [15] and [16] where the authors propose a learned metric index for ANN search. In contrast to our work they learn a whole tree of prediction models to index a metric space.

## 2.3 Contributions

LIS may offer the best of two worlds in spatial query processing, i.e., a data-centric, collision-free partitioning of the database and a search method that returns a result in constant time w.r.t. page accesses even in the worst-case. In this short paper, we explore the applicability of LIS to ANN query processing. In particular, we propose a general schema of a LIS for ANN query processing and implement this schema with existing techniques, e.g.  $k$ -means clustering for data partitioning. We present some first results

on the performance of various predictive models from machine learning and derive implications for future work.

### 3 Towards a Learned Index for ANN Search

The data set  $\mathcal{D}$  is stored on disk in blocks (pages) of a fixed capacity  $c$ . Thus, depending on  $c$ ,  $\mathcal{D}$  is distributed over a set  $\mathcal{P}$  of  $p$  pages on disk. Processing an object  $o \in \mathcal{D}$  in RAM requires to load the entire page  $P_o \in \mathcal{P}$  on which  $o$  is stored.

The key to any search index is that the data objects are not randomly distributed over  $\mathcal{P}$ . Rather, objects that are similar to each other w.r.t. the distance  $dist$  should be placed on the same page. There are many possible solutions for producing such a clustered partitioning, e.g. using the buckets of LSH, the leaf nodes of a search tree or use an unsupervised learning method. Here, we experimented with  $k$ -means clustering, which aims at partitioning the data into  $k$  disjoint clusters maximizing the compactness of these partitions. The idea is, to use  $k$ -means in such a way, that the number of points assigned to each cluster is constrained by a minimum capacity (for efficient storage usage)  $c_{min}$  and a maximum capacity  $C_{max}$  in order to map each cluster to one data page ( $C_{max}$  usually depends on  $c$  from above). Extensions such as Constraint  $k$ -means [17] are able to cope with these issues but are computationally very complex. Instead, in our study, we propose to just use traditional  $k$ -means clustering. The points assigned to a cluster  $C_i (1 \leq i \leq k)$  are mapped to page  $P_i \in \mathcal{P}$ .

For query processing, we need to predict the page  $P \in \mathcal{P}$ , the query object  $q$  would have been placed on. This page likely contains the NN of  $q$  (depending on the partitioning, etc.). This prediction could be done by any machine learning model that can learn the mapping of an object to the corresponding disk page. Analogously to hashing, such a predictive model is a function

$$M : \mathbb{F} \rightarrow \mathcal{P}$$

from the feature space  $\mathbb{F}$  of the data into the set of data pages that depends on some model-specific parameters  $\theta_M$ . In general, we can learn (train) the corresponding parameters  $\theta_M$  from  $\mathcal{D}$  (and the corresponding partitioning  $C_1, \dots, C_k$ ). Given a query object  $q \in \mathbb{F}$  and a predictive model  $M$  trained on  $\mathcal{D}$ , we can predict the disk page  $P = M(q)$  by applying  $M$  on  $q$ . The page  $P$  can be loaded into main memory and the NN of  $q$  among all objects stored on  $P$  can be determined and returned as (approximate) result. Since our data partitioning does not produce overflow pages, we only need to access one page, i.e.,  $P = M(q)$ . Thus, the time complexity is guaranteed to be in  $O(1)$  in any cases (we can usually even assume that the model  $M$  fits into main memory). The accuracy of this procedure obviously depends on various aspects such as the accuracy of the prediction, the data partitioning, etc., some will be examined in Section 4. However, we consider the optimization of such aspects as an open challenge for future research, e.g. by aggregating more information from the partitions such as centrality measures, distance bounds, etc.

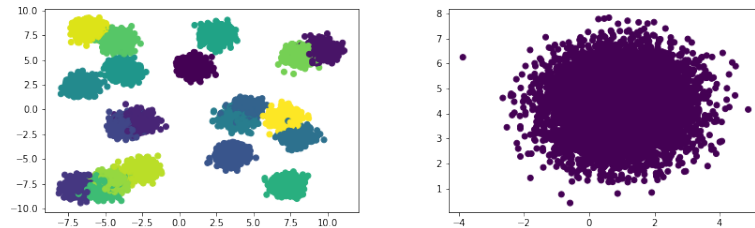


Fig. 2: Random 2D projections of sample clustered (left) and non-clustered (right) data.

## 4 Evaluation

### 4.1 Set-up

In order to get a first impression of the proposed LIS for ANN query processing, we used synthetic data sets generated by the `make_blobs` function from `sklearn`<sup>1</sup>. In all experiments, we generated five different random datasets and report average results. We conducted two general runs w.r.t. the data distributions: clustered and non-clustered data. Figure 2 depicts arbitrary 2D projections of two sample data sets from both runs. We used 20-dimensional synthetic datasets consisting of 5000, 10000, 30000 and 50000 samples. The clustered datasets had 20 clusters with a cluster standard deviation of 0.5 and the non-clustered datasets have only a single gaussian blob with a standard deviation of 1.0. Additionally, we used a low dimensional embedding of the popular MNIST data set generated by a fully connected Autoencoder (AE). Since this paper is a preliminary study of the general applicability of LIS to ANN search we did not yet compare to other ANN methods.

We used two different accuracy scores for evaluation. First, to explore the potential of the different predictive models to learn the mapping of objects to pages, we employed a classical train-validation split (called **validation accuracy**). Second, to measure the approximation accuracy of the query (called **test accuracy**), we used a withheld third sub-set of the data (not used in partitioning or training of the predictive model) as query objects, compared the results of these queries with the correct NN computed by a brute force search. The accuracy is determined by the ratio of the amount of zero distance hits and the amount of query objects. Additionally, we report the mean relative error for ANN search in our repository<sup>2</sup>.

For the partitioning step, we used the  $k$ -means implementation from `sklearn`. For comparison, we used the leaf nodes of a kd-tree (also from `sklearn`) as an alternative data partitioning. As predictive models, we used diverse classifiers from `sklearn`, including: Naïve Bayes, Decision Tree and Random Forest, Support Vector Machine (SVM) with a linear and an rbf kernel, and a simple dense multi-layer perceptron (MLP). For

<sup>1</sup> [https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make\\_blobs.html](https://scikit-learn.org/stable/modules/generated/sklearn.datasets.make_blobs.html)

<sup>2</sup> <https://github.com/huenemoerder/kmeans-lis.git>

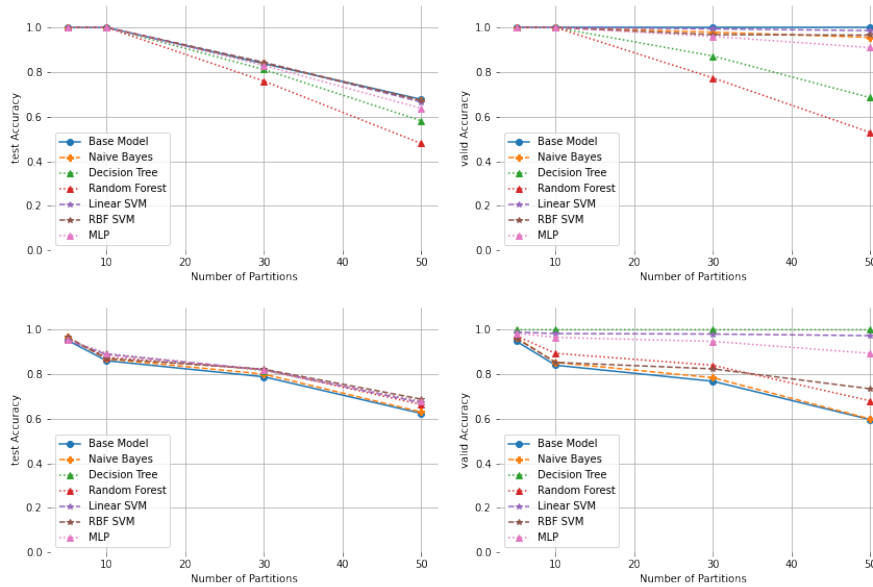


Fig. 3: Test accuracy (left charts) and validation accuracy (right charts) on **clustered** data sets (upper charts:  $k$ -means partitioning; lower charts: kdtree partitioning).

these preliminary experiments we did not perform hyper-parameter tuning but used reasonable default parameters. As a "Base Model", we assign each query object to its closest centroid of the corresponding partition (validation accuracy of 1.0 by design). The "size" of this model grows linearly with the number of partitions, i.e., database size, and is expected to not fit into the cache (requiring additional page accesses on application). The AE for the MNIST data set was implemented in pytorch<sup>3</sup> with only one single linear layer that maps the flattened images (784 dimensional array) to a latent space vector of 32 dimensions (using Leaky ReLU as activation).

## 4.2 Results

We analysed the relationship between the test accuracy and the number of samples and number of partitions, i.e., data pages. In all runs, we kept the capacity of pages fixed but changed the number of data points  $n$  accordingly. Figure 3 displays this relationship on clustered data sets. In general, we can see that both the test accuracy and the validation accuracy drops with increasing number of partitions. This is somehow intuitive: with increasing number of partitions (and data points), the mapping that has to be learned by the predictive model becomes more and more complex. It is interesting to note that for most models the validation error (right charts) remains better than the test accuracy (left charts), i.e. even though, the mapping is learned well, the true NNs for the query

<sup>3</sup> <https://pytorch.org/>

objects are approximated not quite as well. In these cases, the partitioning model seems to not optimally fit the real data distribution and therefore even with a perfect predictive model some queries can be placed in an unsuitable data page. This is also reflected in the fact that the kd-tree partitioning performs even worse in terms of test accuracy, since the clustered dataset was created in a way that favours  $k$ -means. We can also observe that the Decision Tree classifier shows perfect validation accuracy for the kd-tree partitioning, while showing the worst performance for  $k$ -means. This suggests that choosing a fitting pair of prediction and partitioning algorithm is vital to at least result in a high validation accuracy. These observations are further confirmed by the non-clustered data sets (the results can be found in our repository<sup>4</sup>). Additionally, this is further reflected in our results on MNIST in Table 1, where the test accuracies for the kd-tree partitioning are significantly worse than the ones for  $k$ -means. Generally further experiments and benchmarking are obviously necessary to obtain more significant results.

## 5 Summary

In this short paper, we applied the idea of LIS to ANN query processing and examined its general applicability to this problem. We explored a new data partitioning based on  $k$ -means clustering and applied the standard predictive models from machine learning in a simple set up. The results are generally promising for synthetic (clustered/non-clustered) and real data such that we think it is worth putting more future focus on LIS. For example, exploring new ways for data partitioning including a more thorough evaluation of different existing partitioning schemes could be interesting. Also, understanding the relationship between data characteristics, properties of the partitioning, and the accuracy of different predictive models could be a promising research direction that may lead to approaches that better integrate partitioning and learning. Additionally, exploring postprocessing methods to increase accuracy, e.g. use additional information from training as well as from the partitioning like distance bounds would be helpful. Last not least, the application of LIS to other types of similarity queries is still an open research question.

<sup>4</sup> <https://github.com/huenemoerder/kmean-lis.git>

Table 1: Results on MNIST data set ( $k$ -means partitioning)

Classifier	$k$ -means		KDTree	
	Validation Accuracy	Test Accuracy	Validation Accuracy	Test Accuracy
<i>Base Model</i>	<b>1.000</b>	0.8808	0.5407	0.4974
<i>Naive Bayes</i>	0.9140	0.8479	0.6140	0.5409
<i>Decision Tree</i>	0.8560	0.8121	<b>0.9997</b>	0.6160
<i>Random Forest</i>	0.7315	0.7089	0.4610	0.4066
<i>Linear SVM</i>	0.9973	0.8800	0.9630	0.6165
<i>RBF SVM</i>	0.9845	<b>0.8810</b>	0.8588	<b>0.6388</b>
<i>MLP</i>	0.9455	0.8736	0.8678	0.5994

## References

1. Kraska, T., Beutel, A., Chi, E.H., Dean, J., Polyzotis, N.: The case for learned index structures. In: Proc. Int. Conf. on Management of Data (SIGMOD), Houston, TX. (2018) 489–504
2. Ciaccia, P., Patella, M., Zezula, P.: M-tree: An efficient access method for similarity search in metric spaces. In: Proc. Int. Conf. on Very Large Databases (VLDB). (1997)
3. Sakurai, Y., Yoshikawa, M., Uemura, S., Kojima, H., et al.: The a-tree: An index structure for high-dimensional spaces using relative approximation. In: Proc. Int. Conf. on Very Large Databases (VLDB). (2000) 5–16
4. Amsaleg, L., Jónsson, B.P., Lejsek, H.: Scalability of the nv-tree: Three experiments. In: Proc. Int. Conf. on Similarity Search and Applications (SISAP), Lima, Peru. Volume 11223 of LNCS., Springer (2018) 59–72
5. Christiani, T.: Fast locality-sensitive hashing frameworks for approximate near neighbor search. In: Proc. Int. Conf. on Similarity Search and Applications (SISAP), Newark, NJ. Volume 11807 of LNCS., Springer (2019) 3–17
6. Jafari, O., Nagarkar, P., Montaña, J.: mmlsh: A practical and efficient technique for processing approximate nearest neighbor queries on multimedia data. In: Proc. Int. Conf. on Similarity Search and Applications (SISAP), Copenhagen, Denmark. Volume 12440 of LNCS., Springer (2020) 47–61
7. Jafari, O., Nagarkar, P., Montaña, J.: Improving locality sensitive hashing by efficiently finding projected nearest neighbors. In: Proc. Int. Conf. on Similarity Search and Applications (SISAP), Copenhagen, Denmark. Volume 12440 of LNCS., Springer (2020) 323–337
8. Ahle, T.D.: On the problem of  $p_{1-1}$  in locality-sensitive hashing. In: Proc. Int. Conf. on Similarity Search and Applications (SISAP), Copenhagen, Denmark. Volume 12440 of LNCS., Springer (2020) 85–93
9. Weber, R., Schek, H.J., Blott, S.: A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In: Proc. Int. Conf. on Very Large Databases (VLDB). (1998) 194–205
10. Ferhatosmanoglu, H., Tuncel, E., Agrawal, D., Abbadi, A.E.: Vector approximation based indexing for non-uniform high dimensional data sets. In: Proc ACM Int. Conf. on Information and Knowledge Management (CIKM), McLean, VA. (2000) 202–209
11. Houle, M.E., Oria, V., Rohloff, K., Wali, A.M.: Lid-fingerprint: A local intrinsic dimensionality-based fingerprinting method. In: Proc. Int. Conf. on Similarity Search and Applications (SISAP), Lima, Peru. Volume 11223 of LNCS., Springer (2018) 134–147
12. Aumüller, M., Bernhardsson, E., Faithfull, A.J.: Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. In: Proc. Int. Conf. on Similarity Search and Applications (SISAP), Munich, Germany. Volume 10609 of LNCS., Springer (2017) 34–49
13. Berrendorf, M., Borutta, F., Kröger, P.: k-distance approximation for memory-efficient k-NN retrieval. In: Proc. Int. Conf. on Similarity Search and Applications (SISAP), Newark, NJ. Volume 11807 of LNCS., Springer (2019) 57–71
14. Amato, G., Falchi, F., Gennaro, C., Vadicamo, L.: Deep permutations: Deep convolutional neural networks and permutation-based indexing. In: Proc. Int. Conf. on Similarity Search and Applications (SISAP), Tokyo, Japan. Volume 9939 of LNCS. (2016) 93–106
15. Antol, M., Ol’ha, J., Slanináková, T., Dohnal, V.: Learned metric index—proposition of learned indexing for unstructured data. *Information Systems* **100** (2021) 101774
16. Slanináková, T., Antol, M., Ol’ha, J., Vojtěch, K., Dohnal, V.: Data-driven learned metric index: an unsupervised approach. In: International Conference on Similarity Search and Applications, Springer (2021) To appear.
17. Bennett, K., Bradley, P., Demiriz, A.: Constrained k-means clustering. In: Technical Report MSR-TR-2000-65, Microsoft Research. (2000)



## Chapter 11

# SePass: Semantic Password Guessing using k-nn Similarity Search in Word Embeddings

This chapter consists of the following publication:

[39] Maximilian Hünemörder, Levin Schäfer, Nadine-Sarah Schüler, Michael Eichberg, and Peer Kröger. “SePass: Semantic Password Guessing Using k-Nn Similarity Search In Word Embeddings”. In: *Advanced Data Mining and Applications: 18th International Conference, ADMA 2022, Brisbane, QLD, Australia, November 28–30, 2022, Proceedings, Part II*. Brisbane, QLD, Australia: Springer-Verlag, 2022, pp. 28–42. ISBN: 978-3-031-22136-1. DOI: 10.1007/978-3-031-22137-8\_3. URL: [https://doi.org/10.1007/978-3-031-22137-8\\_3](https://doi.org/10.1007/978-3-031-22137-8_3)

It was reproduced with permission from Springer Nature.

**Statement of Originality:** The idea for this paper was initially conceived by M. Eichberg and L. Schäfer, based on their hands-on experience as LEA (Law Enforcement Agency). The main problem of semantic password guessing was then presented to P. Kröger, **M. Hünemörder** and N. Schüler, who added ideas on how to solve the problem. A major part of the software was then developed as part of L. Schäfer’s master thesis, supervised by **M. Hünemörder** and N. Schüler, which was then discussed with M. Eichberg and P. Kröger. Additional experiments on comparing related methods and other results and the necessary code were added by **M. Hünemörder** and run on additional data by M. Eichberg. L. Schäfer and **M. Hünemörder** mainly wrote the manuscript for the final paper, assisted by N. Schüler, M. Eichberg and P. Kröger.

- **Conception:** Eichberg (Lead), Schäfer (Lead), **Hünemörder** (Support), Schüler (Support)
- **Planning:** All Authors
- **Execution:** Schäfer (Lead), **Hünemörder** (Lead), Eichberg (Support)
- **Manuscript:** **Hünemörder** (Lead), Schäfer (Lead), Schüler (Support), Eichberg (Support) and Kröger (Support)

## SePass: Semantic Password Guessing using k-nn Similarity Search in Word Embeddings

Maximilian Hünemörder<sup>1\*</sup>[0000-0001-9848-3714], Levin Schäfer<sup>1\*</sup>, Nadine-Sarah Schüler<sup>2</sup>, Michael Eichberg<sup>3</sup>, and Peer Kröger<sup>1</sup>

<sup>1</sup> Christian-Albrechts Universität, Kiel, Germany

<sup>2</sup> Ludwig-Maximilians-Universität, Munich, Germany

<sup>3</sup> Federal Criminal Police Office, Germany

**Abstract.** Password guessing describes the process of finding a password for a secured system. Use cases include password recovery, IT forensics and measuring password strength. Commonly used tools for password guessing work with passwords leaks and use these lists for candidate generation based on handcrafted or inferred rules. These methods are often limited in their capability of producing entirely novel passwords, based on vocabulary not included in the given password lists. However, there are often semantic similarities between words and phrases of the given lists that are highly relevant for guessing the actual used passwords. In this paper, we propose *SePass*, a novel method that utilizes word embeddings to discover and exploit these semantic similarities. We compare SePass to a number of competitors and illustrate that our method not only is on par with these competitors, but also generates a significant higher amount of entirely novel password candidates. Using SePass in combination with existing methods, such as PCFG, improves the number of correctly guessed passwords considerably.

**Keywords:** password guessing · password cracking · semantic word embeddings · similarity search · nearest neighbors · law enforcement · nlp

### 1 Introduction

Password-protected devices such as notebooks, tablets, smartphones or secure hard drives are ubiquitous and, thus, can be central to criminal investigations. In such cases, gaining access to these devices might lead to crucial evidence and may help preventing further crime.

Up until today, passwords are still the primary mechanism to protect a user's private information, even though additional measures, such as two-factor authentication, are steadily added. A huge benefit of passwords is that they do not involve additional devices or resources and are safe if the underlying passwords have enough entropy. In that case the possible search space is plainly too large to be attacked using brute force search, at least in any reasonable time frame.

---

\* Both authors contributed equally to this research. The corresponding author is Maximilian Hünemörder ( mah@informatics.uni-kiel.de )

2 Hünemörder et al.

Compared to application scenarios, such as internet forums or online accounts, mobile device users need to type in their password to unlock their device frequently and need to be able to remember them. Using password managers or similar tools is usually not practicable to unlock the devices themselves. This typically encourages users to utilize passwords which consist of or resemble real words and are usually relevant to their everyday life, their culture or social environment with little to no modification. Nevertheless, guessing passwords in this context remains a significant challenge.

The most common approach to password guessing is a deductive approach: it uses dictionaries based on previous leaks, e.g., the *rockyou* leak [4,5], potentially combined with some proven set of rules, e.g., provided by tools like hashcat [13], to derive password candidates. This is often enough to guess a certain amount of passwords but it is obviously bounded by the limits of the deductive model, i.e., by the dictionary and the rule set.

This limitation cannot be overcome by extending the model, e.g., by using a more general purpose dictionary or ontologies, which significantly enlarges the search space or performing a brute force attack, which in turn renounces a focused strategy to traverse the search space for generating promising candidates.

In contrast, a data-driven approach is more promising since it predicts candidates without being limited by predefined terms or rules. Recently, machine learning methods using statistical models (e.g. [17]) or deep learning (e.g. [9]) have reported promising results for password guessing in general.

However, these methods may be typically too generic in specific applications and, thus, fail to incorporate the hidden semantics of typical passwords found in leaks. Even though these methods may be able to guess passwords that are based on vocabulary not included in the training set (i.e., the leaked lists).

For example, when analyzing famous leaks, it becomes evident that one domain for passwords are the names of luxury brands. But, even if a leak already contains brand names such as *Armani* and *Chanel* none of the existing tools would propose a password based on *Burberry* because this term would be syntactically too different from the previous two passwords – though being an *obvious* candidate. Furthermore, these predictions can usually not be tried out in any practical time frame.

In order to address this shortcoming a method is needed that is able to extend the vocabulary used for the predictions, i.e., new terms not seen in the training set. We propose **SePass**, a method to generate passwords based on the vocabulary of an existing password list by semantically extending the given vocabulary using word embeddings. Focused leaks, most prominently the *rockyou* leak, often trainingshow semantic similarities between words and phrases. Our proposed method SePass uses pretrained word embeddings to suggest additional, semantically similar words. These plain words could potentially be the basis for passwords used by people belonging to the same peer-group. We refer to these words as *base words*.

Real passwords are built from such base words but usually follow certain rules of modification or have additional characters added, i.e., prefixes/postfixes

to base words and/or combinations of multiple base words. By applying extracted password mangling rules to newly generated base words from our word embedding, we can extend the given list with additional passwords candidates exclusively found by SePass. We empirically show in our experiments that this will generate password candidates that are not produced by other methods. More importantly Sepass provides a new foundation on which existing or future methods can be built upon. Our experiments show, that SePass improves the prediction accuracy when combined with other existing methods. To summarize, the contributions of our paper are:

- We provide SePass, – to the best of our knowledge – the first method for password generation that unravels the hidden semantics in a password list by using word embeddings and, thus, is capable to semantically extend a given set of passwords.
- We present a working prototype implementation of a tool that addresses the creation of password candidates for people belonging to the same peer-group.
- We conduct an experimental study under realistic constraints comparing SePass with several state-of-the-art password generation methods.
- Our experiments show that using our proposed method as an augmentation to already existing password guessing methods, will improve both, the precision (number of correctly guessed passwords) and the effective time consumption (i.e. the number of guesses needed).

## 2 Related Work

Password guessing denotes the task of exactly matching an unknown string of characters used as a password for any kind of security system. Use cases include password recovery, IT forensics and measuring password strength. More generally, guessing a password is achieved by sequentially trying out password candidates until the correct one, then called a hit, is found. To be precise, passwords are generally not stored in clear text but rather as hash values. This requires that the true password must be recovered and is not readily available and therefore also limits the damage done by possible leaks. The used hash functions, such as *Sha512*, *PBKDF2* or *scrypt*, vary greatly in complexity but generally try to ensure that deriving the hash value from a given password always takes considerable time even on the most modern and specialized hardware. For the remainder of this paper, we only consider the basic problem of guessing the correct password. Hence, the concrete hash function is not relevant and is not further considered.

Password guessing methods differ by the way password candidates are generated. The most common methods for password guessing are brute force and dictionary attacks. A *brute force attack* consists of trying out all possible combinations of possible characters from a chosen alphabet to generate a password of a certain length. While brute forcing is the only method that guarantees a hit, it also evidently becomes unfeasible with increasing password length. *Dictionary attacks* on the other hand depend on lists of possible passwords, which are often

4 Hünemörder et al.

times collected or designed by experts. Another common source of dictionaries are data breaches and passwords leaks, e.g the *rockyou* leak consisting of more than 14 million password from the eponymous forum in 2009 [5].

While the methods mentioned above excel at generating vast amounts of password candidates they do not consider the plausibility of these passwords. In contrast, statistical password guessing methods utilize statistics based on existing password lists to focus on probable password candidates.

Assuming that human-created passwords are unlikely to be random combinations of characters, but rather follow a natural distribution stemming from the mother language of the user, password generation can be seen as a natural language processing (NLP) problem. Therefore statistical methods can be used to model the letter or character distribution of existing password lists and then sample new passwords. These kinds of methods aim for a high accuracy at a smaller amount of generated password candidates.

A method based on Markov modelling was introduced by Narayan et al. in 2005 [12]. The authors model the password distribution using a markovian assumption. Markov based NLP models predict which characters are likely to follow another character. Markov models intended for password guessing usually considers the last  $n - 1$  characters, so called  $n$ -gram Markov model. Then they modify sampled passwords by applying predefined regular expressions, i.e. the mangling rules. Currently Hashcat and JTR include such markovian models as an additional attack mode.

Building on Narayan et al.'s method, Dürmuth et al. introduced OMEN [6], which specifically sorts the generated password candidates in order of decreasing probability, something the original method was not capable of. A more general improvement of markovian models using neural networks was introduced by Melchier et al. [10]. Weir et al. [17] introduce a method that learns word mangling rules from existing password lists based on probabilistic context-free grammars (PCFG), a method stemming from NLP. They learn template structures of passwords by finding common and frequent patterns in clear text password leaks. For example, 'L4D8S1' would describe all passwords consisting of 4 lowercase letters followed by 8 digits and a single special character.

A semantic extension of Weir et al.'s PCFG was introduced as Semantic Password Guesser by Veras et al. [14]. They combine PCFGs with Wordnet [11] to enhance their grammars with semantic meaning. Their structures then use overarching semantic categories of words, i.e. umbrella terms instead of defining characters and numbers. These can then be used to describe the string of characters that is supposed to be placed at a certain part of a generated password candidate. An example base structure would be '*[sport][city][special]*' and a password generated from this could be '*footballhamburg?*'.

The semantic password generator is the most related approach to our method, but there are two major differences: First, they do not generate candidates based on words not present in the training data. Second, because their method is based on a hierarchical tree structure, they only consider a single context per base word.

For example, while the word “apple” would probably be categorized as a “fruit”, it is also semantically similar to “tech companies”.

In a further study from 2021 [15] the authors updated their method and investigated the semantic differences of commonly used password leaks. They found that semantic patterns found in some leaks correspond to the context of these leaks, i.e. the demographics of users of a forum or the general subject of the website the passwords were leaked from.

More recently, deep learning methods were introduced in order to depend less on strong assumptions about the word mangling rules that form passwords. These methods often use deep generative models and are trained on password lists to model the probabilistic space of passwords and can generate new passwords directly without applying rules.

An example of a deep generative model for password guessing is PasswordGAN [9]. This method uses a generative adversarial network, specifically a Wasserstein GAN [8] to generate large amounts of password candidates. In the course of their research, the authors found that the amount of candidates that need to be generated to reach similar or better results is significantly larger than those needed for statistical methods.

A review of other deep generative model architectures for password guessing was compiled by Biesner et al. in 2020 [2].

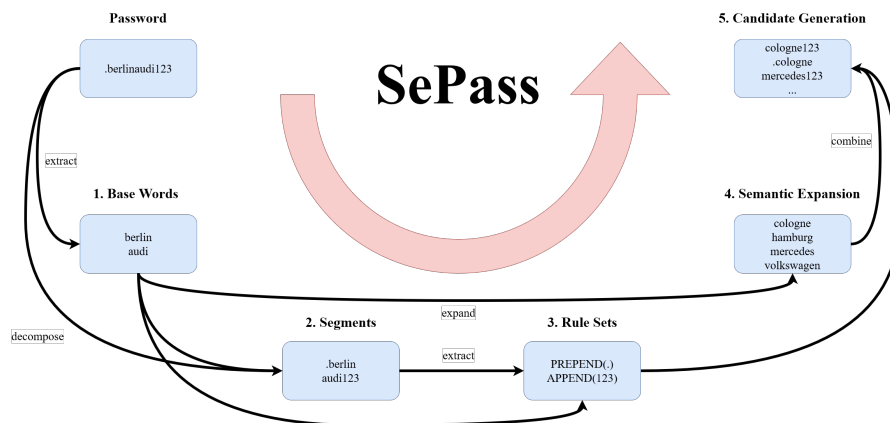


Fig. 1: Graphical illustration of the five steps of generating new password candidates.

### 3 Semantic Password Guessing

In this section, we describe our method and the procedures used to generate and sort a new candidate list for password guessing based on a well focused

6 Hünemörder et al.

password list stemming from a specific peer group. The main focus of our method lies on the semantic context of the *base words* extracted from the given list. Deploying word embeddings, we derive new base words that are semantically similar. In this context, a base word is a substring of characters that is included in a password and has some kind of semantic meaning and cannot be broken down without losing its meaning. For example, the German city name “Berlin” would be considered a *base word* for passwords like “berlin123”, “BeRl1naudi” as well as simply “Berlin”.

### 3.1 Generation of new password candidates

Under the assumption that base words used by a specific clientele or distinct group of people are semantically similar, we use pretrained word embedding models to exploit these semantic similarities. These models enable us to find similar words to expand the given password list with previously unseen vocabulary. Word embeddings are a popular method from natural language processing and allow for words and other character strings to be mapped into a high dimensional vector space in order to be used in downstream tasks [1, 16].

The goal is that semantically similar words are placed closely together according to some distance or similarity measure. For example, as euclidean distance is known for its adverse behavior in high dimensions, the cosine similarity is a popular choice when working with these high dimensional vectors. To obtain such a vector space, large-scale text corpora are processed. The resulting embedding is a vectorized representation of every single word in the training’s corpora, where we can assume that semantically similar words are also similar in the vector space.

For the current version of *SePass* we use state of the art pretrained word embedding models from the *FastText* [3] toolkit. These models are available in 157 different languages [7] and are light-weight, extensive and publicly available<sup>4</sup>. We use the 10 most relevant European languages based on general usage and leaks that we analyzed. Those languages are: *English, German, French, Italian, Spanish, Portuguese, Turkish, Dutch, Finnish*.

In addition to finding new base words using word embeddings, we need to generate actual password candidates from these novel base words using a set of word mangling rules. These rules are simple functions that transform a base word in a step-by-step manner into a password candidate. Examples for such functions can be adding, removing or replacing certain characters as well as changing single characters to upper or lower case and much more<sup>5</sup>. The rule set [PREPEND(x), APPEND(123), LOWER(), REPLACE(s, \$)] for example would transform the input word `Password` into `xpa$$word123`.

The following five steps, also summarized in Figure 1, describe how the proposed method takes a list of known passwords and generates additional candidates and rule sets for each word embedding.

<sup>4</sup> <https://fasttext.cc/docs/en/crawl-vectors.html>

<sup>5</sup> [https://hashcat.net/wiki/doku.php?id=rul\\_e\\_based\\_attack](https://hashcat.net/wiki/doku.php?id=rul_e_based_attack)

*Step 1: Extraction of the base words* Using a given word list  $P$  of known passwords, we extract base words from each password  $p$ , such that the list of base words for  $p$  is of minimal length and covers as many of the password as possible. This is done in two steps using the vocabulary  $V$  of a pretrained embedding model  $e$ . First, a decomposition into sub-words of  $p$  belonging to  $V$  is determined recursively. Hereby, the best decomposition is characterized by a minimum number of unused letters in  $p$ . In case of a tie, we prefer the solution with less base words. For example, for the password 'blueberry123a' the solution ['blueberry', '123'] wins against the solution ['blue', 'berry', '123'], each with a single unused character ('a') in the password. If no base words were detected for  $p$ , we additionally try to find non-obvious base words using the existing *rulegen* algorithm from PACK<sup>6</sup>. For example, to find the base words of passwords containing so called *leet speak*, i.e. replacements of characters with similar looking numbers such as 'passw0rd'<sup>7</sup>.

*Step 2: Decomposition into segments* For every password  $p$ , for which base words were found in Step 1, the password is split into multiple segments. Each segment contains exactly one base word. With the exception of the last segment of  $p$ , segments contain only the unmatched letters to the left of the base word. For example, the password berl i n?audi 123 would be split into the segments berl i n and ?audi 123.

*Step 3: Extraction of a rule set* Using existing methods from *rulegen*, based on all passwords in the source list  $P$ , a set of word mangling rules is derived such that all individual segments (from Step 2) can be created from the extracted base words (from Step 1). This is achieved by using the Levenshtein distance between the base words and the corresponding segments, e.g. Levenshtein distance (bberl i n, berl i n) = 1. We finally sort these rules by their occurrence frequency in  $P$ .

*Step 4: Semantic Expansion* This fourth step is the cornerstone of our method and also where it deviates the most when compared to previous work. Using a pretrained word embedding – or embeddings if multiple languages or corpora are used – we collect for each base word in the source password list the  $k$  most similar words in the vocabulary of the word embedding using a  $k$ -nearest neighbor query. Note, that  $k$  is not a hyperparameter. Instead  $k$  is calculated based on the number of password candidates that are intended to be generated.

Given, the number of pretrained models  $|E|$ , the intended number of password candidates  $n$  overall, the intended number of password candidates for a single embedding  $n_e = \frac{n}{|E|}$ , the number of rules generated in Step 3  $|R|$ , the hyperparameter *relevant ruleset ratio*  $rr$ , and the list of extracted base words  $BW_{old}$ , we first calculate the amount of base words we want to mangle

<sup>6</sup> <https://github.com/iphelix/pack>

<sup>7</sup> <https://github.com/hashcat/hashcat/blob/master/rules/unix-ninja-leetpeak.rule>

8 Hünemörder et al.

$$|BW| = \frac{n_e}{rr \cdot |R|} \quad (1)$$

and then

$$k = \frac{|BW|}{|BW_{old}|} \quad (2)$$

*Step 5: Generating new password candidates* Finally, by applying every rule from  $R$  to every word of the expanded base words list, we create the final list of new password candidates.

### 3.2 Sorting of the password candidates

Depending on the chosen initial word list and the parameters for the embedding, the newly generated list can grow in size considerably. This requires sorting the candidates based on the likeliness of being a real password – especially in cases where the time to guess a password is limited and does not allow trying out a large number of passwords. When executing the five steps described above, the candidate additionally is paired with a password score  $pws$ . The higher this score, the more suitable a candidate is considered to be. In accordance with the candidate being a combination of a base word and a rule, the password score is also made up of a word score  $ws$  and a rule score  $rs$  as shown in equation 3. The value of the rule score is simply determined by the relative occurrence of the specific rule in the total set of rules. The word score is calculated with the help of the embedding model. For every original base word we calculate how often a specific word  $w$  is present in the  $k$ -neighbors by using the same methods mentioned above. The sum of all these distances of the base words  $BW$  to  $w$  is used as word score for  $w$ . The formula for the calculation of the word score is shown in equation 4.

$$pws = ws \cdot rs \quad (3)$$

$$ws(w) = \sum_{i=0}^{|BW_{old}|} \begin{cases} \text{Cos}\Theta(BW_i, w), & \text{if } w \in knn(BW_i) \\ 0, & \text{otherwise} \end{cases} \quad (4)$$

## 4 Test Bed

In order to evaluate SePass and compare it with the current state-of-the-art, we performed a series of experiments. We primarily evaluate the use case of generating a list of novel password candidates from a relatively small training set, i.e., a highly focused leak, for example originating from a darknet or an extremist forum. As mentioned in Section 2, most other methods are based on learning candidates from large general password leaks and then testing the

generated candidate lists on other smaller leaks. To capture the characteristics of this real application scenario, we opt for a different evaluation scheme, where we only use a small password list, which we split into a train and a test sets. We trained and evaluated all models on a compute server running Ubuntu 20.04.3 LTS with 62 GB of RAM and an AMD Ryzen 7 3700X 8-Core Processor.

#### 4.1 Data Sets

We conducted all our experiments on two different datasets: a small real world list which is not yet widely available and is therefore only used for evaluation purposes and – for reproducibility purposes – we generated a second synthetic list which is a small excerpt of *rockyou* [5]. This synthetic list shares statistical similarities with the first list w.r.t. average length of the passwords, used languages and used rules. Both lists have the same length of entries, i.e., 66.490 passwords.

#### 4.2 Compared Methods

We compared SePass to the following password prediction methods that offer publicly available code repositories and represent the different existing paradigms of password guessing.

**Hashcat Best64** As a baseline we used hashcat with a basic rule set consisting of 64 word mangling rules that were created in a competition held by the community of hashcat<sup>8</sup>. These handcrafted rules are very simple instructions, such as appending single digits or letters, reversing the order of the password or replacing certain characters, for example, e with 3 or i with 1.

**OMEN** In order to represent the various markovian methods we utilized the original implementation of OMEN<sup>9</sup>. OMEN is one of the best performing probabilistic password guessers, meaning it uses candidate occurrence frequencies to output the most likely passwords. It was written in C, making it extremely fast compared with its competitors.

**PCFG** We picked Probabilistic Context-Free Grammars (PCFG) as a representative method based on statistical modeling. We used the pcfg cracker repository<sup>10</sup>, which was developed by one of the authors of the original publication [17]. As the authors mention in the notes on their repository, the tool is actually aimed at a similar use case as ours.

**Semantic PCFG** We chose this method because it is aimed at using semantic connections between words and, as such, follows a related concept to our approach. The authors have published their code on a git repository<sup>11</sup>.

<sup>8</sup> <https://github.com/hashcat/hashcat/blob/master/rules/best64.rule>

<sup>9</sup> <https://github.com/RUB-SysSec/OMEN/blob/master/README.md>

<sup>10</sup> [https://github.com/lakirw/pcfg\\_cracker](https://github.com/lakirw/pcfg_cracker)

<sup>11</sup> <https://github.com/vialab/semantic-guesser>

10 Hünemörder et al.

**PassGAN** We chose PassGAN [9] as one of the most well-known deep learning approaches for password generation. While we could not find a code repository from the original authors, we used a re-implementation<sup>12</sup> which contains a pretrained version of PassGAN.

### 4.3 Experimental Set-up and Evaluation Metric

We evaluated the accuracy of the competitors by splitting both our real and synthetic password lists into a training and a test set. The test sets each contain a random sample of 20 percent of the full lists. We applied each method to the training sets and generated a password candidate list each. We then compared these lists to our test sets. For PassGAN we did not train the model ourselves, but instead opted for the pretrained version that is included in the repository and was trained on *rockyou* [4], because our training sets would be magnitudes too small for PassGAN to be reasonably trained on. Still, this is a more than fair comparison, since both our training and test sets heavily overlap with the *rockyou* leak. We used the trained models to generate a list of 50 million password candidates to simulate a guessing attack on our test lists.

As usual in related work, our evaluation metric is the percentage of hits on the test set after  $n$  guesses, called hits@ $n$  which is defined as

$$\text{hits@}n = \frac{|P_m^{0\dots n} \cap P_{\text{test}}|}{|P_{\text{test}}|},$$

where  $P_m$  denotes the set of password candidates generated by a single method  $m$  and  $P_{\text{test}}$  denotes the attacked test set. We report the results of the competitors for  $n = 50$  million minus the number of duplicates in Table 1. In addition, we also report the hits@ $n$  value of the competitors in Figures 2a and 2b, which illustrate how quickly the corresponding methods may be able to successfully finish the attack.

Method	Hits@n in % on synthetic data	Hits@n in % on real world data	# of unique candidates generated
Ours	36.59	34.90	$50 \cdot 10^6$
hashcat-Best64	17.15		3,199,660
OMEN	32.35		$50 \cdot 10^6$
PCFG	36.52	39.39	$50 \cdot 10^6$
Semantic PCFG	20.22		24,903,549
PassGAN	15.27		24,761,815
Ours + PCFG	<b>44.25</b>	<b>45.33</b>	$50 \cdot 10^6$

Table 1: Prediction accuracy (hits@ $n$ ) and number of unique passwords generated after duplicate removal.

<sup>12</sup> <https://github.com/brannondorsey/PassGAN>

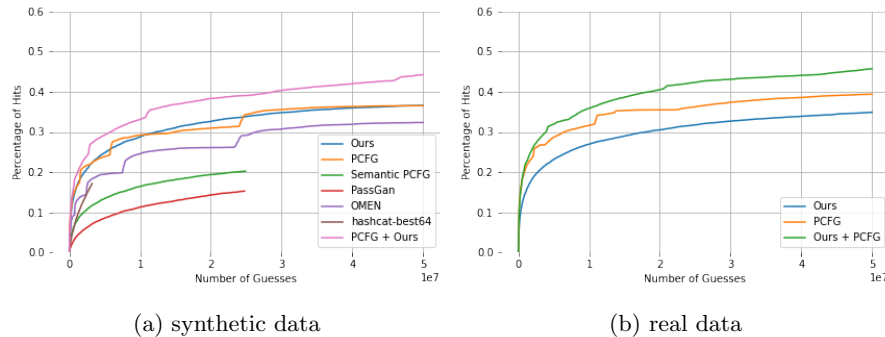


Fig. 2: Percentage of hits (hits@ $n$ ) with increasing number of guesses  $n$

## 5 Results and Discussion

In this section, we compare the results for the test bed and present further experiments performed to derive more insights into the strengths and weaknesses of the individual methods.

### 5.1 Accuracy Results

For reproduction purposes, the implementation of the experiments on the synthetic data set is publicly available<sup>13</sup>. We decided to only use the methods for the real data set that performed best on the synthetic list. Table 1 displays the hits@ $n$  results. In Figures 2a and 2b the hits@ $n$  are plotted as functions over  $n$  guesses, i.e., the effective time consumptions.

*Hashcat Best64* The hashcat Best64 rules seem to be a fitting baseline: while the method only produces a small number of unique candidates (e.g. 3.2 million on our synthetic data, i.e., the amount of passwords times 64 rules) a huge number of these are hits (17.5% correct guesses on the test set – see Table 1). Considering this method is based on applying fairly simple rules to mangle the base words from the training set, we can conclude that at least 17.5% of the passwords in our test set are rather trivially constructed. The graphs in Figure 2a seem to indicate that our method, OMEN and PCFG are able to guess these trivial passwords at a faster rate than hashcat, while Semantic PCFG and PassGAN are slower, but do or will eventually surpass this threshold as well.

*PassGAN* PassGAN performs the worst of all methods tested on the synthetic data set, as seen in Figure 2a. This is of particular interest since the pretrained PassGAN model was expected to have an advantage on our synthetic data set since PassGAN was trained on the rockyou leak and our synthetic data set consists

<sup>13</sup> <https://github.com/Knuust/SePass>

12 Hünemörder et al.

of mostly passwords found in rockyou. This might be explained by the fact that PassGan and similar GPU based methods are designed to generate exorbitantly huge amounts of guesses. Therefore, as the authors state in the PassGAN paper, it might take a lot more guesses before it catches up with the other methods. Combined with the amount of duplicates this method produces, we have to concede that PassGAN does not fit our use case, which is extrapolating from a small, focused vocabulary. Another explanation might be, that it does not guess passwords that stem purely from rockyou and it might even be at a disadvantage since it was not trained on our specific data set. And lastly, we used a 3rd party reimplementation of PassGAN, since no implementation from the authors is available, which might perform differently than originally presented. Therefore, as a consequence of the poor performance and not being suitable for our use case, we exclude PassGAN from experiments on the real dataset.

*Semantic PCFG* The Semantic PCFG password guesser seems to be performing better than PassGAN, but not as well as the original PCFG. This is surprising since the semantic PCFG method is based upon the original PCFG method. We assume the reason for this performance is comparable to the problems found with PassGAN.

*SePass, OMEN and PCFG* The best performing methods are OMEN, PCFG and our own method SePass. They all result in a similar percentage of hits at 50 million guesses, with our method SePass and PCFG coming out on top as seen in Table 1 and Figure 2a. As mentioned earlier this was to be expected since all three methods aim at a similar use case of giving more weight to accuracy in less guesses rather than generating a large amount of password candidates in a short time. One advantage specifically concerning the implementation of PCFG and OMEN is that both actually come with a few a priori rules, similar to the Best64 concept. For example, these rules include adding commonly used dates and keyboard walks (qwerty, etc.). These are applied additionally to enhance the base words and therefore lead to an enhanced performance for both OMEN and PCFG. PCFG and OMEN perform similar in Figure 2a. This can also be explained by the fact that the PCFG implementation is based on the OMEN repository.

Considering these results, we only ran our method and the best other method, i.e., PCFG, on the real data set. The results can be found in Table 1 and Figure 2b. It is evident, that when testing on this real world data set, PCFG outperforms SePass by a small percentage. To show that our method still provides additional benefit, we conducted an additional test, where we combined both lists proposed by SePass and PCFG.

*Combination* We combined both lists by zipping them together, i.e. by taking the first element of each, then the second, etc., which results in a list with double the length. Then we cut this down to 50 million guesses in order to compare them to the other methods. We can observe that the combination does indeed perform even better than the individual methods on both the synthetic and the

real dataset and are able to crack almost 50% of each test set. This is expected since our method adds the capability of using novel base words but can not generate the same amount of candidates using the mangling rules and the base words from the training set as PCFG does in 50 million guesses. This leads us to the conclusion that for a future version of our tool we should build upon the mangling rules of PCFG or other competitors and combine those with the novel base words found using our proposed approach.

## 5.2 Unseen Base Words

The main motivation behind our work was that existing methods, while very good at applying mangling rules to base words and creating passwords from existing lists, are generally not able to guess completely new base words without either using very specific handcrafted dictionaries or, at least partially, brute forcing passwords. We therefore investigated if and how well competitors find such new base words. Formally, given a vocabulary universe  $V$ , we are looking for base words  $B_{\text{new}} \subseteq V$  consisting of all words that are included in the passwords from the test set  $B_{\text{test}}$ , but can not be found in the corresponding training set,  $B_{\text{train}}$ , i.e.,  $B_{\text{new}} = B_{\text{test}} - B_{\text{train}}$ .

Firstly, in order to generate an extensive vocabulary universe, we collected the union of all vocabularies from the 10 language embedding models that we used in our method. This resulted in a set of exactly 12,953,300 unique base words. This is about 7 million words less than expected because while each model has a vocabulary of 2 million words, often times languages overlap and use the same terms. Because the models were trained on very large internet corpora the vocabularies can also include artifacts, e.g. very long words or numbers and special characters that can include outliers and errors. In order to investigate only natural words for the following experiments on novel base words, we removed everything from these vocabularies that includes any digits or other special characters.

Next, we searched for each word found in our vocabulary universe  $V$  in both the train and test set in both our password lists. We then subtracted the list of base words found in the train set from the ones found in the test set. This resulted in 13,428 novel base words, i.e. a set of base words that are used only in the test set but cannot be found in the training set.

Afterwards, we checked how many of these test base words can be found by our method and PCFG. We therefore look at the set of hits for each method, i.e. the intersection between the list of password candidates and the test set. We then search for each base word in these two sets and build the intersection with the set of base words contained in  $B_{\text{test}}$ . We found that SePass found 2,439 more novel base words than PCFG (which is almost 6 times more).

This demonstrates that SePass is able to extrapolate from the base words and significantly outperforms PCFG in this regard. On the other hand PCFG is also able to find a few novel base words. When taking a closer look at the new words that PCFG found, we can see that these often are random combinations

14 Hünemörder et al.

of existing words or predetermined rules, for example `qwertyuiop` which is explicitly included in the PCFG repository as a keyboard walk. In order to validate

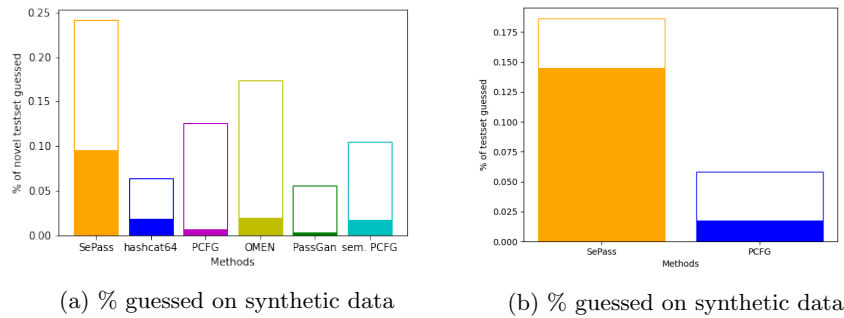


Fig. 3: Bar plots showing the percentages of passwords found in the synthetic (left) and real (right) test set, that include base words not found in the train set for each method. The filled part of each bar shows the percentage found exclusively by the corresponding method.

the performance of SePass, we found the corresponding passwords these novel base words were used for. This resulted in a list of 5,296 passwords, which consequently contain base words that are not included in the train set. The amount of passwords is lower than the amount of novel base words, since a password can include multiple base words.

We then calculated the percentage of novel passwords found by each method. The result is collected in a bar diagram in Figure 3 for both the synthetic dataset (a) and the real one (b). While the edge of each bar shows the percentage of passwords found, the filled areas represent the passwords this method found exclusively.

This means we see our expectations about SePass confirmed. Looking at the synthetic dataset, not only did SePass guess more of these novel passwords overall, SePass also finds significantly more novel passwords than any other method. Additionally, while PCFG performed better on the real dataset overall, SePass guesses 6 times more exclusive novel passwords on this dataset as well. In general, we can see that our method performs similarly well to related methods and is able to guess a significant amount of unique passwords.

## 6 Conclusion

We introduced SePass, a novel password guessing algorithm. The foundation of SePass are word embeddings which are used to identify new base words given the vocabulary extracted from a list of passwords. After that, we applied the

rules extracted from the passwords list to the found base words to generate password candidates that are semantically related to those found in the original passwords list. SePass compares favourably with the known methods used in this application field. It distinguishes itself from existing methods by being able to exclusively generate more passwords containing novel base words than any other method tested. We therefore conclude, that our tool, especially when used in combination with other methods like PCFG, can reach a high percentage of correctly guessed passwords, surpassing their individual scores.

## References

1. Almeida, F., Xexéo, G.: Word embeddings: A survey. CoRR **abs/1901.09069** (2019), <http://arxiv.org/abs/1901.09069>
2. Biesner, D., Cvejoski, K., Georgiev, B., Sifa, R., Krupicka, E.: Generative deep learning techniques for password generation (2020)
3. Bojanowski, P., Grave, E., Joulin, A., Mikolov, T.: Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics* **5**, 135–146 (2017)
4. Burns, W.J.: Common password list (rockyou.txt) (2019), <https://www.kaggle.com/wjburns/common-password-list-rockyoutxt>
5. Cubrilovic, N.: Rockyou hack: From bad to worse (2009), <https://techcrunch.com/2009/12/14/rockyou-hack-security-myspace-facebook-passwords/>
6. Dürmuth, M., Angelstorf, F., Castelluccia, C., Perito, D., Chaabane, A.: Omen: Faster password guessing using an ordered markov enumerator. In: *International Symposium on Engineering Secure Software and Systems*. pp. 119–132. Springer (2015)
7. Grave, E., Bojanowski, P., Gupta, P., Joulin, A., Mikolov, T.: Learning word vectors for 157 languages. In: *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)* (2018)
8. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., Courville, A.C.: Improved training of wasserstein gans. CoRR **abs/1704.00028** (2017), <http://arxiv.org/abs/1704.00028>
9. Hitaj, B., Gasti, P., Ateniese, G., Perez-Cruz, F.: Passgan: A deep learning approach for password guessing. In: *International Conference on Applied Cryptography and Network Security*. pp. 217–237. Springer (2019)
10. Melicher, W., Ur, B., Segreti, S.M., Komanduri, S., Bauer, L., Christin, N., Cranor, L.F.: Fast, lean, and accurate: Modeling password guessability using neural networks. In: *Proceedings of the 25th USENIX Conference on Security Symposium*. p. 175–191. SEC'16, USENIX Association, USA (2016)
11. Miller, G.A.: *WordNet: An electronic lexical database*. MIT press (1998)
12. Narayanan, A., Shmatikov, V.: Fast dictionary attacks on passwords using time-space tradeoff. In: *Proceedings of the 12th ACM Conference on Computer and Communications Security*. p. 364–372. CCS '05, Association for Computing Machinery, New York, NY, USA (2005)
13. Steube, J.: hashcat (2002), <https://hashcat.net/hashcat/>
14. Veras, R., Collins, C., Thorpe, J.: On the semantic patterns of passwords and their security impact (01 2014)

- 16 Hünemörder et al.
15. Veras, R., Collins, C., Thorpe, J.: A large-scale analysis of the semantic password model and linguistic patterns in passwords. *ACM Trans. Priv. Secur.* **24**(3) (apr 2021)
16. Wang, S., Zhou, W., Jiang, C.: A survey of word embeddings based on deep learning. *Computing* **102**(3), 717–740 (2020)
17. Weir, M., Aggarwal, S., Medeiros, B.d., Glodek, B.: Password cracking using probabilistic context-free grammars. In: 2009 30th IEEE Symposium on Security and Privacy. pp. 391–405 (2009)



## Chapter 12

# Stirring the Pot - Teaching Reinforcement Learning Agents a Push-Your-Luck board game

This chapter consists of a preprint version of the following publication:

©2021 IEEE. Reprinted, with permission, from [40] Maximilian Hünemörder, Mirjam Bayer, Nadine-Sarah Schüler, and Peer Kröger. “Stirring the Pot - Teaching Reinforcement Learning Agents a “Push-Your-Luck” board game”. In: *2022 IEEE Conference on Games (CoG)*. 2022, pp. 600–603. DOI: 10.1109/CoG51982.2022.9893657

**Statement of Originality:** The idea for this paper was conceived by **M. Hünemörder** and then discussed with N. Schüler. It was then used as a part of a master project for a group of students. Based on the preliminary work of these students the software was extended and more extensive experiments were conducted by M. Bayer, **M. Hünemörder** and N. Schüler. Finally, the manuscript was written up by **M. Hünemörder**, N. Schüler, M. Bayer and was reviewed and edited by P. Kröger.

- **Conception:** Hünemörder, Schüler
- **Planning:** Hünemörder, Schüler, Bayer
- **Execution:** Bayer, Hünemörder, Schüler,
- **Manuscript:** Hünemörder (Lead), Bayer, Schüler and Kröger

# Stirring the Pot - Teaching Reinforcement Learning Agents a "Push-Your-Luck" board game

Maximilian Hünemörder  
Christian-Albrechts-University  
Kiel, Germany  
mah@informatik.uni-kiel.de

Mirjam Bayer  
Christian-Albrechts-University  
Kiel, Germany  
miba@informatik.uni-kiel.de

Nadine-Sarah Schöler  
Ludwig-Maximilians University  
Munich, Germany  
n.schueler@lmu.de

Peer Kröger  
Christian-Albrechts-University  
Kiel, Germany  
pkr@informatik.uni-kiel.de

**Abstract**—Recent successes in AI research concerning traditional games like GO, have led to increased interest in the field of reinforcement learning. Modern board game design, however, has risen in complexity. This paper introduces a novel task for reinforcement learning: "Quacks of Quedlinburg". A modern board game with risk management, deck building, and the option to choose a specific rule set out of thousands of possible combinations for every game. We provide an environment based on the game and perform initial experiments. In these, we found that Deep Q-Learning agents can significantly outperform simple heuristics.

**Index Terms**—Risk management, Board Game Environment, Reinforcement Learning, DQN, DDQN

## I. INTRODUCTION

Board games have played a part in human history for thousands of years. Since games often serve as simple simulations for real life decision making processes, they provide interesting tasks for modern artificial intelligence research. For example, Chess holds a special place in the history of AI research [1], presumably because it is internationally played, has perfect information, is completely deterministic and reasonably brute forcible [2]. However, modern board games, especially so-called "German Board Games", e.g. "Settlers of Catan", have become much more complex. They incorporate tasks like resource or risk management, can involve social interaction and generally complex strategic decision-making.

A popular subdomain of artificial intelligence is reinforcement learning (RL). RL research and board games are linked since early work was based on learning to solve games like Tic Tac Toe, Checkers, and Blackjack [3]. While early RL agents performed worse than traditional algorithms on such games, the addition of deep learning techniques, e.g. Deep Q-Learning [4], lead to recent success stories for more complex games like GO [5]. These developments show that RL can even tackle problems that were thought to be very hard by using traditional AI.

Therefore, in addition to many modern board (and card) games, that have already been researched in the context of RL [6]–[11], we introduce the game "Quacks of Quedlinburg" as a novel and interesting RL task. This game from 2018 has a risk management component akin to Blackjack and a

Parts of this work has been done in the MARISPACE-X project funded by German Ministry of Economy (BMW)



Fig. 1: A player game board (cauldron) and a rule card for the red ingredients. The drop marker in the cauldron indicates the starting place each turn. The bag of ingredients in the top-right corner, exemplary ingredients and some rubies are placed around the game board. (Screenshot of Tabletop Simulator)

deckbuilding mechanic that strongly rewards planning multiple rounds ahead. For example, a riskier behavior in the beginning might lead to a higher reward in the later rounds of the game. The main reason why we believe this game in particular to be of interest to the RL research community is that it has a set of interchangeable rules, that allow for over 45.000 different rule sets. In this preliminary work, however, we only focus on a single set of rules. Our contributions can be summarized as follows:

- A novel application for RL based on the popular board game "Quacks of Quedlinburg"
- An implementation of this game in Python and a corresponding environment to be used by RL agents
- Preliminary experiments using simple heuristic AIs and the established RL algorithms Deep Q Network (DQN) and Double Deep Q-Network (DDQN)

## II. THE GAME

"Quacks of Quedlinburg" is an award-winning German board game by Wolfgang Warsch. In this game, each player embodies a quack doctor, who is mixing up their own brew

given various ingredients by successively drawing from an opaque bag. One of the ingredients however is spoiling the mixture and when a cumulative value of 7 is exceeded, this player's cauldron explodes and they are restricted in their following actions for this turn. Every turn of the game is played in two phases, a brewing and a scoring phase. While brewing, each player blindly pulls ingredients out of their own bag and places them in their cauldron in order, advancing on the drawn circular path, see Fig. 1. Each ingredient is marked by a color and a value, indicating additional rules to be applied throughout the game. In this process, the player can decide to stop drawing from their bag at any time, given their cauldron did not explode yet. After all players have either stopped or exploded, the second phase begins. In case of an explosion, the affected player has to choose to either forgo his earned VP for this turn or waive his chance to buy new chips. In the scoring phase, all advances on each player's cauldron are scored as *victory points (VP)* and marked on a common score board. Further ingredient rules are applied, then each player can use his advances from the previous phase to buy new ingredients (two distinct ones per turn) and add them to their bag. With every bought ingredient, the risk of pulling a white ingredient out of the bag is reduced. This is repeated for nine turns, after which the game ends and the player with most VP wins the game. Note, that in this preliminary work we use a simplified single-player version of the original game, as it contains the main mechanics and ideas but was reduced in complexity and some game features were omitted for now. (Full instruction sheet<sup>1</sup>)

We programmed the above described game in Python using a virtual representation of the score board (the cauldron) and the white, orange, green, red, blue, yellow and purple ingredients (black was omitted here as it only works in a multiplayer game). Each ingredient color is associated with a specific rule. The full game contains multiple different rules per color to choose from, as mentioned above, we will limit ourselves to one rule per color. Our selection can be found in Table I. One further resource of the game we implemented are *rubies*. A player obtains a ruby by either earning one from an ingredient (blue and green) or by stopping on a certain position. Pairs of rubies can be traded for advancing a *drop* placed in the middle of the cauldron, that corresponds to the starting position each turn. The rubies can further be saved up and count as additional VP at the end of the game, making collecting them a secondary objective.

All information of the game, such as the scoring board and each player's belongings, is saved in a collection of variables called *gamestate*. Each player's cauldron and ingredient bag are represented by lists of ingredients. Each ingredient is represented as a tuple containing the color and value of that chip (i.e. [*'white'*, 2]). For each of the nine turns, we store each player's temporary advances, such as preliminary victory points, earned money as well as whether

that player exploded or voluntarily stopped in the current turn. The number of rubies, the assured victory points and the starting position marked by the drop conclude the gamestate.

### III. EXPERIMENT SETUP AND COMPARED METHODS

To study this strategic decision task, we observed different AI agents playing the game. The implementation can be found in our public git repository<sup>2</sup>.

#### A. Environment

In order to train and evaluate the RL agents, we build an environment using the TensorFlow PyEnvironment class, that wraps around the implementation of the game itself. This allows us to return rewards to the RL agents based on the game state. The environment performs the agents' chosen action, yielding the game state update. A description of the exact observable game state and action space can be found in our repository.

1) *Reward*: We chose to use the earned victory points (VP) as the reward measurement for the agent. This guarantees that the reward directly corresponds to the main objective of the game. Additionally, we avoid skewing the learned strategies by rewarding actions that might be artificially enforced and do not lead to direct VP gain. Consequently, the task becomes more laborious as not every action results in victory point, i.e. an immediate reward. However the following section shows that this reward function is sufficient for the agents to learn that an action without an immediate gain can still result in a higher overall result.

2) *Legal Actions*: A challenge this game presents is that some game states entail that an agent cannot perform certain actions. The legal actions vary based on the current phase of the game, the chosen rule set and the game's progress. We therefore add a mask of currently legal actions to the observations the agents receive.

#### B. Random Agent

In order to be certain, that "*Quacks*" is not only based on pure chance we programmed a random agent, which randomly picks an action at each step of the environment. Even using the same method to bound the legal moves mentioned above, this agent is expected to perform poorly, because in the brewing phase the agent is faced with a string of binary decisions to draw an ingredient from the bag or stop drawing (similar to Blackjack). When randomly choosing at this point, the agent usually stops very early, resulting in little reward and no possibility to expand by buying more ingredients.

#### C. Heuristic Agents

For a more advanced baseline, we implemented a number of heuristic agents, that act in simple, predetermined ways.

<sup>1</sup><https://cdn.1j1ju.com/medias/ba/73/db-the-quacks-of-quedlinburg-rulebook.pdf>, Visited: 13.05.2022

<sup>2</sup><https://github.com/huenemoerder/quacks-rl>

Rule Color	Rule Text	Time to apply the rule	Costs (1, 2, 4)
Red	"If there are already orange chips in your pot, move the red chip up 1 or 2 places"	Instantly when drawn	6, 10, 16
Blue	"If this chip is on a ruby space, you IMMEDIATELY receive 1 ruby."	Instantly when drawn	4, 8, 14
Green	"If the last or second-to-last chip in your pot is a green chip, gain one ruby"	At the end of each turn	4, 8, 14
Yellow	"Your first placed yellow chip is moved 1 extra space, the 2nd yellow chip 2 extra spaces and the 3rd yellow chip 3 extra spaces"	Instantly when drawn (available from turn 2)	8, 12, 18
Purple	"For 1, 2 or 3 purple chips you receive the indicated bonus. 1: 1 VP - 2: 1 VP + 1 ruby - 3: 2 VP + 1 drop"	At the end of each turn (available from turn 3)	9

TABLE I: Overview of exemplary rules in the game (white and orange have no special rules)

1) *"Explosive" Agent*: The explosive agent is programmed to always draw as many chips as possible until its cauldron explodes so it is not allowed to draw chips anymore. Afterwards the agent always chooses to take the earned VP instead of buying any ingredients. Consequently, the contents of its bag stays the same and it can never advance far on the board, not gaining many VP.

2) *"Single-Color" Agent*: For each color ingredient (red, green, blue, yellow and purple) we designed different agents, that would draw chips until the risk to explode is higher than 70%. In the case of an explosion, the agent will buy chips in the first 6 turns and choose VP starting from turn 7. When buying, the agent is programmed to buy the most expensive chip available of its favored color and if there is enough money left, additionally buy one orange chip. A weakness of this type of agent is that it will not use its money efficiently especially in the later turns.

3) *"Expensive" Agent*: The expensive agent uses a strategy many first time (human) players choose. Analogously to the single-color agent, it draws chips from the bag until the risk of exploding is high (> 70%), however during the buying phase it buys chips that deliberately utilize all of its available money, i.e. it buys the most expensive and the second most expensive chip. This leads to less wasted money but more variation in the color choice.

#### D. DQN

The Deep Q-Network (DQN) approach was proposed in 2015 by DeepMind, [4] and was first applied on "Atari 2600 Games" [12]. Exemplary for our RL algorithms we trained two off-policy agents on the environment described in Section III-A. Each trained agent consists of a learned Q function which approximates the expected return at a given state ( $s$ ) for a single action ( $a$ ). This function is the agent's policy which enables it to select the best action to take given an input state. Due to our complex game that has an abundance of game state options, simple q-learning based on value iteration to fill a q-value table is not feasible. Instead the optimal Q function is approximating by training a neural network using a loss  $L(\theta)$ , that is computed during each training step by computing the difference of the predicted q values  $Q(s, a, \theta^{pred})$  to the target values  $Q(s, a, \theta^{target})$  of the Bellman Equation factored with  $\gamma$  added to the reward  $r$ .

$$L(\theta) = Q(s, a | \theta^{pred}) - (r + \gamma \max_a Q(s, a | \theta^{target}))$$

The used Q network architecture contains two dense hidden layers, the first layer with 150 and the second with 75 neurons.

Both layers use ReLu activation and are initialized using an truncated norm distribution. The output layer contains as many elements as the environment allows actions. The TensorFlow library, [13], provides a variety of pre-implemented agents including a DQN agent, which we initialized using this Q network. The policy was updated using the Adam optimizer and we used epsilon greedy exploration with a probability of 0.1. The agents were equipped with a TensorFlow Uniform replay buffer with a capacity of maximum 100,000 trajectories. Because all parties, i.e. the agent and the environment are Python based, integration was seamless. The serial interface is defined by the structure of the observation and action tensors, the two components which are passed in between the agent and the game. As mentioned in section III-A2, in order to communicate the legal actions at any state, the agent is given a mask that encodes this information. The mask is applied to the network's q values for a given state. The q values of any illegal action are set to the minimum of the q values, ensuring only legal actions are chosen. The training was performed using a batch size of 64 and a learning rate of 1e-4. The displayed victory points in Fig. 2a were achieved by the DQN agent after 267.000 training steps.

#### E. DDQN

In addition to the DQN agent a Double Deep Q-Network (DDQN) was trained. DDQN is an amplification of DQN proposed in 2015 by Hado van Hasselt et al. [14]. A DDQN agent takes advantage of fixed Q targets using an additional Q network, the target network. Additionally, this algorithm solves the overestimation that is to be expected from the DQN agent by using two networks. The local network will be used to calculate the single q target for the current state but the second network is used to calculate the q values in the loss objective. The same agent parameters and implementation base as for the DQN agent were chosen. The victory points shown in Fig. 2a were achieved by the DDQN agent after 425.000 training steps.

## IV. RESULTS

We evaluated each of the compared agents by calculating the average victory points over the same 1,000 seeded games. The seed ensures a fair comparison since the played games are deterministic. The results are shown in Fig 2a.

Starting from the left of the figure, as expected the random agent performs poorly, receiving 1.4 VP on average. The explosive agent performs better and is able to obtain an average of 18.5 VP. The single-color agents as well as the expensive

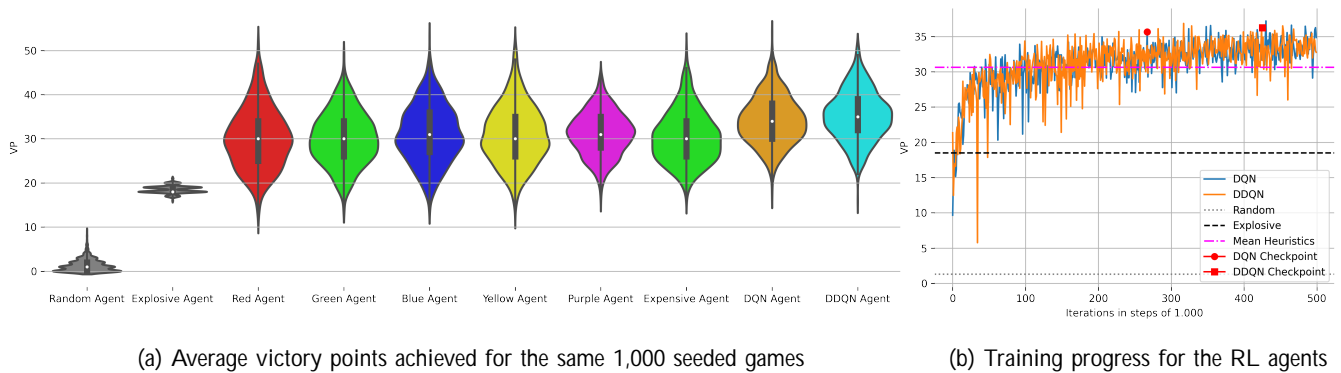


Fig. 2: Overview and comparison of all agents

agent all perform similarly, reaching an average VP score of around 30, (Red: 30.1, Green: 30.1, Blue: 31.0, Yellow: 30.9, Purple: 31.2, Expensive: 30.5). This implies that our chosen set of color rules is balanced and no color is significantly better on its own. The DQN and DDQN agents outperform the base line agents. Both RL agents yield a higher mean of VP, DQN with an average of 34.2 VP and DDQN with 35.2 VP respectively. Hypothetically, if pitted against each other the DQN agent would win 35.3 % of all seeded games against the heuristic agents, while the DDQN would win 42.6 %. When compared directly against each other the DDQN performs slightly better with a winrate of 53.1 %. We trained both agents for 500,000 iterations, we evaluated 20 random games each 1,000 iterations and picked the iterations where they performed the best. In Figure 2b these returned results are plotted and the checkpoints are marked. The average returns of the other agents are marked as horizontal lines. The agents were initialized with 500 games taken from a random policy, but almost instantly overtake the random agent and after 10,000 consistently reach a higher score than the explosive agent. After 100,000 iterations the agents have roughly found their optimum and outperform all the heuristic agents. The almost equal performance of DQN and DDQN might point to there not being a better strategy at least for the exact rules combination. We will explore this further in future work.

## V. CONCLUSION AND FUTURE WORK

Our hitherto research concludes that RL agents can learn the simplified version of the game and detect optimal strategies, despite the strong luck component. Following up on this, the next steps would entail extending our environment to support multiplayer and adding the corresponding rules and features. Besides that, we aim to train more advanced RL agents. So far, we only used one rule during training. In the future, we aim to extend the game to incorporate the 6 different rules every ingredient color can be assigned. In the full game, i.e. using all six ingredients, this leads to  $6^6$  different possible rule combinations. We would like to perform more studies on how agents would adapt to the application of the high variation of

rules available in the full game and how a generalized strategy for the enhanced game could look.

## REFERENCES

- [1] D. Heath and D. Allum, "The historical development of computer chess and its impact on artificial intelligence," in *Deep Blue Versus Kasparov: The Significance for Artificial Intelligence*, 1997.
- [2] C. Koch. (2016) How the computer beat the go master. [Online]. Available: <https://www.scientificamerican.com/article/how-the-computer-beat-the-go-master/>
- [3] R. Sutton. (1997) History of reinforcement learning. [Online]. Available: <http://www.incompleteideas.net/book/1/node7.html>
- [4] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.
- [5] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, 01 2016.
- [6] M. Pfeiffer, "Reinforcement learning of strategies for settlers of catan," 05 2022.
- [7] K. Xenou, G. Chalkiadakis, and S. Afantenos, "Deep reinforcement learning in strategic board game environments," in *Multi-Agent Systems*, M. Slavkovik, Ed. Cham: Springer International Publishing, 2019, pp. 233–248.
- [8] Q. Gendre and T. Kaneko, "Playing catan with cross-dimensional neural network," *CoRR*, vol. abs/2008.07079, 2020. [Online]. Available: <https://arxiv.org/abs/2008.07079>
- [9] R. Canaan, X. Gao, Y. Chung, J. Togelius, A. Nealen, and S. Menzel, "Evaluating the rainbow DQN agent in hanabi with unseen partners," *CoRR*, vol. abs/2004.13291, 2020. [Online]. Available: <https://arxiv.org/abs/2004.13291>
- [10] L. Perez, "Mastering terra mystica: Applying self-play to multi-agent cooperative board games," *CoRR*, vol. abs/2102.10540, 2021. [Online]. Available: <https://arxiv.org/abs/2102.10540>
- [11] D. Zha, K. Lai, Y. Cao, S. Huang, R. Wei, J. Guo, and X. Hu, "Rlcard: A toolkit for reinforcement learning in card games," *CoRR*, vol. abs/1910.04376, 2019. [Online]. Available: <http://arxiv.org/abs/1910.04376>
- [12] M. G. Bellemare, J. Veness, and M. Bowling, "Investigating contingency awareness using atari 2600 games," in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.
- [13] M. Abadi and A. A. *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: <https://www.tensorflow.org/>
- [14] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," *CoRR*, vol. abs/1509.06461, 2015. [Online]. Available: <http://arxiv.org/abs/1509.06461>



## Chapter 13

# Conclusion and Future Work

The recent success of data science and machine learning in real-world applications is reflected in how research in these areas is conducted. There is research with a more theoretical focus, and other research with an engineering approach to algorithm design or the design of so-called *data-to-pattern* pipelines. Such an application-driven approach may be particularly appropriate for unsupervised learning, because unsupervised learning relies on assumptions about a task and its data set. The stronger such an assumption is, the simpler the model can be. For example, *K*-means is a popular model not in spite of, but because it makes very strong assumptions about the exact number of clusters and the spherical nature of each cluster. Having easy-to-understand attributes is beneficial for domain experts using an algorithm because it allows them to apply their background knowledge more directly to the problem at hand.

This offers a large design space for future research. Designing methods with specific applications in mind and using the available domain and background knowledge might reduce future over-optimism in data science research. As illustrated in Chapter 9, meta-research into the practices of algorithm design can lead to finding essential insights on how to improve the scientific process for data science research.

This thesis has already explored the concept of incorporating background knowledge into unsupervised algorithms. For example, Chapters 6 and 7 introduce novel subspace clustering algorithms that use background knowledge and show that they can improve clustering results. Future work could focus on finding specific applications of subspace clustering and designing specific algorithms for those domains. In addition, comparing complex methods that use deep clustering with simpler methods that approximate nonlinear subspaces using linear subspace clusters could help to better understand the more complicated models. For CoMAD PCA (see Chapter 4) and CODEC (see chapter5), a similar approach of finding a specific application and comparing the result to more complex models may be an exciting avenue of research. OAB (Chapter 8) already provides a benchmark for anomaly detection that can be used to design new algorithms based on industrial applications. Learned indexes, see chapters 3 and 10, are already very explicitly designed for their application. However, spatial indexing in particular could still benefit from including background information about the specific database being indexed. Our work on semantic password guessing, see chapter 11, is an excellent example of using detailed knowledge about

the intended use case. We work with law enforcement agents (LEAs) to prioritize passwords that are more likely to match their clientele, rather than trying out password candidates that are unlikely to be created by the same people. Future work will include incorporating this semantic idea or other background information into other password guessing methods, and perhaps using context-specific embeddings, such as training a language model on a text corpus specific to the LEAs' tasks. A more elaborate approach might involve some conditional GAN architecture [60].

# Bibliography

- [1] Mohamed Abbas, Adel El-Zoghabi, and Amin Shoukry. “DenMune: Density peak based clustering using mutual nearest neighbors”. In: *Pattern Recognition* 109 (2021), p. 107589. ISSN: 0031-3203. DOI: <https://doi.org/10.1016/j.patcog.2020.107589>. URL: <https://www.sciencedirect.com/science/article/pii/S0031320320303927>.
- [2] Elke Achtert et al. “Deriving quantitative models for correlation clusters”. In: *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2006, pp. 4–13.
- [3] Elke Achtert et al. “Robust Clustering in Arbitrarily Oriented Subspaces”. In: *Proceedings of the 2008 SIAM International Conference on Data Mining (SDM)*, pp. 763–774. DOI: 10.1137/1.9781611972788.69. eprint: <https://epubs.siam.org/doi/pdf/10.1137/1.9781611972788.69>. URL: <https://epubs.siam.org/doi/abs/10.1137/1.9781611972788.69>.
- [4] Charu C Aggarwal et al. “Fast algorithms for projected clustering”. In: *ACM SIGMoD record* 28.2 (1999), pp. 61–72.
- [5] Charu C. Aggarwal and Philip S. Yu. “Finding Generalized Projected Clusters in High Dimensional Spaces”. In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’00. Dallas, Texas, USA: Association for Computing Machinery, 2000, pp. 70–81. ISBN: 1581132174. DOI: 10.1145/342009.335383. URL: <https://doi.org/10.1145/342009.335383>.
- [6] Rakesh Agrawal et al. “Automatic subspace clustering of high dimensional data for data mining applications”. In: *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*. 1998, 94–105.
- [7] Mihael Ankerst et al. “OPTICS: Ordering points to identify the clustering structure”. In: *ACM Sigmod record* 28.2 (1999), pp. 49–60.
- [8] Matej Antol et al. “Learned Metric Index — Proposition of learned indexing for unstructured data”. In: *Information Systems* 100 (2021), 101–117. ISSN: 0306-4379. DOI: <https://doi.org/10.1016/j.is.2021.101774>. URL: <https://www.sciencedirect.com/science/article/pii/S0306437921000326>.
- [9] David Arthur and Sergei Vassilvitskii. “K-Means++: The Advantages of Careful Seeding”. In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA ’07. New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035. ISBN: 9780898716245.

- [10] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. "Correlation clustering". In: *Machine learning* 56.1 (2004), pp. 89–113.
- [11] Jayanta Basak. "Learning Hough Transform: A Neural Network Model". In: *Neural Computation* 13.3 (2001), pp. 651–676.
- [12] Christian Bauckhage. *k-Means Clustering Is Matrix Factorization*. 2015. DOI: 10.48550/ARXIV.1512.07548. URL: <https://arxiv.org/abs/1512.07548>.
- [13] Anna Beer, Daniyal Kazempour, and Thomas Seidl. "Rock - Let the points roam to their clusters themselves". In: *Advances in Database Technology - 22nd International Conference on Extending Database Technology, EDBT 2019, Lisbon, Portugal, March 26-29, 2019*. Ed. by Melanie Herschel et al. OpenProceedings.org, 2019, pp. 630–633. DOI: 10.5441/002/edbt.2019.73. URL: <https://doi.org/10.5441/002/edbt.2019.73>.
- [14] Richard E. Bellman. *A Guided Tour*. Princeton: Princeton University Press, 2015. ISBN: 9781400874668. DOI: doi : 10.1515/9781400874668. URL: <https://doi.org/10.1515/9781400874668>.
- [15] Christian Böhm et al. "Computing Clusters of Correlation Connected Objects". In: *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*. SIGMOD '04. Paris, France: Association for Computing Machinery, 2004, pp. 455–466. ISBN: 1581138598. DOI: 10.1145/1007568.1007620. URL: <https://doi.org/10.1145/1007568.1007620>.
- [16] Piotr Bojanowski et al. "Enriching Word Vectors with Subword Information". In: *CoRR abs/1607.04606* (2016). arXiv: 1607.04606. URL: <http://arxiv.org/abs/1607.04606>.
- [17] Leon Bottou and Yoshua Bengio. "Convergence properties of the k-means algorithms". In: *Advances in neural information processing systems* 7 (1994).
- [18] P.S. Bradley and O.L. Mangasarian. "k-Plane Clustering". In: *Journal of Global Optimization* 16 (Jan. 2000), pp. 23–32. DOI: 10.1023/A:10083-24-62-55-22.
- [19] Markus M. Breunig et al. "LOF: Identifying Density-Based Local Outliers". In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*. SIGMOD '00. Dallas, Texas, USA: Association for Computing Machinery, 2000, pp. 93–104. ISBN: 1581132174. DOI: 10.1145/342009.335388. URL: <https://doi.org/10.1145/342009.335388>.
- [20] Julian Busch et al. "Implicit Hough Transform Neural Networks for Subspace Clustering". In: *2021 International Conference on Data Mining, ICDM 2021 - Workshops, Auckland, New Zealand, December 7-10, 2021*. IEEE, 2021, pp. 441–448. DOI: 10.1109/ICDMW53433.2021.00060. URL: <https://doi.org/10.1109/IC%5C-DM%5C-W53%5C-43%5C-3.2021.00060>.

- [21] Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. "Density-Based Clustering Based on Hierarchical Density Estimates". In: *Advances in Knowledge Discovery and Data Mining*. Ed. by Jian Pei et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 160–172. ISBN: 978-3-642-37456-2.
- [22] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning*. The MIT Press, Sept. 2006. ISBN: 9780262255899. DOI: 10.7551/mitpress/9780262033589.001.0001. URL: <https://doi.org/10.7551/mitpress/9780262033589.001.0001>.
- [23] Chun-Hung Cheng, Ada Waichee Fu, and Yi Zhang. "Entropy-Based Subspace Clustering for Mining Numerical Data". In: *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. KDD '99. San Diego, California, USA: Association for Computing Machinery, 1999, pp. 84–93. ISBN: 1581131437. DOI: 10.1145/312129.312199. URL: <https://doi.org/10.1145/312129.312199>.
- [24] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR abs/1810.04805* (2018). arXiv: 1810.04805. URL: <http://arxiv.org/abs/1810.04805>.
- [25] Pedro Domingos. "A Few Useful Things to Know about Machine Learning". In: *Commun. ACM* 55.10 (Oct. 2012), pp. 78–87. ISSN: 0001-0782. DOI: 10.1145/2347736.2347755. URL: <https://doi.org/10.1145/2347736.2347755>.
- [26] Ehsan Elhamifar and René Vidal. "Sparse Subspace Clustering: Algorithm, Theory, and Applications". In: *CoRR abs/1203.1005* (2012). arXiv: 1203.1005. URL: <http://arxiv.org/abs/1203.1005>.
- [27] Ehsan Elhamifar and Rene Vidal. "Sparse subspace clustering". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, 2790–2797. DOI: 10.1109/CVPR.2009.5206547.
- [28] Martin Ester et al. "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise". In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. KDD'96. Portland, Oregon: AAAI Press, 1996, pp. 226–231.
- [29] Karl Pearson F.R.S. "LIII. On lines and planes of closest fit to systems of points in space". In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572. DOI: 10.1080/14786440109462720.
- [30] Alhussein Fawzi et al. "Discovering faster matrix multiplication algorithms with reinforcement learning". In: *Nature* 610 (Oct. 2022), pp. 47–53. DOI: 10.1038/s41586-022-05172-4.

- [31] Paolo Ferragina, Fabrizio Lillo, and Giorgio Vinciguerra. “Why Are Learned Indexes So Effective?” In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, 3123–3132. URL: <https://proceedings.mlr.press/v119/ferragina20a.html>.
- [32] Aristides Gionis, Piotr Indyk, and Rajeev Motwani. “Similarity Search in High Dimensions via Hashing”. In: *Proceedings of the 25th International Conference on Very Large Data Bases*. VLDB ’99. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1999, pp. 518–529. ISBN: 1558606157.
- [33] Sanjay Goil, Harsha Nagesh, and Alok Choudhary. “Mafia: Efficient and scalable subspace clustering for very large data sets”. In: *Proc. 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Citeseer*. Citeseer, 1999, pp. 443–452.
- [34] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. <http://www.deeplearningbook.org>. MIT Press, 2016.
- [35] Antonin Guttman. “R-Trees: A Dynamic Index Structure for Spatial Searching”. In: *Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’84. Boston, Massachusetts: Association for Computing Machinery, 1984, pp. 47–57. ISBN: 0897911288. DOI: 10.1145/602259.602266. URL: <https://doi.org/10.1145/602259.602266>.
- [36] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. 2nd ed. Springer, 2009. URL: <http://biostat.stanford.edu/~tibshirani/ElmStatLearn/>, [http://biostat.stanford.edu/~tibshirani/Hastie2001/weatherwax\\_epstein\\_hastie\\_solutions\\_manual.pdf](http://biostat.stanford.edu/~tibshirani/Hastie2001/weatherwax_epstein_hastie_solutions_manual.pdf).
- [37] Tony Hey et al. *The Fourth Paradigm: Data-Intensive Scientific Discovery*. Microsoft Research, Oct. 2009. ISBN: 978-0-9825442-0-4. URL: <https://www.microsoft.com/en-us/research/publication/fourth-paradigm-data-intensive-scientific-discovery/>.
- [38] Maximilian Hünemörder, Peer Kröger, and Matthias Renz. “Towards a Learned Index Structure for Approximate Nearest Neighbor Search Query Processing”. In: *Similarity Search and Applications - 14th International Conference, SISAP 2021, Dortmund, Germany, September 29 - October 1, 2021, Proceedings*. Ed. by Nora Reyes et al. Vol. 13058. Lecture Notes in Computer Science. Springer, 2021, pp. 95–103. DOI: 10.1007/978-3-030-89657-7\_8. URL: [https://doi.org/10.1007/978-3-030-89657-7\\_8](https://doi.org/10.1007/978-3-030-89657-7_8).
- [39] Maximilian Hünemörder et al. “SePass: Semantic Password Guessing Using k-Nn Similarity Search In Word Embeddings”. In: *Advanced Data*

- Mining and Applications: 18th International Conference, ADMA 2022, Brisbane, QLD, Australia, November 28–30, 2022, Proceedings, Part II.* Brisbane, QLD, Australia: Springer-Verlag, 2022, pp. 28–42. ISBN: 978-3-031-22136-1. DOI: 10.1007/978-3-031-22137-8\_3. URL: [https://doi.org/10.1007/978-3-031-22137-8\\_3](https://doi.org/10.1007/978-3-031-22137-8_3).
- [40] Maximilian Hünemörder et al. “Stirring the Pot - Teaching Reinforcement Learning Agents a “Push-Your-Luck” board game”. In: *2022 IEEE Conference on Games (CoG)*. 2022, pp. 600–603. DOI: 10.1109/CoG51982.2022.9893657.
- [41] Maximilian Archimedes Xaver Hünemörder et al. “CODEC - Detecting Linear Correlations in Dense Clusters using coMAD-based PCA”. In: *Proceedings of the Conference on “Lernen, Wissen, Daten, Analysen”, Berlin, Germany, September 30 - October 2, 2019*. Ed. by Robert Jäschke and Matthias Weidlich. Vol. 2454. CEUR Workshop Proceedings. CEUR-WS.org, 2019, pp. 111–114. URL: [http://ceur-ws.org/Vol-2454/paper\\_74.pdf](http://ceur-ws.org/Vol-2454/paper_74.pdf).
- [42] Maximilian Archimedes Xaver Hünemörder et al. “SIDEKICK: Linear Correlation Clustering with Supervised Background Knowledge”. In: *Similarity Search and Applications - 12th International Conference, SISAP 2019, Newark, NJ, USA, October 2-4, 2019, Proceedings*. Ed. by Giuseppe Amato et al. Vol. 11807. Lecture Notes in Computer Science. Springer, 2019, pp. 221–230. DOI: 10.1007/978-3-030-32047-8\_20. URL: [https://doi.org/10.1007/978-3-030-32047-8\\_20](https://doi.org/10.1007/978-3-030-32047-8_20).
- [43] Armand Joulin et al. “Bag of Tricks for Efficient Text Classification”. In: *CoRR abs/1607.01759* (2016). arXiv: 1607.01759. URL: <http://arxiv.org/abs/1607.01759>.
- [44] Armand Joulin et al. “FastText.zip: Compressing text classification models”. In: *CoRR abs/1612.03651* (2016). arXiv: 1612.03651. URL: <http://arxiv.org/abs/1612.03651>.
- [45] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. 1st. USA: Prentice Hall PTR, 2000. ISBN: 0130950696.
- [46] Karin Kailing, Hans-Peter Kriegel, and Peer Kröger. “Density-connected subspace clustering for high-dimensional data”. In: *Proceedings of the 2004 SIAM international conference on data mining*. SIAM. 2004, pp. 246–256.
- [47] Leonard Kaufman and Peter Rousseeuw. *Finding Groups in Data: An Introduction To Cluster Analysis*. Jan. 1990. ISBN: 0-471-87876-6. DOI: 10.2307/2532178.

- [48] Daniyal Kazempour, Maximilian Archimedes Xaver Hünemörder, and Thomas Seidl. "On coMADs and Principal Component Analysis". In: *Similarity Search and Applications - 12th International Conference, SISAP 2019, Newark, NJ, USA, October 2-4, 2019, Proceedings*. Ed. by Giuseppe Amato et al. Vol. 11807. Lecture Notes in Computer Science. Springer, 2019, pp. 273–280. DOI: 10.1007/978-3-030-32047-8\_24. URL: [https://doi.org/10.1007/978-3-030-32047-8\\_24](https://doi.org/10.1007/978-3-030-32047-8_24).
- [49] Andreas Kipf et al. "SOSD: A benchmark for learned indexes". In: *arXiv preprint arXiv:1911.13014* (2019).
- [50] Tim Kraska. "Towards instance-optimized data systems". In: *Proceedings of the VLDB Endowment* 14.12 (2021).
- [51] Tim Kraska et al. "The Case for Learned Index Structures". In: *Proceedings of the 2018 International Conference on Management of Data*. SIGMOD '18. Houston, TX, USA: Association for Computing Machinery, 2018, pp. 489–504. ISBN: 9781450347037. DOI: 10.1145/3183713.3196909. URL: <https://doi.org/10.1145/3183713.3196909>.
- [52] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. "Clustering High-Dimensional Data: A Survey on Subspace Clustering, Pattern-Based Clustering, and Correlation Clustering". In: *ACM Trans. Knowl. Discov. Data* 3.1 (Mar. 2009). ISSN: 1556-4681. DOI: 10.1145/1497577.1497578. URL: <https://doi.org/10.1145/1497577.1497578>.
- [53] Hans-Peter Kriegel, Peer Kröger, and Arthur Zimek. "Subspace clustering". In: *WIREs Data Mining and Knowledge Discovery* 2.4 (2012), 3-51–364. DOI: <https://doi.org/10.1002/widm.1057>. eprint: <https://wires.onlinelibrary.wiley.com/doi/pdf/10.1002/widm.1057>. URL: <https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1057>.
- [54] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. "Isolation Forest". In: *2008 Eighth IEEE International Conference on Data Mining*. 2008, pp. 413–422. DOI: 10.1109/ICDM.2008.17.
- [55] Andreas Lohrer et al. "OAB - An Open Anomaly Benchmark Framework for Unsupervised and Semisupervised Anomaly Detection on Image and Tabular Data Sets". In: *2021 International Conference on Data Mining Workshops (ICDMW)*. 2021, pp. 991–1000. DOI: 10.1109/ICDMW-2021-00129.
- [56] Laurens van der Maaten and Geoffrey Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9.86 (2008), pp. 2579–2605. URL: <http://jmlr.org/papers/v9/vandermaaten08a.html>.
- [57] Prasanta Chandra Mahalanobis. "On the generalized distance in statistics". In: *Proceedings of the National Institute of Sciences (Calcutta)* 2 (1936), pp. 49–55.
- [58] Ryan Marcus et al. "Benchmarking learned indexes". In: *arXiv preprint arXiv:2006.12804* (2020).

- [59] Tomas Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).
- [60] Mehdi Mirza and Simon Osindero. "Conditional Generative Adversarial Nets". In: *CoRR abs/1411.1784* (2014). arXiv: 1411. 1784. URL: <http://arxiv.org/abs/1411.1784>.
- [61] Michael Mitzenmacher. "A model for learned bloom filters and optimizing by sandwiching". In: *Advances in Neural Information Processing Systems* 31 (2018).
- [62] Vikram Nathan et al. "Learning Multi-Dimensional Indexes". In: *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*. SIGMOD '20. Portland, OR, USA: Association for Computing Machinery, 2020, pp. 985–1000. ISBN: 9781450367356. DOI: 10.1145/3318464.3380579. URL: <https://doi.org/10.1145/3318464.3380579>.
- [63] R Ng and Jiawei Han. "Efficient and effective clustering method for spatial data mining". In: *Proc. of the 20th VLDB Conference*, pp. 144–155.
- [64] Raymond T. Ng and Jiawei Han. "CLARANS: A method for clustering objects for spatial data mining". In: *IEEE transactions on knowledge and data engineering* 14.5 (2002), pp. 1003–1016.
- [65] Charles Egerton Osgood. *The Measurement of Meaning*. London: Urbana, University of Illinois Press, 1957.
- [66] Rajeev Raman, Venkatesh Raman, and Srinivasa Rao Satti. "Succinct indexable dictionaries with applications to encoding  $k$ -ary trees, prefix sums and multisets". In: *ACM Transactions on Algorithms* 3.4 (Nov. 2007), p. 43. DOI: 10.1145/1290672.1290680. URL: <https://doi.org/10.1145/1290672.1290680>.
- [67] LKPJ Rduseeun and P Kaufman. "Clustering by means of medoids". In: *Proceedings of the statistical data analysis based on the L1 norm conference, neuchatel, switzerland*. Vol. 31. 1987.
- [68] Alex Rodriguez and Alessandro Laio. "Clustering by fast search and find of density peaks". In: *science* 344.6191 (2014), pp. 1492–1496.
- [69] Ibrahim Sabek et al. *When Are Learned Models Better Than Hash Functions?* 2021. DOI: 10.48550/ARXIV.2107.01464. URL: <https://arxiv.org/abs/2107.01464>.
- [70] Hanan Samet. *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005. ISBN: 0123694469.
- [71] Bernhard Schölkopf et al. "Support Vector Method for Novelty Detection". In: *Advances in Neural Information Processing Systems*. Ed. by S. Solla, T. Leen, and K. Müller. Vol. 12. MIT Press, 1999. URL: <https://proceedings.neurips.cc/paper/1999/file/875c-25fb777f25776ffa9076e44fcfd776-Paper.pdf>.

- [72] Erich Schubert and Peter J. Rousseeuw. “Fast and eager k-medoids clustering: O(k) runtime improvement of the PAM, CLARA, and CLARANS algorithms”. In: *Information Systems* 101 (2021), p. 101804. ISSN: 0306-4379. DOI: <https://doi.org/10.1016/j.is.2021.101804>. URL: <https://www.sciencedirect.com/science/article/pii/S0306437921000557>.
- [73] Erich Schubert and Peter J. Rousseeuw. “Faster K-Medoids Clustering: Improving the PAM, CLARA, and CLARANS Algorithms”. In: *Similarity Search and Applications: 12th International Conference, SISAP 2019, Newark, NJ, USA, October 2–4, 2019, Proceedings*. Newark, NJ, USA: Springer-Verlag, 2019, pp. 171–187. ISBN: 978-3-030-32046-1. DOI: 10.1007/978-3-030-32047-8\_16. URL: [https://doi.org/10.1007/978-3-030-32047-8\\_16](https://doi.org/10.1007/978-3-030-32047-8_16).
- [74] Erich Schubert et al. “DBSCAN revisited, revisited: why and how you should (still) use DBSCAN”. In: *ACM Transactions on Database Systems (TODS)* 42.3 (2017), pp. 1–21.
- [75] David Sculley. “Web-scale k-means clustering”. In: *Proceedings of the 19th international conference on World wide web*. 2010, pp. 1177–1178.
- [76] Jonathon Shlens. “A Tutorial on Principal Component Analysis”. In: *CoRR abs/1404.1100* (2014). arXiv: 1404.1100. URL: <http://arxiv.org/abs/1404.1100>.
- [77] Terézia Slanináková et al. “Data-Driven Learned Metric Index: An Unsupervised Approach”. In: *Similarity Search and Applications: 14th International Conference, SISAP 2021, Dortmund, Germany, September 29 – October 1, 2021, Proceedings*. Dortmund, Germany: Springer-Verlag, 2021, pp. 81–94. ISBN: 978-3-030-89656-0. DOI: 10.1007/978-3-030-89657-7\_7. URL: [https://doi.org/10.1007/978-3-030-89657-7\\_7](https://doi.org/10.1007/978-3-030-89657-7_7).
- [78] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [79] Joshua B Tenenbaum, Vin de Silva, and John C Langford. “A global geometric framework for nonlinear dimensionality reduction”. In: *science* 290.5500 (2000), pp. 2319–2323.
- [80] Theresa Ullmann et al. “Over-optimistic evaluation and reporting of novel cluster algorithms: an illustrative study”. In: *Advances in Data Analysis and Classification* (2022), pp. 1–28. DOI: 10.1007/s11634-022-00496-5.
- [81] Kapil Vaidya et al. “Partitioned Learned Bloom Filter”. In: *CoRR abs/2006.03176* (2020). arXiv: 2006.03176. URL: <https://arxiv.org/abs/2006.03176>.
- [82] Andrea Vattani. “The hardness of k-means clustering in the plane”. In: 2010.
- [83] Sheng Zhou et al. *A Comprehensive Survey on Deep Clustering: Taxonomy, Challenges, and Future Directions*. 2022. DOI: 10.48550/ARXIV.2206.07579. URL: <https://arxiv.org/abs/2206.07579>.