TECHNISCHE UNIVERSITÄT WIEN
Vienna University of Technology

Diplomarbeit

Vergleich von Algorithmen zur Ausreißererkennung in höherdimensionalen Räumen

Ausgeführt am Institut für
Analysis und Scientific Computing
der Technischen Universität Wien

unter der Anleitung von

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Felix Breitenecker
und
Dipl.-Ing. Dr.techn. Nikolas Popper

durch

Klaus Köpplinger
Kirchengasse 21
2460 Bruck an der Leitha

Bruck an der Leitha, 16. Mai 2017

Hiermit versichere ich, die vorliegende Diplomarbeit selbstständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubter Hilfe bedient zu haben, dass ich dieses Diplomarbeitsthema bisher weder im Innoch im Ausland in irgendeiner Form als Prüfungsarbeit vorgelegt habe und dass diese Arbeit mit der vom Begutachter beurteilten Arbeit vollständig übereinstimmt.

Bruck an der Leitha, 16. Mai 2017

Danke meinen Eltern für ihre Unterstützung.

Zusammenfassung

Die vorliegende Diplomarbeit beschreibt die Probleme, die bei der Ausreißererkennung in höherdimensionalen Räumen vorkommen, und stellt Algorithmen vor, die in der Lage sind, Ausreißer in solchen Räumen zu finden. Die ersten Kapitel beinhalten eine allgemeine Einführung in die Thematik, führen den Begriff Ausreißer ein und gehen auf den Unterschied zu Noise und Novelty ein. Danach wird ein kurzer Überblick über Algorithmen zur Ausreißererkennung gegeben. Im Weiteren wird der *Fluch der Dimensionalität* dargestellt. Dieser Begriff bezeichnet die geringer werdende Aussagekraft von Metriken mit wachsender Dimension, verursacht durch die abnehmende Dichte des Datenraums. Die notwendigen Sätze zu Aussagen zum Verhalten von Metriken in höherdimensionalen Räumen werden bewiesen. Im Anschluss werden die Algorithmen *Ausreißererkennung in Unterräumen*, *Ausreißererkennung mit Hilbert-Kurven* und *Angle Based Outlier Detection* sowie ihre Eigenschaften detailliert erklärt. Diese Algorithmen wurden speziell für die Ausreißererkennung in höherdimensionalen Räumen entwickelt. Sie basieren allerdings auf jeweils unterschiedlichen Ideen um Ausreißer zu finden, was die Frage aufwirft, ob ein Algorithmus den anderen überlegen ist. In den letzten beiden Kapiteln versuchen wir, diese Frage zu beantworten, indem wir mit diesen Algorithmen Ausreißer in Datenräumen suchen und die Ergebnisse dieser Suche analysieren.

Abstract

This master thesis describes the problems that occur at outlier detection in high-dimensional spaces and presents algorithms which are able to find outliers in such spaces. The first chapters lead into the topic, introduce the term outlier and describe the distinction to noise and novelty. Next the *curse of dimensionality* is presented. This concept denotes the weaker significance of a metric if the dimension increases, caused by the low density of high-dimensional data spaces. The required theorems covering the behavior of metrics in high-dimensional spaces are proved. Subsequently the algorithms *Outlier Detection in Subspaces*, *Outlier Detection using Hilbert curves* and *Angle Based Outlier Detection* and their characteristics are explained. These algorithms were designed specifically for outlier detection in high-dimensional spaces. Each of them is based on its own idea to find outliers, raising the question: Is one algorithm superior to the other algorithms? In the final chapters we try to answer this question by searching for outliers in data spaces with these algorithms and analyze the results of these searches.

Inhaltsverzeichnis

1 Einführung **1**

1.1 Ausreißererkennung als Teil von Knowledge Discovery in Databases und der Statistik . . . 1
1.2 Anwendungen . . . 2
1.3 Probleme bei Ausreißererkennung . . . 2
1.4 Ausreißererkennung in höherdimensionalen Räumen . . . 3

2 Definition des Begriffs Ausreißer **6**

2.1 Allgemeine Definitionen . . . 6
2.2 Verschiedene Definitionen . . . 6
2.3 Arten von Ausreißern . . . 7
2.4 Unterschied zwischen Ausreißern, Noise und Novelty . . . 9
2.4.1 Noise . . . 9
2.4.2 Novelty . . . 10

3 Verfahren zur Ausreißererkennung **12**

4 Fluch der Dimensionalität **14**

4.1 Grundlagen aus der Wahrscheinlichkeitstheorie . . . 14
4.2 Der Satz von Beyer et al. . . . 16
4.3 Der Einfluss der Norm . . . 18
4.3.1 Quasinorm und Fractional Distance Metric . . . 23

5 Ausreißererkennung in Unterräumen **26**

5.1 Der Sparsity Coefficient $S(\mathcal{C})$ . . . 30
5.2 Evolutionärer Algorithmus zur Ausreißererkennung . . . 31
5.3 Wahl der Parameter κ und ϕ . . . 32
5.4 Laufzeitanalyse . . . 35

6 Ausreißererkennung mit Hilbert-Kurven **37**

6.1 Hilbert-Kurven . . . 37
6.2 Benötigte Definitionen und Formeln . . . 39
6.3 Beschreibung des Algorithums . . . 42
6.4 Qualität der Näherungslösung . . . 45
6.5 Laufzeitanalyse . . . 52

7 Angle-Based Outlier Detection **53**
7.1 Der Angle-Based Outlier Factor 53
7.2 FastABOD 54
7.3 LB-ABOD 56
7.4 Monte-Carlo-Algorithmus mit Laufzeit $N \log^2(N)$ 60

8 Implementierung und Vergleich der Algorithmen **70**
8.1 Arrhythmia-Datensatz 70
8.1.1 Laufzeit 70
8.1.2 Qualität der Resultate 74
8.2 Datensatz Bag-of-words von Stack Exchange 78
8.3 Grammatical-Facial-Expressions-Datensatz 83

9 Resümee **88**

A Formelzeichen, Symbole und Abkürzungen **90**

Abbildungsverzeichnis

2.1 Temperaturverlauf Nottingham Castle 1920 und 1921. 8
2.2 Collective Outliern 9
2.3 Ausreißer und Noise 11

4.1 Differenz zwischen nächstem und entferntestem Datum bei gleichverteilten Daten, abhängig von Dimension und p-Norm 22
4.2 Einheitskreise von drei Quasinormen mit unterschiedlichen Parametern und der Summennorm 24

5.1 `iris`-Dataset ohne Attribut *Breite Kelchblatt* 27
5.2 `iris`-Dataset, alle Attribute 28
5.3 Konstruktion der Quader. 29

6.1 Hilbert-Kurve zweiter Ordnung 38
6.2 Die ersten drei Folgenglieder einer Hilbert-Kurven-Konstruktion 39
6.3 Drei Teilwürfel mit den Ordnungen 1 (A), 2 (B) und 3 (C). 41
6.4 Teilwürfel bei der Berechnung von MaxCube und MinCube 41
6.5 Berechnung von MaxDist und MinDist 41
6.6 Berechnung von BoxRadius 41
6.7 Anwendung von Lemma 6.11 43
6.8 Daten innerhalb des grauen Gebiets liegen c-zentral. 49
6.9 Abschätzung des Abstands zu einem c-zentralem Datum x. 49

7.1 Winkel zu anderen Daten bei einem normalen Datum 54
7.2 Winkel zu anderen Daten bei einem Ausreißer 54
7.3 ABOF von Beispieldaten unter Verwendung der euklidischen Metrik. . . . 55
7.4 Winkel und Zufallsvektor r 60

8.1 Laufzeit der Algorithmen in Abhängigkeit von der Anzahl der Daten, Datensatz Arrhythmia 72
8.2 Laufzeit der Algorithmen in Abhängigkeit von der Dimension, Datensatz Arrhythmia 73
8.3 Laufzeit Evolutionary OD in Sekunden in Abhängigkeit der Parameter m, ϕ und κ. Datensatz Arrhythmia 74
8.4 Laufzeit Hilbert OD in Sekunden in Abhängigkeit der Parameter k und h. Datensatz Arrhythmia 74
8.5 Recallwerte der Algorithmen, Datensatz Arrhythmia. 76

8.6 Ähnlichkeit der Algorithmen, Datensatz Arrhythmia. 77
8.7 Ähnlichkeit zw. Evolutionary OD und den anderen Algorithmen, Datensatz Arrhythmia. 78
8.8 Ähnlichkeit der Algorithmen, Datensatz bag-of-words. 80
8.9 Recallwerte der Algorithmen, Datensatz bag-of-words. 81
8.10 Anzahl der Terme in Abhängigkeit von der Bewertung als Ausreißer. . . . 82
8.11 Ähnlichkeit zw. Evolutionary OD und den anderen Algorithmen, Datensatz bag-of-words. 83
8.12 Recallwerte der Algorithmen, Datensatz GFE. 85
8.13 Ähnlichkeit der Algorithmen, Datensatz GFE. 87

Tabellenverzeichnis

5.1 Laufzeit der einzelnen Teile des Algorithmus Evolutionary Outlier Detection in Subspaces 36

6.1 verwendete Variablen der Algorithmen Hilbert Outlier Detection, Scan und InnerScan 44
6.2 Eigenschaften der Klasse für die Daten 44

7.1 Laufzeit der einzelner Zeilen von Algorithmus Near-linear Angle-Based Outlier Detection 67

8.1 Fix gesetzte Parameter, Laufzeitanalyse, Datensatz Arrhythmia 71
8.2 Laufzeit der Algorithmen in Abhängigkeit von der Anzahl der Daten, Datensatz Arrhythmia 71
8.3 Laufzeit der Algorithmen in Sekunden in Abhängigkeit von der Dimension, Datensatz Arrhythmia. 72
8.4 Laufzeit der Algorithmen in Sekunden in Abhängigkeit der Parameter, Datensatz Arrhythmia 73
8.5 Fix gesetzte Parameter, Analyse der Qualität, Datensatz Arrhythmia . . . 74
8.6 Position der Ausreißer, Datensatz Arrhythmia. 75
8.7 Ergebnisse Evolutionary OD, Datensatz Arrhythmia. 77
8.8 Fix gesetzte Parameter, Datensatz bag-of-words. 79
8.9 Position der Ausreißer, Datensatz bag-of-words 81
8.10 Fix gesetzte Parameter, Datensatz GFE. 84
8.11 Laufzeit der Algorithmen, Datensatz GFE 86
8.12 Position der Ausreißer, Datensatz GFE. 86

1 Einführung

1.1 Ausreißererkennung als Teil von Knowledge Discovery in Databases und der Statistik

Wenn sich ein System ungewöhnlich verhält, können Ausreißer entstehen. Ausreißer beinhalten daher Information über ungewöhnliche Systemzustände. In manchen Anwendungen kann diese Information wichtiger oder interessanter sein als die Information, die sich in den normalen Daten befindet. „One person's noise is another person's signal" [Kno98, S. 392]. Ausreißer können z. B. Hinweise auf ein Sicherheitsproblem liefern, auf das ein Benutzer des Systems reagieren muss. In solchen Anwendungen ist es sinnvoll, sich näher mit Ausreißern zu beschäftigen, z. B. sie zu suchen, ihre Ursache zu finden und mögliche Reaktionen daraus abzuleiten.

Ausreißererkennung beschäftigt sich mit Verfahren, die Ausreißer finden, und ist im Rahmen dieser Arbeit ein Teil des Gebiets *Knowledge Discovery in Databases (KDD)*. Bei KDD versucht man aus vorhandenen, meist großen, Datenbeständen neue, bisher unbekannte Informationen zu bekommen. [Han12, S. 6–8] zerlegen KDD in folgende Schritte:

1. **Data-Cleaning** Entfernen von inkonsistenten, falschen oder redundanten Daten und von Noise.
2. **Data-Integration** Kombinieren und Zusammenführen der vorhandenen Datenquellen (z. B. Verknüpfen von Daten aus mehreren Datenbanken).
3. **Data-Selection** Auswählen der Daten, die für die Analyse bedeutend sind.
4. **Data-Transformation** Umwandeln der Daten in eine für die Analyse passende Form und Struktur.
5. **Data-Mining** Gewinnen von Informationen aus den Daten (der zentrale Teil von KDD).
6. **Pattern-Evaluation** Auswerten und Interpretieren der gewonnenen Information, Herausfinden der interessanten Ergebnisse des vorigen Schritts.
7. **Knowledge-Presentation** Aufbereiten der gewonnenen Information in einer für das Zielpublikum passenden Form.

Der Begriff Data-Mining wird oft synonym zum Begriff Knowledge Discovery in Databases verwendet, ist aber eigentlich nur ein Schritt von KDD. Falls man Ausreißer sucht, gehört diese Aufgabe zum Schritt Data-Mining. Andere Untersuchungen, die man

beim Data-Mining durchführen kann, sind u. a.: Feststellen von Zusammenhängen und Korrelationen, Klassifikation, Regressionsanalyse, Clusteranalyse. Welche dieser Untersuchungen man durchführt, hängt von der jeweiligen Anwendung ab.

Viele Algorithmen, die Ausreißer erkennen, können auch Noise erkennen (Schritt Data-Cleaning). Ausreißer und Noise sind allerdings verschiedene Dinge, wir werden die Unterschiede in Kapitel 2.4 betrachten.

In der Statistik spielen Ausreißer ebenfalls eine Rolle. Es existieren diverse Tests zur Ausreißererkennung. Für KDD sind die meisten dieser Tests aber nur von geringem Nutzen, hauptsächlich aufgrund von zwei Ursachen: Erstens können viele Tests nur mit univariaten Daten umgehen, bei KDD sind die Daten aber in der Regel multivariat. Zweitens gehen diese Tests von einer bestimmten Häufigkeitsverteilung der Daten aus. Bei KDD ist die Häufigkeitsverteilung der Daten allerdings oft unbekannt. Man kann dann nicht sicher sein, ob der Test für die gegebenen Daten geeignet ist.

1.2 Anwendungen

Für Ausreißererkennung gibt es eine Vielzahl von Anwendungen. Einige Beispiele:

- Banken und Versicherungen verwenden Ausreißererkennung zur *Betrugsbekämpfung*. Sie suchen in Buchungen oder Schadensmeldungen nach ungewöhnlichen Daten, um Kreditkartenbetrug, Insiderhandel oder Versicherungsbetrug zu erkennen. Für Telefongesellschaften können Ausreißer in den Verbindungsdaten ein Hinweis auf Missbrauch (z. B. Lockanrufe) sein.
- Administratoren können mit Ausreißererkennung *Angriffe auf Computer und Computernetze* entdecken, indem sie nach auffälligen Mustern bei Systemaufrufen, Verbindungsanfragen oder in den übertragenen Daten suchen.
- Unübliche *Sensordaten* von Maschinen oder Anlagen können auf einen Schaden, eine Fehlfunktion oder bevorstehendes Versagen hinweisen.
- In der *Medizin* verwendet man Ausreißererkennung, um Symptome zu finden: Dazu untersucht man etwa Bilder von Tomografen oder Zeitreihen von Elektrokardiogrammen (EKG) oder Elektroenzephalogrammen (EEG).
- In der *Bildverarbeitung*: Man sucht nach Merkmalen, die bei Satellitenbildern auf außergewöhnliche Wetter- und Klimaphänomene oder bei Überwachungssystemen auf Einbrüche hindeuten.
- *Fehlerhafte Eingaben* von Computeranwendern oder *Messfehler* von Sensoren können durch Ausreißererkennung gefunden werden.

1.3 Probleme bei Ausreißererkennung

Ein guter Algorithmus zur Ausreißererkennung fügt möglichst viele Daten richtig der Menge der normalen Daten oder der Menge der Ausreißer hinzu. Die Aufgabe für eine

bestimmte Anwendung einen solchen Algorithmus zu finden gilt als schwierig.

Wie wir im Kapitel 2.2 sehen werden, gibt es in der Literatur keine eindeutige, exakte Definition des Begriffs Ausreißer. Die Grenze zwischen Ausreißer und normalem Datum ist oft unscharf. Um eine solche Grenze zu ziehen (indem man z. B. den Parametern eines Algorithmus brauchbare Werte zuweist), benötigt man oft genaue Informationen über die Daten. Diese Informationen sind nicht immer vorhanden. Die Grenze wird dann möglicherweise schlecht gesetzt und der Algorithmus klassifiziert Daten falsch.

> [...] fundamental considerations when selecting an appropriate methodology for an outlier detection system are: [...] selecting a suitable neighbourhood of interest for an outlier. The selection of the neighbourhood of interest is non-trivial. [Hod04, S. 7]

Folgende Umstände erschweren die Grenzziehung zwischen normalen Daten und Ausreißern:

- Bei vielen Anwendungen sind nicht alle Arten des normalen Verhaltens bekannt.
- Ausreißer landen manchmal zufällig in einer Gruppe normaler Daten und sind dann nicht zu erkennen (z. B. kann eine durch Kreditkartenbetrug verursachte Buchung ähnlich wie normale Buchungen sein).
- Im Laufe der Zeit können sich bei normalen Daten die Größenordnungen der Attribute ändern.
- Ausreißer sind manchmal schwierig von Noise zu unterscheiden (siehe Kapitel 2.4).

„The Data Model is Everything“ [sic] [Agg13, S. 7]. Das Zitat führt uns zu einem weiteren Problem: Alle Algorithmen gehen von der Annahme aus, dass die normalen Daten einem bestimmten Modell entsprechen. Es stellt sich die Frage: Ist diese Annahme richtig? Falls nicht, wird der Algorithmus schlechte Ergebnisse liefern. Damit man ein passendes Modell wählen kann, benötigt man ebenfalls Informationen über die Daten oder ausreichendes Expertenwissen.

Zusätzlich weist [Agg13] noch auf den Aspekt des Over- und Underfittings bei der Wahl eines passenden Modells hin.

1.4 Ausreißererkennung in höherdimensionalen Räumen

Als höherdimensional gilt ein Raum im Rahmen der Ausreißererkennung ab einer Dimension von ungefähr 50. Die meisten Algorithmen zur Ausreißererkennung sind für solche Dimensionen nicht geeignet. Zwei Beispiele: Verfahren aus der Statistik verwenden ein ungeeignetes (häufig univariates) Modell. Das Verfahren *Convex Peeling* sucht Ausreißer mithilfe der konvexen Hülle der Daten. Diese kann man aber für Dimensionen größer als drei nicht mehr schnell berechnen.

Bei Algorithmen, die mit Abständen zwischen den Daten oder mit der Dichte der Daten im Raum arbeiten, wächst der Berechnungsaufwand linear mit der Dimension. Ein

weitaus größeres Problem für diese Algorithmen ist aber der Fluch der Dimensionalität (*curse of dimensionality*). „All these effects are far more fundamental problems then [sic] mere complexity issues . . . “ [Kri08]. Diese Feststellung gilt auch für Ausreißererkennung mit Neuralen Netzen: „The neural networks attempt to fit a surface over the data and there must be sufficient data density to discern the surface“ [Hod04, S. 23].

Mit dem Fluch der Dimensionalität werden wir uns ausführlich in Kapitel 4 auseinandersetzen. Der Kern des Problems liegt in der folgenden Aussage: Sei D ein Datenraum; dann gilt unter bestimmten Voraussetzungen

$$\frac{\max_{x \in D}(\|x\|) - \min_{x \in D}(\|x\|)}{\min_{x \in D}(\|x\|)} \rightarrow_p 0 \text{ für } \dim(D) \rightarrow \infty,$$

wobei $\rightarrow_p$ die Konvergenz in Wahrscheinlichkeit bezeichnet. Das Verhältnis zwischen der Differenz der Normen und der kleinsten Norm heißt *relativer Kontrast* (relative contrast).

Dass der relative Kontrast gegen 0 konvergiert, führt dazu, dass der Unterschied zwischen dem nächsten und dem weitesten Nachbarn relativ klein wird. Alle Daten scheinen gleich weit voneinander entfernt zu sein und die Dichte des Datenraums wird sehr gering. Ziel der Ausreißererkennung ist es Daten in Bereichen mit geringer Dichte zu finden. Je geringer die Dichte des gesamten Datenraums ist, desto schwieriger ist dieses Ziel zu erreichen.

Dieses Konvergenzverhalten trifft auf die meisten Metriken zu, insbesondere auch auf die oft verwendete euklidische Metrik und die Manhatten-Metrik. Die oben erwähnten abstands- und dichtebasierten Algorithmen verwenden üblicherweise direkt oder indirekt eine solche Metrik. Die Ergebnisse dieser Algorithmen werden schlechter, je größer die Dimension des Datenraums ist.

> In such a case, the nearest neighbor problem becomes ill defined [sic], since the contrast between the distances of different data points does not exist. In such cases, even the concept of proximity may not be meaningful from a qualitative perspective: a problem which is even more fundamental than the performance degradation of high dimensional algorithms. [Agg01a]

Eine andere Sichtweise auf den Fluch der Dimensionalität betrachtet die Anzahl der Konfigurationen: Wenn jede Koordinate eines n-dimensionalen Vektors $x = (x_1, \ldots, x_n)$ die Werte 1 bis ϕ annehmen kann, gibt es insgesamt ϕ^n Konfigurationen. Diese Anzahl wächst exponentiell mit n.

[Kri08] fügen noch hinzu, dass das Problem durch unbedeutende Attribute verschlimmert wird, die bei höherdimensionalen Daten häufig vorhanden sind. Hinneburg et al. werfen in [Hin00] die Frage auf, ob es sinnvoll ist alle Attribute bei der Abstandsberechnung gleich zu behandeln.

An Algorithmen zur Ausreißererkennung in höherdimensionalen Räumen stellen wir daher folgende Anforderungen:

- Der Algorithmus muss mit weit verstreuten Daten im Datenraum und mit Daten, deren Abstände ähnlich groß sind, umgehen können.

- Die Laufzeit darf für hohe Dimensionen nicht zu lang sein.
- Die Ergebnisse müssen eine Begründung liefern, aufgrund welcher Attribute ein Datum ein Ausreißer ist. Der Anwender muss wichtige Attribute von unwichtigen unterscheiden können.

Es gibt verschiedene Möglichkeiten und Ideen, um diese Ziele zu erreichen: [Agg01a] schlägt vor eine p-Norm

$$\|x\|_p := \left(\sum_{i=1}^{n} |x_i|^p \right)^{1/p}$$

mit $0 < p < 1$ zu verwenden (eine *Quasinorm*, vgl. Kapitel 4.3.1). Je kleiner p ist, desto weniger ist die durch die Norm induzierte Metrik anfällig für den Fluch der Dimensionalität (die Manhatten-Metrik ist deshalb auch weniger anfällig als die euklidische Metrik).

Eine andere Methode besteht darin, nicht im gesamten Datenraum nach Ausreißern zu suchen, sondern nur in Unterräumen. Hier ist das Problem zu lösen aus den 2^n Unterräumen eines n-dimensionalen Datenraums den geeignetsten Unterraum zu finden.

Eine dritte Möglichkeit ist es, ein Modell zu wählen, dass ohne Abstände auskommt. In Kapitel 7 wird der *ABOD-Algorithmus* (Angle-Based Outlier Detection) vorgestellt, der statt Abstände, Winkel verwendet.

2 Definition des Begriffs Ausreißer

2.1 Allgemeine Definitionen

Zunächst einige allgemeine Definitionen, die wir für die weiteren Kapitel benötigen.

Definition 2.1. Die Menge aller Daten nennen wir *Datenraum* und bezeichnen sie mit D. Der Datenraum ist ein Vektorraum über $\mathbb{R}$ mit Dimension n, die Anzahl der Daten bezeichnen wir mit N.

$$D \subseteq \mathbb{R}^n, \quad |D| = N.$$

Der Kleinbuchstabe $q \in D$ steht immer für ein Abfragedatum, von dem wir wissen möchten, ob es ein Ausreißer ist oder nicht.

Mit $\| \ \|$ bezeichnen wir wie üblich eine Norm, in der Regel verwenden wir eine p-Norm $\| \ \|_p$:

Definition 2.2.

$$\| \ \|_p : \ \mathbb{R}^n \to \mathbb{R}_+^0, \quad x \mapsto \sqrt[p]{\sum_{i=1}^{n} |x_i|^{\,p}}.$$

Weiters benötigen wir noch eine Metrik. Die Metrik d ist meistens eine Minkowski-Metrik, also von einer p-Norm abgeleitet.

Definition 2.3.

$$\mathrm{d} : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_+^0, \quad (x, y) \mapsto \|x - y\|,$$
$$\mathrm{d}_p : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_+^0, \quad (x, y) \mapsto \|x - y\|_p.$$

2.2 Verschiedene Definitionen

„Ungefähr“ ist klar, was ein Ausreißer ist. Es ist aber schwierig, den Begriff so zu definieren, dass man ein Datum immer eindeutig als Ausreißer oder nicht als Ausreißer bezeichnen kann. Im Folgenden werden wir einige Definitionen auflisten, auf die in der Literatur häufig verwiesen wird. Die erste Definition basiert auf [Gru69]:

Definition 2.4. Ein *Ausreißer* ist eine Beobachtung, die merklich von den anderen Beobachtungen der Stichprobe abweicht, in der sie auftritt.

Die beiden folgenden Definitionen basieren auf [Haw80] und [Bar94]:

Definition 2.5. Ein *Ausreißer* ist ein Datum, das so weit von den anderen Daten abweicht, dass der Verdacht aufkommt, dass dieses Datum durch einen anderen Mechanismus erzeugt wurde.

Definition 2.6. Ein *Ausreißer* ist ein Datum (oder eine Teilmenge von Daten) welche in Widerspruch zu den restlichen Daten zu stehen scheint.

Salopp formuliert ist ein Ausreißer also ein Datum das „irgendwie“ nicht zu den übrigen Daten „passt“. Diese Definition ist nicht exakt und auch nicht streng im mathematischen Sinn. Das deckt sich mit der Situation bei der praktischen Anwendung: Ob ein einzelnes Datum bereits ein Ausreißer ist, oder noch ein normales Datum, ist nicht immer eindeutig. Die Einordnung hängt u. a. vom Modell und den verwendeten Parametern ab. „It is often a subjective judgement, as to what constitutes a “sufficient” deviation for a point to be considered an outlier“ [Agg13, S. 3].

Von [Kno98] stammt folgende Definition:

Definition 2.7. Ein Datum q eines Datenraums D ist ein $\mathrm{DB}(\pi, d)$-*Ausreißer* (Distance-Based-Outlier), wenn zumindest bei einem Anteil π der Daten in D der Abstand $\mathrm{d}(q, x)$ zu q größer als d ist. Sei $M := \{x \in D \ : \ \mathrm{d}\,(q, x) > d\}$; dann gilt:

$$q \in \mathrm{DB}(p, d) :\Longleftrightarrow \ \frac{|M|}{|D|} \geq \pi. \tag{2.1}$$

Diese Definition ist exakt, benötigt allerdings ein Modell, in dem eine Metrik vorhanden ist. Die Parameter π und d muss man im Vorhinein festlegen. Dazu benötigt man A-priori-Informationen über die Daten. Insbesondere ein schlechter Wert von d kann zu einem schlechten Ergebnis führen: Wählt man d zu klein, klassifiziert der Algorithmus möglicherweise zu viele Daten als Ausreißer. Wählt man d zu groß, erkennt der Algorithmus möglicherweise manche Ausreißer nicht mehr [Agg01b]:

[Ram00] überarbeiteten die Definition von [Kno98].

Definition 2.8. Für ein Datum q einer Menge von Daten D sei $d^k(q)$ der Abstand zum k-nächsten Nachbarn (d. h. es gibt genau $k-1$ Daten $x \in D, x \neq q$ mit $\mathrm{d}\,(q, x) \leq d^k(q)$). Ein Datum q ist ein $D^k_{\mathfrak{n}}$-*Ausreißer* wenn es maximal $\mathfrak{n} - 1$ Daten $x \in D$ gibt mit $d^k(q) < d^k(x)$.

Definition 2.8 hat gegenüber Definition 2.7 den Vorteil, dass die Wahl der Parameter einfacher ist. Weiters erhält man durch die Werte $d^k(x)$ eine Reihung der Ausreißer.

2.3 Arten von Ausreißern

Definition 2.9. Ein einzelnes Datum, das für sich alleine von den restlichen Daten abweicht, wird als *Point Outlier* oder *Global Outlier* bezeichnet.

Point Outlier sind die einfachste Art von Ausreißern, die meisten Verfahren suchen nach dieser Art. Bei der Suche nach Point Outliern ist das Verfahren, mit der die Abweichung festgestellt wird (z. B. die verwendete Metrik), wesentlich ([Han12], [Sin12]).

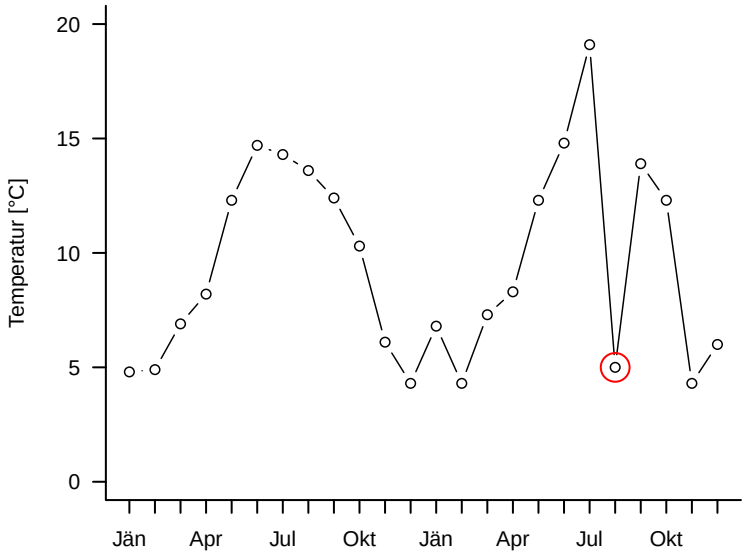

Abbildung 2.1: Temperaturverlauf Nottingham Castle 1920 und 1921. Die Temperatur im August 1921 ist ein Contextual Outlier.

Definition 2.10. Ein Datum, das nur im Rahmen eines bestimmten Kontexts ein Ausreißer ist, heißt *Contextual Outlier*. Die Daten besitzen zwei Arten von Attributen: Die *Contextual Attributes* legen den Kontext des Datums fest. Die *Behavioral Attributes* legen die Charakteristik des Datums fest.

Beispiel 2.11. Betrachte Abbildung 2.1: Eine Lufttemperatur von 5 °C in Nottingham Castle ist im Jänner kein Ausreißer, im August schon. Die Lufttemperatur ist ein Behavioral Attribute, Ort und Kalenderdatum sind Contextual Attributes. Die Daten gehören zum `nottem`-Dataset von R [RCT12].

Wie in Beispiel 2.11 kommen Contextual Outliers häufig bei der Analyse von Zeitreihen oder von Geodaten vor. Zeit bzw. Ort bilden dabei die Contextual Attributes. Die Suche nach Point Outliern ist ein Spezialfall der Suche nach Contextual Outliern, bei der die Menge der Contextual Attributes leer ist.

Definition 2.12. Eine Teilmenge der Daten wird *Collective Outlier* genannt, wenn die Teilmenge als Ganzes von der Gesamtmenge der Daten oder der Struktur der Gesamtmenge abweicht. Die einzelnen Elemente der Teilmenge für sich alleine müssen keine Point Outlier sein.

Beispiel 2.13. Eine einzelne Anfrage im Rahmen eines Denial-of-Service-Angriffs auf einen Server ist kein Ausreißer. Alle Anfragen des Angriffs bilden einen Collective Outlier.

Damit ein Algorithmus Collective Outlier finden kann, benötigt er Informationen über die Beziehung zwischen den einzelnen Daten.

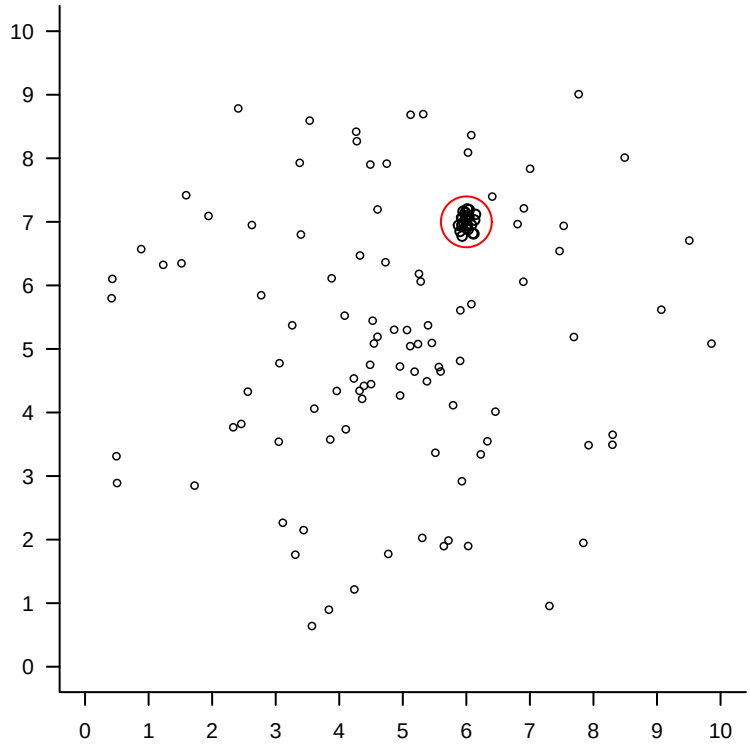

Abbildung 2.2: Collective Outlier bei $(6, 7)$

2.4 Unterschied zwischen Ausreißern, Noise und Novelty

2.4.1 Noise

Wie der Begriff Ausreißer wird auch der Begriff *Noise* unterschiedlich definiert, abhängig vom Zugang:

- Noise wird vom selben Mechanismus wie die normalen Daten erzeugt, weicht von den normalen Daten aber aufgrund der Varianz oder von Zufallsfehlern ab. Ausreißer werden dagegen durch einen anderen Mechanismus erzeugt (siehe Definition 2.5) [Han12].
- Noise besteht aus Daten, die für die jeweilige Datenanalyse nicht wichtig sind, aber die Datenanalyse erschweren [Sin12].
- Noise besteht aus Daten, die zwischen normalen Daten und Ausreißern liegen. Die Grenzen zwischen diesen drei Mengen sind oft nicht eindeutig und ergeben sich aus der jeweiligen Anwendung [Agg13].

Zum letzten Punkt bemerkt Aggarwal noch:

> Furthermore, the separation between noise and anomalies is not pure, and many data points created by a noisy generative process may be deviant enough to be interpreted as anomalies on the basis of the outlier score. [...] it is the interest of the analyst, which regulates the distinction between noise and an anomaly. [...] The crucial point to understand here is that anomalies need to be unusual in an interesting way [...]. [Agg13, S. 4–5]

Ausreißer sind „interessant“, denn es sind mit ihnen die Fragen verbunden: Warum gibt es sie? Und: Wodurch werden sie verursacht? Noise dagegen ist uninteressant, weil dessen Herkunft, Erzeugung usw. uninteressant ist. Viele Verfahren zum Finden von Ausreißern sind auch zum Finden von Noise geeignet, da der Unterschied häufig eben rein in der Beurteilung durch den Anwender liegt.

Der Umgang mit Noise gehört im Rahmen des Knowledge Discovery in Databases zum Data-Cleaning. Daher kümmert man sich um Noise vor der Ausreißererkennung. [Sin12] unterscheidet dabei zwei Möglichkeiten:

Noise-Removal Man entfernt die Daten, die als Noise gelten.

Noise-Accommodation Man verändert das Modell derart, dass es gegen Noise robust bzw. immun ist.

Beispiel 2.14. Betrachte Abbildung 2.3: Die weißen Punkte wurden mit einer zweidimensionalen Normalverteilung mit unkorrellierten Koordinaten erzeugt. Bei beiden Koordinaten beträgt der Mittelwert 5 und die Varianz 1. Der rote Kreis hat den Mittelpunkt $(5, 5)$ und einen Radius von ~ 2.07. Dadurch liegen 10 weiße Punkte außerhalb des Kreises. Die blauen Punkte wurden mit einem anderen Mechanismus erzeugt: Einer Gleichverteilung im Gebiet $[0, 10]^2$.

Die Abbildung zeigt einige Probleme bei der Unterscheidung zwischen Noise und Ausreißern auf. Nach der Definition von [Han12] ist der Punkt z Noise. Die Punkte w, x und y sind Ausreißer, da sie von einem anderen Mechanismus erzeugt wurden. Laut [Agg13] dagegen sind die Punkte x, y und z Noise, Punkt w zählt er zu den normalen Daten. Auch die meisten Algorithmen zur Ausreißererkennung würden die Punkte w, x, y und z auf diese Art bewerten. Ob als Ausreißer, Noise oder normale Daten hängt vom konkreten Algorithmus und dessen Parametern ab.

Punkt w wurde zwar von einem anderen Mechanismus erzeugt, wird aber von allen Algorithmen zu den normalen Daten gezählt, da er zufällig bei den normalen Daten liegt.

2.4.2 Novelty

Noveltys sind Daten, die sich von den vorhandenen Daten unterscheiden, aber nicht als Ausreißer, sondern als normale Daten klassifiziert werden sollen. Wenn Noveltys zum ersten Mal auftreten, erscheinen sie häufig als Ausreißer. Das Modell muss dann so abgeändert (oder angepasst) werden, sodass Noveltys als normale Daten gelten.

Beispiel 2.15. In einem Computernetz verwenden normale Benutzer ein neues Verfahren zum Datenaustausch. Das soll nicht als Angriff gewertet werden, sondern als neue, aber normale Kommunikation.

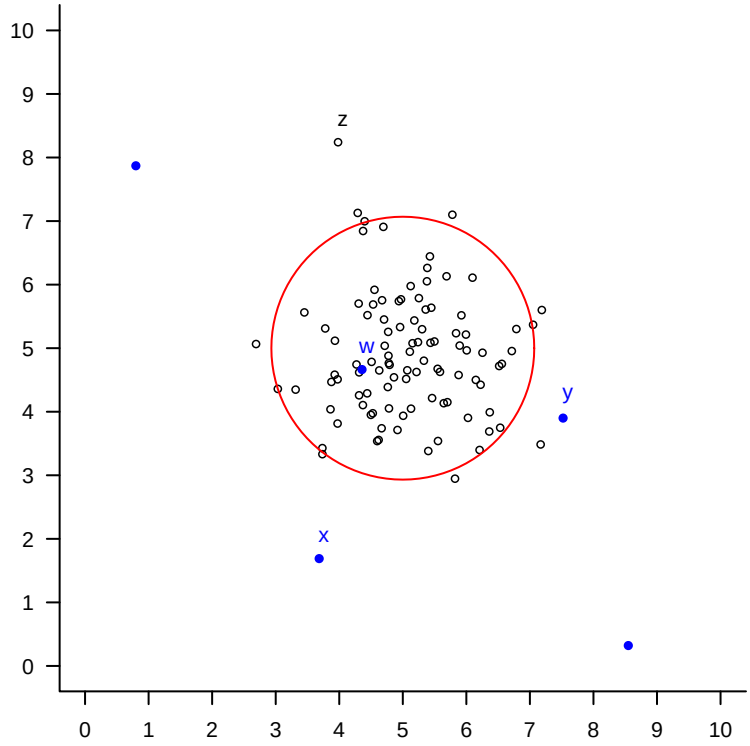

Abbildung 2.3: Ausreißer (blaue Punkte) und Noise (weiße Punkte außerhalb des roten Kreises)

3 Verfahren zur Ausreißererkennung

Die Verfahren zur Ausreißererkennung kann man nach unterschiedlichen Gesichtspunkten einteilen. Zunächst kann das Modell des Verfahrens mit oder ohne Trainingsdaten angepasst werden:

Überwachte Ausreißererkennung (Supervised Outlier Detection) Die Trainingsdaten umfassen sowohl normale Daten als auch Ausreißer. Ein Problem bei diesen Verfahren ist es, gute Trainingsdaten zu bekommen. Häufig gibt es weitaus mehr normale Daten als Ausreißer. Durch dieses Ungleichgewicht kann es leicht zu einer Überanpassung kommen ([Agg13, S. 29]). Außerdem decken die Trainingsdaten unter Umständen nicht alle möglichen normalen und außergewöhnlichen Systemzustände ab.

Semi-überwachte Ausreißererkennung (Semi-Supervised Outlier Detection) Hier beinhalten die Trainingsdaten normale Daten, aber keine Ausreißer. Solche Verfahren verwenden ein Modell, das an die normalen Daten angepasst ist. Ein Datum, das diesem Modell nicht entspricht, gilt als Ausreißer. Der Vorteil von solchen Verfahren ist umso größer, je schwieriger Daten über Ausreißer zu bekommen sind.

Unüberwachte Ausreißererkennung (Unsupervised Outlier Detection) Diese Verfahren benötigen keine Trainingsdaten. Manchmal benötigt der Anwender allerdings Apriori-Wissen um Parameter des Verfahrens richtig zu setzen. Die Modelle basieren oft auf der Annahme, dass sich die normalen Daten ähnlich sind, also einen Cluster bilden. Wenn diese Annahme nicht stimmt, versagen diese Verfahren (beachte z. B. Abbildung 2.2: Die Daten, die den Collective Outlier bilden, sind sich ähnlich, die normalen Daten aber nicht).

Weiters kann man die Verfahren nach der Art des Ergebnisses unterscheiden:

Score Jedes Datum wird mit einer Zahl bewertet. Die Zahl liefert eine Aussage, inwieweit das Datum ein Ausreißer ist. Der Anwender kann entweder die Daten mit der extremsten Bewertung als Ausreißer betrachten oder alle Daten, deren Bewertung eine gewisse Grenze überschreitet.

Label Jedes Datum wird entweder als Ausreißer oder als normales Datum gekennzeichnet. Bei manchen dieser Verfahren kann der Anwender die Grenze zwischen diesen beiden Klassen durch die Wahl der Parameter des Verfahrens bestimmen.

Bei der Entscheidung, ob ein bestimmtes Datum ein Ausreißer ist, kann der Algorithmus des verwendeten Verfahrens alle anderen Daten verwenden, oder nur einen Teil der andern Daten:

Globale Verfahren Alle anderen Daten werden für die Entscheidung herangezogen. Globale Verfahren verwenden die Annahme, dass nur ein einziger Mechanismus die normalen Daten erzeugt.

Lokale Verfahren Nur eine Teilmenge der anderen Daten wird für die Entscheidung verwendet. Das Problem besteht bei diesen Verfahren darin geeignete Teilmengen zu finden.

Schließlich kann man die Verfahren aufgrund des verwendeten Modells einteilen. Es existiert eine Vielzahl an verwendeten Modellen, daher gibt es in der Literatur keine einheitliche Variante die Modelle zu klassifizieren (z. B. werden tiefen- und abweichungsbasierte Verfahren manchmal zu den statistischen Verfahren gezählt). Außerdem ist diese Klassifizierung bei vielen Verfahren nicht eindeutig. Verschiedene Autoren teilen das gleiche Verfahren mitunter verschiedenen Klassen zu. Die folgende Auflistung gibt eine ungefähren Überblick über die eingesetzten Verfahren.

Statistische Verfahren (statistical models) Die grundlegende Annahme dieser Verfahren ist, dass die Verteilung der Daten einem statistischen Modell entspricht. Ausreißer sind Daten, die nach diesem Modell nur mit geringer Wahrscheinlichkeit auftreten.

Tiefenbasierte Verfahren (depth-based models) Bei diesen Verfahren nimmt man an, dass die Ausreißer am Rand liegen. Man findet sie, indem man z. B. die konvexe Hülle berechnet.

Abweichungsbasierte Verfahren (deviation-based models) Diese Verfahren berechnen den Einfluss einzelner Daten auf die Varianz. Die Annahme ist, dass Ausreißer einen größeren Einfluss haben als normale Daten.

Abstands- und dichtebasierte Verfahren (distance-based, density-base models) Abstandsbasierte Verfahren betrachten die Abstände zwischen den Daten; dichtebasierte Verfahren betrachten die Dichte um die Daten, um Ausreißer zu finden.

Maschinelles Lernen Die Verfahren dieser Gruppe verwenden z. B. Neuronale Netze, Support Vector Machines oder Decision Trees zur Ausreißererkennung.

Clusterbasierte Verfahren (clustering-based models) Mithilfe von Algorithmen aus der Clusteranalyse werden die Daten zu Gruppen (Clustern) zusammengefasst. Daten die zu keiner Gruppe oder zu einer Gruppe mit wenigen Daten gehören, gelten als Ausreißer.

Zusätzlich gibt es noch Klassen, die nur in einzelnen Werken vorkommen, z. B. informationstheoretische Verfahren in [Agg13, S. 16]. Aufgrund der speziellen Probleme bei der Ausreißererkennung in höherdimensionalen Räumen fasst eine Reihe von Autoren ([Agg13], [Han12]) die Algorithmen, die für diese Aufgabenstellung geeignet sind, ebenfalls in einer eigenen Klasse zusammen.

Um die Nachteile der einzelnen Verfahren zu umgehen, kann man zwei oder mehr Verfahren zu *Ensembles* oder *hybriden Systemen* kombinieren.

4 Fluch der Dimensionalität

In diesem Kapitel beschäftigen wir uns mit dem Hauptproblem der Ausreißererkennung in höherdimensionalen Räumen: Dem *Fluch der Dimensionalität* (curse of dimensionality).

Definition 4.1. Sei $d^1(q)$ der Abstand zum nächsten Nachbarn eines Abfragedatums q. Die Nearest-Neighbor-Search heißt ε-δ-*instabil* wenn bei einem Anteil δ der Daten der Abstand zu q kleiner $(1+\varepsilon)d^1(q)$ ist, d. h. wenn gilt:

$$\frac{\left|\{x \in D \mid \mathrm{d}(q,x) < (1+\varepsilon)d^1(q)\}\right|}{N} \geq \delta.$$

Der Satz von Beyer et al. zeigt, dass unter verhältnismäßig schwachen Voraussetzungen mit wachsender Dimension eine Nearest-Neighbor-Search mit Wahrscheinlichkeit 1 ε-δ-instabil wird. Das bedeutet, dass der Unterschied der Abstände zwischen Ausreißern und den normalen Daten immer kleiner wird.

Definition 4.2. Der *relative Kontrast* (relative contrast) ist definiert als

$$\frac{\max\limits_{x \in D}(\|x\|) - \min\limits_{x \in D}(\|x\|)}{\min\limits_{x \in D}(\|x\|)}.$$

Wenn wir den Ursprung als Abfragepunkt nehmen, $q = \mathbf{0}$, folgt aus dem Satz von Beyer et al. dass der relative Kontrast in Wahrscheinlichkeit gegen 0 konvergiert. Die Differenz der Abstände zwischen nächstem und entferntesten Nachbarn wächst also langsamer als der Abstand zum nächsten Nachbarn. [Agg01a] zeigen, dass die verwendete Norm (bzw. Metrik) einen großen Einfluss auf die Konvergenzgeschwindigkeit hat (siehe Kapitel 4.3).

4.1 Grundlagen aus der Wahrscheinlichkeitstheorie

Mit dem Begriff *Zufallsgröße* meinen wir entweder eine Zufallsvariable oder einen Zufallsvektor auf einem Wahrscheinlichkeitsraum $(\Omega, \mathfrak{S}, \mathbb{P})$. Für Zufallsgrößen verwenden wir Großbuchstaben (X, Y, ...), Folgen von Zufallsgrößen bezeichnen wir mit X_n. Die Zeichen $\to_{f.\,s.}$ und $\to_p$ verwenden wir wie üblich für die Konvergenz fast sicher und die Konvergenz in Wahrscheinlichkeit:

Definition 4.3. Eine Folge von reellen Zufallsgrößen X_n *konvergiert fast sicher* gegen die reelle Zufallsgröße X, im Zeichen $X_n \to_{f.\,s.} X$, wenn gilt:

$$\mathbb{P}\left[\{\omega \in \Omega \ : \ \lim_{n\to\infty} X_n(\omega) = X(\omega)\}\right] = 1.$$

Definition 4.4. Eine Folge von reellen Zufallsgrößen X_n *konvergiert in Wahrscheinlichkeit* gegen die reelle Zufallsgröße X, im Zeichen $X_n \to_p X$, wenn für jedes $\varepsilon > 0$ gilt:

$$\lim_{n\to\infty} \mathbb{P}\left[|X_n - X| > \varepsilon\right] = 0.$$

Satz 4.5. *Es gilt $X_n \to_p X$ genau dann, wenn es für jede Teilfolge X_{n_i} von X_n eine Teilfolge $X_{n_{i_j}}$ von X_{n_i} gibt mit $X_{n_{i_j}} \to_{f.s.} X$.*

Beweis. Siehe [Sch11]. □

Aussagen folgender Art über eine stetige Funktion g sind als *continuous mapping theorem* bekannt.

Satz 4.6. *Sei $g : \mathbb{R} \to \mathbb{R}$ eine stetige Funktion.*

$$X_n \to_{f.s.} X \Rightarrow g(X_n) \to_{f.s.} g(X),$$
$$X_n \to_p X \Rightarrow g(X_n) \to_p g(X).$$

Beweis. Die erste Implikation folgt direkt aus der Stetigkeit von g. Um die zweite Implikation zu beweisen betrachten wir eine beliebige Teilfolge $g(X_{n_i})$ der Folge $g(X_n)$: Aufgrund von Satz 4.5 und der Voraussetzung $X_n \to_p X$ können wir eine Teilfolge $X_{n_{i_j}}$ von X_{n_i} finden mit $X_{n_{i_j}} \to_{f.s.} X$. Aus der ersten Implikation folgt $g(X_{n_{i_j}}) \to_{f.s.} g(X)$. Wir haben somit eine Teilfolge unserer beliebig gewählten Teilfolge $g(X_n)$ gefunden, die fast sicher gegen $g(X)$ konvergiert. Wiederum aufgrund von Satz 4.5 gilt daher $g(X_n) \to_p g(X)$. □

Satz 4.7. *Seien X_n und Y_n zwei Folgen von reellen Zufallsvariablen. Weiters gelte $X_n \to_p \zeta$ und $Y_n \to_p \xi$ mit $\zeta,\ \xi \in \mathbb{R}$. Dann folgt:*

$$X_n + Y_n \to_p \zeta + \xi,$$
$$X_n Y_n \to_p \zeta\xi.$$

Ist zusätzlich $\xi \neq 0$ gilt darüber hinaus

$$X_n / Y_n \to_p \zeta/\xi.$$

Beweis. Mit $X \equiv \zeta$ und $Y \equiv \xi$ wird der Beweis wortwörtlich wie in Satz 4.6 geführt. □

Mithilfe der Tschebyscheff-Ungleichung

$$\mathbb{P}\left[|X - \mathbb{E}X| > \delta\right] \leq \frac{\mathbb{V}X}{\delta^2}$$

($\delta \in \mathbb{R}$, $\delta > 0$) beweisen wir das folgende Lemma:

Lemma 4.8. *Sei X_n eine Folge von reellen Zufallsvariablen mit $\lim_{n\to\infty} \mathbb{E}X_n = \zeta$ und mit $\lim_{n\to\infty} \mathbb{V}X_n = 0$. Dann gilt*

$$X_n \to_p \zeta.$$

Beweis. Da die Varianz gegen 0 konvergiert, gilt

$$\forall \varepsilon > 0 : \exists n_1(\varepsilon) : \mathbb{P}\left[\left|X_{n_1(\varepsilon)} - \mathbb{E}X_{n_1(\varepsilon)}\right| > \frac{\delta}{2}\right] \leq \varepsilon.$$

Und da der Erwartungswert gegen ζ konvergiert, gilt

$$\forall \delta > 0 : \exists n_2(\delta) : \left|\zeta - \mathbb{E}X_{n_2(\varepsilon)}\right| < \frac{\delta}{2}.$$

Insgesamt daher:

$$\exists n : \mathbb{P}\left[|X_n - \zeta| > \delta\right] \leq \varepsilon.$$

Da ε beliebig gewählt war, folgt die Behauptung. □

4.2 Der Satz von Beyer et al.

Definition 4.9. In diesem Abschnitt verwenden wir $\forall n \in \mathbb{N}$ die folgende Bezeichnungen.

- Es sei D^n eine Folge von Datenräumen, $|D^n| := N$.
- $p \in \mathbb{R}$, $p > 0$ ist eine beliebige reelle, positive Konstante.
- $F_1^{data}, F_2^{data}, \ldots$ und $F_1^{query}, F_2^{query}, \ldots$ seien zwei Folgen von Verteilungsfunktionen.
- $X_{n,1}, \ldots, X_{n,N} \sim F_n^{data}$ bilden N voneinander unabhängige Zufallsvektoren. Die Realisationen dieser Zufallsvektoren bilden die Daten $x_{n,1}, \ldots, x_{n,N} \in D^n$.
- $Q_n \sim F_n^{query}$ ist ein Zufallsvektor, seine Realisation ist ein Abfragedatum $q_n \in D^n$.
- d_{min} und d_{max} stehen für den minimalen und maximalen Abstand zwischen q_n und den restlichen Daten von D^n:

$$d_{min} := \min\left(\{\mathrm{d}(x_{n,i}, q_n) \mid 1 \leq i \leq N-1\}\right),$$
$$d_{max} := \max\left(\{\mathrm{d}(x_{n,i}, q_n) \mid 1 \leq i \leq N-1\}\right).$$

Satz 4.10 (Satz von Beyer et al.)**.** *Mit den Definitionen von 4.9 gilt für jedes reelle $\varepsilon > 0$ und jedes $i \in \{1, \ldots, N\}$:*

$$\lim_{n\to\infty} \mathbb{V}\left[\frac{(\mathrm{d}(X_{n,i}, Q_n))^p}{\mathbb{E}[(\mathrm{d}(X_{n,i}, Q_n))^p]}\right] = 0 \quad \Rightarrow \quad \lim_{n\to\infty} \mathbb{P}[d_{max} \leq (1+\varepsilon) d_{min}] = 1.$$

Beweis. Um die Notation zu vereinfachen definieren wir

$$\zeta_n := \mathbb{E}[(\mathrm{d}(X_{n,i}, Q_n))^p]$$

Beachte, dass ζ_n unabhängig von i ist, da die $X_{n,i}$ alle gleich verteilt sind. Es gilt trivialerweise

$$\mathbb{E}\left[\frac{(\mathrm{d}(X_{n,i}, Q_n))^p}{\zeta_n}\right] = 1,$$

und weiters:

$$\lim_{n\to\infty} \mathbb{E}\left[\frac{(\mathrm{d}(X_{n,i}, Q_n))^p}{\zeta_n}\right] = 1.$$

Gemeinsam mit der Voraussetzung

$$\lim_{n\to\infty} \mathbb{V}\left[\frac{(\mathrm{d}(X_{n,i}, Q_n))^p}{\zeta_n}\right] = 0$$

und Lemma 4.8 folgt

$$\frac{(\mathrm{d}(X_{n,i}, Q_n))^p}{\zeta_n} \quad \to_p \quad 1.$$

Sei nun

$$X_n := \left(\frac{(\mathrm{d}(X_{n,1}, Q_n))^p}{\zeta_n}, \ldots, \frac{(\mathrm{d}(X_{n,N}, Q_n))^p}{\zeta_n}\right)$$

ein Zufallsvektor. Es gilt

$$X_n \to_p (1, \ldots, 1).$$

Die Funktionen min und max sind stetig, daher folgt aus den Sätzen 4.6 und 4.7:

$$\min(X_n) \to_p \min(1, \ldots, 1) = 1,$$
$$\max(X_n) \to_p \max(1, \ldots, 1) = 1,$$

$$\frac{\max(X_n)}{\max(X_n)} \to_p \frac{1}{1} = 1.$$

Es ist $d_{max} = \zeta_n \max(X_n)$ und $d_{min} = \zeta_n \min(X_n)$ und somit gilt

$$\frac{d_{max}}{d_{min}} = \frac{\zeta_n \max(X_n)}{\zeta_n \min(X_n)} = \frac{\max(X_n)}{\min(X_n)},$$
$$\frac{d_{max}}{d_{min}} \to_p 1.$$

Laut der Definition der Konvergenz in Wahrscheinlichkeit ist dieser Ausdruck äquivalent zu

$$\lim_{n\to\infty} \mathbb{P}\left[\left|\frac{d_{max}}{d_{min}} - 1\right| \leq \varepsilon\right] = 1.$$

Da $d_{max} \geq d_{min}$ ist, kann man den Betrag ignorieren. Nach einer Umformung erhält man das gewünschte Ergebnis:

$$\lim_{n\to\infty} \mathbb{P}\left[d_{max} \leq (1+\varepsilon)\, d_{min}\right] = 1.$$

□

Man beachte: In der Praxis bezeichnet n die Dimension des Datenraums und p den Exponenten der verwendeten Minkowski-Metrik. Im Satz von Beyer et al. verwenden wir diese Interpretation allerdings nicht. Die Variablen n und p können beliebige Größen beschreiben.

[Bey99] gehen in Folge auf zwei Fragen ein:

- Wie schnell konvergiert die Wahrscheinlichkeit? Bzw. ab welcher Dimension kann man das nächste Datum nur noch schwer von dem am weitest entfernten Datum unterscheiden?
- Inwieweit ist die Bedingung des Satzes (Varianz des Abstands dividiert durch den Erwartungswert konvergiert gegen 0) eine Einschränkung?

Um die erste Frage zu beantworten, führten [Bey99] einige Experimente durch. Bei diesen Experimenten wurde der Effekt bereits bei Daten mit 10–15 Dimensionen sichtbar.

Weiters stellten [Bey99] einige Beispiele vor, in denen die Bedingung erfüllt ist:

Beispiel 4.11. Alle $X_{n,i}$ sind unabhängig und identisch verteilt, alle notwendigen Momente sind endlich und Q_n ist unabhängig von allen $X_{n,i}$.

Beispiel 4.12. Ein Zufallsvektor $X_{n,i} = (X^1_{n,i}, X^2_{n,i}, \ldots, X^n_{n,i})$ wird folgendermaßen definiert: Die stetige Zufallsvariable U_i ist gleichverteilt im Intervall $[0, \sqrt{i}]$.

$$X^1_{n,i} := U_i,$$
$$X^j_{n,i} := U_j + \sum_{k=1}^{j-1} \frac{U_k}{2^{n-k}}.$$

In diesem Beispiel sind alle Dimensionen verschieden verteilt und voneinander abhängig, die Bedingung für den Satz von Beyer et al. ist trotzdem erfüllt.

Beispiel 4.13. Jetzt sind die Koordinaten des Zufallsvektors $X_{n,i} = (X^1_{n,i}, X^2_{n,i}, \ldots, X^n_{n,i})$ alle normalverteilt:

$$X^j_{n,i} := \mathcal{N}(0, 1/j).$$

Da die Varianzen der Zufallsvariablen mit höheren Indizes kleiner sind, könnte man annehmen, dass die Zufallsvariablen mit niedrigeren Indizes bei der Abstandsberechnung dominieren. Das ist aber nicht der Fall, man kann den Satz von Beyer et al. anwenden.

Beyer et al. kommen zu dem Schluss: „The conditions we have identified in which this happens are much broader than the independent and identically distributed (IID) dimensions assumption that other work assumes“ [Bey99, S. 218].

4.3 Der Einfluss der Norm

Die Sätze in diesem Abschnitt zeigen, dass zwischen dem relativen Kontrast und der verwendeten Norm ein enger Zusammenhang besteht.

Lemma 4.14. *Sei X eine gleichverteilte, reelle Zufallsvariable im Intervall $[0, 1]$. Dann gilt:*

$$\mathbb{E}X^p = \frac{1}{p+1}$$
$$\mathbb{V}X^p = \left(\frac{p}{p+1}\right)^2 \left(\frac{1}{2p+1}\right).$$

Beweis.

$$\begin{aligned}\mathbb{E}X^p &= \int_0^1 x \cdot x^{p-1} dx \\ &= \frac{1}{p+1}.\end{aligned}$$

$$\begin{aligned}\mathbb{V}X^p &= \mathbb{E}X^{2p} - \mathbb{E}^2 X^p \\ &= \left(\frac{1}{2p+1}\right) - \left(\frac{1}{p+1}\right)^2 \\ &= \left(\frac{p}{p+1}\right)^2 \left(\frac{1}{2p+1}\right).\end{aligned}$$

□

Satz 4.15. *Seien X_1 und X_2 zwei reelle, n-dimensionale Zufallsvektoren, in allen Koordinaten gleichverteilt im Intervall $[0,1]$. Dann gilt:*

$$\lim_{n\to\infty} \mathbb{E}\left[\frac{(\max(\|X_1\|_p, \|X_2\|_p) - \min(\|X_1\|_p, \|X_2\|_p))\ \sqrt{n}}{\sqrt[p]{n}}\right] = C\frac{1}{\sqrt[p]{p+1}}\sqrt{\frac{1}{2p+1}}.$$

Beweis. Wenn wir Lemma 4.14 auf jede der n Koordinaten von X_1 und X_2 anwenden erhalten wir:

$$\mathbb{E}\left[\|X_1\|_p\right] = \mathbb{E}\left[\|X_2\|_p\right] = \sqrt[p]{n\frac{1}{p+1}}.$$

Daraus folgt (analog auch für X_2):

$$\frac{\|X_1\|_p}{\sqrt[p]{n}} \to_p \sqrt[p]{\frac{1}{p+1}}. \tag{4.1}$$

Wir bringen $|\|X_1\|_p - \|X_2\|_p|$ in eine andere Form und erweitern mit $\sqrt{n}/\sqrt[p]{n}$:

$$\begin{aligned}|\|X_1\|_p - \|X_2\|_p| &= \frac{\left|\|X_1\|_p^p - \|X_2\|_p^p\right|}{\sum_{r=0}^{p-1} \|X_1\|_p^{p-r-1} \|X_2\|_p^r} \\ \frac{|\|X_1\|_p - \|X_2\|_p|\ \sqrt{n}}{\sqrt[p]{n}} &= \frac{\left|\|X_1\|_p^p - \|X_2\|_p^p\right|\ \frac{1}{\sqrt{n}}}{\sum_{r=0}^{p-1} \left(\frac{\|X_1\|_p}{\sqrt[p]{n}}\right)^{p-r-1} \left(\frac{\|X_2\|_p}{\sqrt[p]{n}}\right)^r}.\end{aligned}$$

Auf den Nenner der rechten Seite wenden wir Gleichung 4.1 und Satz 4.7 an:

$$\begin{aligned}\sum_{r=0}^{p-1} \left(\frac{\|X_1\|_p}{\sqrt[p]{d}}\right)^{p-r-1} \left(\frac{\|X_2\|_p}{\sqrt[p]{d}}\right)^r &\to_p \sum_{r=0}^{p-1} \left(\frac{1}{p+1}\right)^{\frac{p-r-1}{p}} \left(\frac{1}{p+1}\right)^{\frac{r}{p}} &(4.2)\\ &\to_p p\left(\frac{1}{p+1}\right)^{(p-1)/p}. &(4.3)\end{aligned}$$

Wir definieren neue Zufallsvariablen Y_i:

$$Y_i := X_{1,i}^p - X_{2,i}^p,$$

wobei $X_{1,i}$ und $X_{2,i}$ die i-ten Koordinaten von X_1 und X_2 sind. Es gilt $\mathbb{E}Y_i = 0$ und aufgrund von Lemma 4.14

$$\mathbb{V}Y_i = 2\left(\frac{p}{p+1}\right)^2\left(\frac{1}{2p+1}\right).$$

Betrachten wir die Summe $\sum_{i=1}^n Y_i$. Aufgrund des zentralen Grenzverteilungssatzes konvergiert diese Summe gegen eine Normalverteilung mit Erwartungswert 0 und einer Standardabweichung von

$$\sqrt{2n\left(\frac{p}{p+1}\right)^2\left(\frac{1}{2p+1}\right)}.$$

Daraus folgt

$$\lim_{n\to\infty}\mathbb{E}\left[\sum_{i=1}^n Y_i\frac{1}{\sqrt{n}}\right] = \lim_{n\to\infty}\mathbb{E}\left[\left|\|X_1\|_p^p - \|X_2\|_p^p\right|\ \frac{1}{\sqrt{n}}\right] = C\frac{p}{p+1}\sqrt{\frac{1}{2p+1}}, \tag{4.4}$$

wobei C eine passende reelle Konstante ist. Wir können jetzt Formel 4.3 und Gleichung 4.4 kombinieren und erhalten abschließend das gewünschte Ergebnis:

$$\lim_{n\to\infty}\mathbb{E}\left[\frac{|\|X_1\|_p - \|X_2\|_p|\ \sqrt{n}}{\sqrt[p]{n}}\right] = C\frac{1}{\sqrt[p]{p+1}}\sqrt{\frac{1}{2p+1}}.$$

□

Satz 4.16. *Wie im vorhergehenden Satz seien X_1 und X_2 zwei reelle, n-dimensionale Zufallsvektoren, in allen Koordinaten gleichverteilt im Intervall $[0,1]$. Dann gilt:*

$$\lim_{n\to\infty}\mathbb{E}\left[\frac{\max(\|X_1\|_p,\|X_2\|_p) - \min(\|X_1\|_p,\|X_2\|_p)}{\min(\|X_1\|_p,\|X_2\|_p)}\ \sqrt{n}\right] = C\sqrt{\frac{1}{2p+1}}.$$

Beweis. Aufgrund von Formel 4.1 und Satz 4.6 gilt

$$\frac{\|X_1\|_p}{\sqrt[p]{n}} \to_p \sqrt[p]{\frac{1}{p+1}}, \quad \frac{\|X_2\|_p}{\sqrt[p]{n}} \to_p \sqrt[p]{\frac{1}{p+1}},$$

$$\Rightarrow \quad \min\left(\frac{\|X_1\|_p}{\sqrt[p]{n}}, \frac{\|X_2\|_p}{\sqrt[p]{n}}\right) \to_p \min\left(\sqrt[p]{\frac{1}{p+1}}, \sqrt[p]{\frac{1}{p+1}}\right) = \sqrt[p]{\frac{1}{p+1}}.$$

Aus Satz 4.15 folgt nun Satz 4.16. □

Korollar 4.17. *Seien die X_i, $1 \le i \le k$, insgesamt k reelle, n-dimensionale Zufallsvektoren, in allen Koordinaten gleichverteilt im Intervall $[0,1]$. Dann gilt:*

$$\frac{C}{\sqrt[p]{p+1}}\sqrt{\frac{1}{2p+1}} \le \lim_{n\to\infty}\mathbb{E}\left[\frac{(\max(\|X_i\|_p) - \min(\|X_i\|_p))\ \sqrt{n}}{\sqrt[p]{n}}\right] \le \frac{(k-1)\ C}{\sqrt[p]{p+1}}\sqrt{\frac{1}{2p+1}}.$$

Beweis. Die erste Ungleichung folgt aus dem Umstand, dass der betreffende Erwartungswert monoton wachsend in k ist. Bezeichnen wir mit x den erwarteten Abstand zwischen zwei zufällig gezogenen Daten, dann beträgt der erwartete Abstand zwischen drei zufällig gezogenen Daten maximal $2x$ und zwischen k zufällig gezogenen Daten maximal $(k-1)x$. Daraus folgt die zweite Ungleichung. □

[Agg01a] präsentieren auch eine verallgemeinerte Version des Satzes 4.15 und des Korollars 4.17 für beliebige Verteilungen:

Satz 4.18. *Seien X_1 und X_2 zwei reelle, n-dimensionale Zufallsvektoren, in allen Koordinaten identisch, aber beliebig verteilt. Dann gilt:*

$$\lim_{n\to\infty} \mathbb{E}\left[\frac{(\max(\|X_1\|_p, \|X_2\|_p) - \min(\|X_1\|_p, \|X_2\|_p))\ \sqrt{n}}{\sqrt[p]{n}}\right] = C_p,$$

wobei C_p eine reelle Konstante abhängig von p ist.

Korollar 4.19. *Seien die X_i, $1 \leq i \leq k$, insgesamt k Zufallsvektoren, in allen Koordinaten identisch, aber beliebig verteilt. Dann gilt:*

$$C_p \leq \lim_{n\to\infty} \mathbb{E}\left[\frac{(\max(\|X_i\|_p) - \min(\|X_i\|_p))\ \sqrt{n}}{\sqrt[p]{n}}\right] \leq (k-1)C_p,$$

wobei C_p eine reelle Konstante abhängig von p ist.

Die Beweise der Verallgemeinerungen sind denen von Satz 4.15 und Korollar 4.17 sehr ähnlich, die genaue Ausführung findet man in [Agg01a].

Die Korollare 4.17 und 4.19 zeigen, dass die Differenz zwischen dem weitest entfernten und dem nächsten Datum mit der Rate $\sqrt[p]{n}/\sqrt{n}$ wächst (Abbildung 4.1). Verwendet man ...

- die Summennorm mit $p = 1$, konvergiert die Differenz gegen unendlich.
- die euklidische Norm mit $p = 2$, ist die Differenz nach oben und unten durch Konstanten begrenzt.
- eine p-Norm mit $p \geq 3$ konvergiert die Differenz gegen 0.

Im Fall $p \geq 3$ konvergiert die Differenz umso schneller, je größer p ist.

Beispiel 4.20. Es wurden zufällig 200 Daten erzeugt, die Dimension des Datenraums betrug 2–15. Die Koordinaten waren in jeder Dimension gleichverteilt im Intervall $[0, 1]$. Als Normen wurden die p-Normen mit $p = 1$, 2 und 10 sowie die Quasinorm mit $p = 1/2$ verwendet (siehe Abschnitt 4.3.1). Der Versuch wurde 100-mal wiederholt, die Mittelwerte der Differenzen zwischen dem Datum mit der größten und der kleinsten Norm sind in Abbildung 4.1 dargestellt.

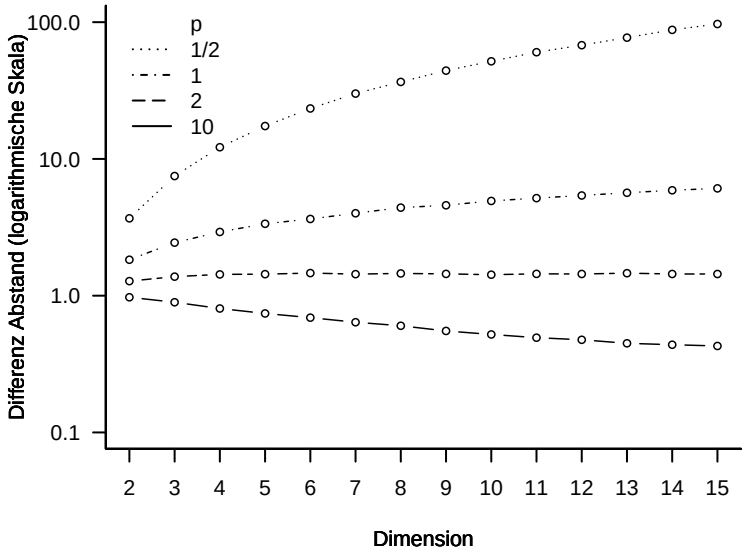

Abbildung 4.1: Differenz zwischen nächstem und entferntestem Datum bei gleichverteilten Daten, abhängig von Dimension und p-Norm

Beachte: Der Satz von Beyer et al. macht eine Aussage über den relativen Kontrast, die Sätze 4.15 und 4.18 sowie Korollar 4.17 und Korollar 4.19 über die Differenz zwischen den Abständen. Laut Satz von Beyer et al. konvergiert der relative Kontrast immer gegen 0, unabhängig von der verwendeten Norm. Korollar 4.17 und Korollar 4.19 liefern ein Indiz, dass der relative Kontrast umso schneller konvergiert, je größer der Wert von p ist.

Satz 4.16 beweist den Satz von Beyer et al. für den speziellen Fall, dass man zwei gleichverteilte Zufallsvariablen hat. Der Satz sagt darüber hinausgehend aus, dass der relative Kontrast mit der Rate $1/\sqrt{n}$ gegen 0 konvergiert. Für fixes n verringert sich der relative Kontrast mit der Rate $1/\sqrt{2p+1}$.

Weiters zeigen [Agg01a] in einem Experiment, dass der relative Kontrast mit hoher Wahrscheinlichkeit größer ist, wenn man statt der euklidischen Norm die Summennorm verwendet. Er kommt zu dem sehr eindeutigen Ergebnis, bei höherdimensionalen Daten möglichst die Summennorm zu bevorzugen.

> This is a surprising result in light of the fact that the Euclidean distance metric is traditionally used in a large variety of indexing structures and data mining applications. The widespread use of the Euclidean distance metric stems from the natural extension of applicability to spatial database systems [...] However, from the perspective of high dimensional data mining applications, this natural interpretability in 2 or 3-dimensional spatial systems is completely irrelevant. [Agg01a, S. 429]

4.3.1 Quasinorm und Fractional Distance Metric

Je kleiner p ist, desto langsamer schrumpft der relative Kontrast; am langsamsten, wenn man die Summennorm mit $p = 1$ verwendet. Daher liegt die Überlegung nahe, für p Werte kleiner 1 zu verwenden.

Definition 4.21. Eine Abbildung

$$\| \| : \mathbb{R}^n \to \mathbb{R}_+, \quad x \mapsto \|x\|$$

mit den Eigenschaften

1. $\|x\| \geq 0$, $\|x\| = 0 \iff x = \mathbf{0}$,
2. $\|a \cdot x\| = |a| \cdot \|x\| \quad \forall a \in \mathbb{R}$,
3. $\|x + y\| \leq C \cdot (\|x\| + \|y\|)$, $\quad C \in \mathbb{R}$ eine von x und y unabhängige Konstante ,

heißt *Quasinorm.*

Satz 4.22. *Eine Abbildung*

$$\| \|_p : \mathbb{R}^n \to \mathbb{R}_+, \quad x \mapsto \left(\sum_{i=1}^{n} |x_i|^{\,p} \right)^{1/p}$$

mit $0 < p < 1$ *ist eine Quasinorm.*

Beweis. Die Eigenschaften 1 und 2 sind trivialerweise erfüllt. Für Eigenschaft 3 rechnet man nach:

$$\begin{aligned}
\left(\sum_{i=1}^{n} |x_i + y_i|^{\,p} \right)^{1/p} &\leq \left(\sum_{i=1}^{n} \left(2 \max\left(|x_i|\,,\ |y_i|\right)\right)^p \right)^{1/p} \\
&= 2 \left(\sum_{i=1}^{n} \max\left(|x_i|^{\,p},\ |y_i|^{\,p}\right) \right)^{1/p} \leq 2 \left(\sum_{i=1}^{n} |x_i|^{\,p} + \sum_{i=1}^{n} |y_i|^{\,p} \right)^{1/p} \\
&\leq 2 \left(2 \max \left(\sum_{i=1}^{n} |x_i|^{\,p},\ \sum_{i=1}^{n} |y_i|^{\,p} \right) \right)^{1/p} \\
&= 2\ 2^{1/p} \max \left(\left(\sum_{i=1}^{n} |x_i|^{\,p} \right)^{1/p},\ \left(\sum_{i=1}^{n} |y_i|^{\,p} \right)^{1/p} \right) \\
&\leq 2^{(p+1)/p} \left(\left(\sum_{i=1}^{n} |x_i|^{\,p} \right)^{1/p} + \left(\sum_{i=1}^{n} |y_i|^{\,p} \right)^{1/p} \right).
\end{aligned}$$

□

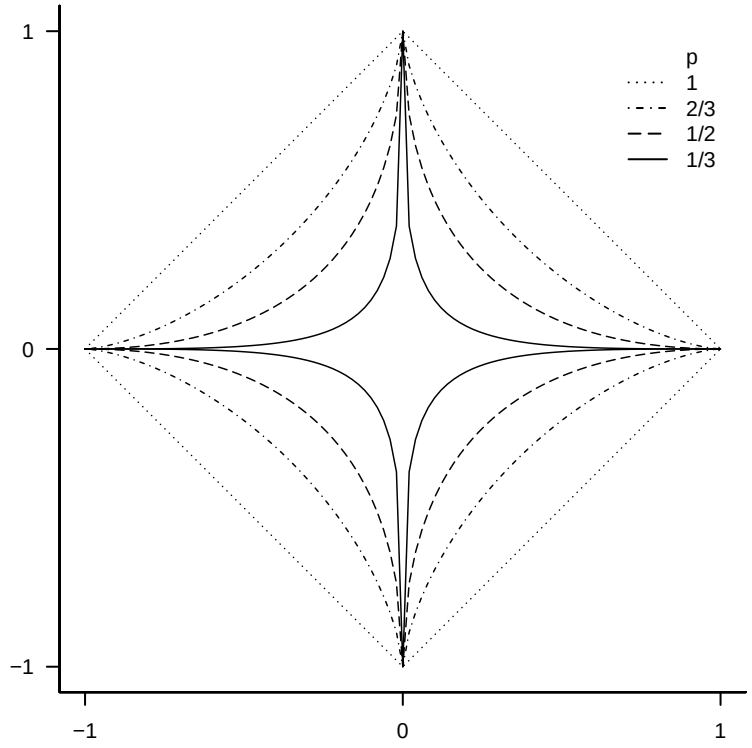

Abbildung 4.2: Einheitskreise von drei Quasinormen mit unterschiedlichen Parametern und der Summennorm

Die Abbildung 4.2 stellt die Einheitskugeln von verschiedenen Quasinormen (mit $p = 1/3$, $1/2$ und $2/3$) und der Summennorm mit $p = 1$ dar.

Eine Quasinorm induziert eine „Metrik“ (die natürlich die Dreiecksungleich nicht erfüllt und daher keine echte Metrik ist). [Agg01a] bezeichnen eine solche Abbildung als *Fractional Distance Metric*:

Definition 4.23. Eine Abbildung

$$\text{fdm}: \ \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_+, \quad (x, y) \mapsto \left(\sum_{i=1}^{n} |x_i - y_i|^{\,p} \right)^{1/p}$$

mit $0 < p < 1$ heißt *Fractional Distance Metric*.

[Agg01a] zeigen, dass die Sätze 4.15, 4.16 und 4.18 sowie Korollar 4.17 und Korollar 4.19 auch für eine Fractional Distance Metric gelten. Um die Rechnungen in den Beweisen zu vereinfachen, beschränken sie p allerdings auf Stammbrüche.

Aus Korollar 4.17 und Korollar 4.19 folgt: Verwendet man eine Fractional Distance Metric, konvergiert die Differenz zwischen dem größten und dem kleinsten Datum mit wachsender Dimension gegen unendlich. Die Experimente in [Agg01a] bestätigen, dass Fractional Distance Metrics zu besseren Resultaten führen als Minkowski-Metriken. Man

sollte daher in praktischen Anwendungen eine Fractional Distance Metric als Alternative in Betracht ziehen.

5 Ausreißererkennung in Unterräumen

[Agg01b] beschreiben einen Algorithmus, der nicht den ganzen Datenraum betrachtet, sondern Unterräume. Dieser Algorithmus sucht nach Ausreißern in Projektionen der Daten.

Die Dichteverhältnisse können bei hochdimensionalen Daten in den einzelnen Attributen oder in Teilmengen der Attribute sehr unterschiedlich sein: Ein Datum, das bei Verwendung aller Attribute nur schwer als Ausreißer erkennbar ist, kann nach einer Projektion auf eine Teilmenge der Attribute eindeutig als Ausreißer erscheinen. Passende Projektionen können daher die Suche nach Ausreißern vereinfachen. Dieser Umstand wird bei praktischen Anwendungen dadurch verstärkt, dass sich Ausreißer oft nur in wenigen Attributen von den normalen Daten unterscheiden.

Beispiel 5.1. Die Abbildungen 5.1 und 5.2 stellen von 150 Schwertlilien Länge und Breite von Kelch- und Kronblatt dar. Das rot markierte Datum könnte ein Ausreißer sein, aber nur wenn man bestimmte Attribute verwendet: Im Unterraum, den die Attribute *Länge Kelchblatt*, *Länge Kronblatt* und *Breite Kronblatt* bilden, ist das Datum kein Ausreißer (Abbildung 5.1 und Zeilen/Spalten 1, 3 und 4 von Abbildung 5.2). Entscheidend ist das Attribut *Breite Kelchblatt* in Kombination mit den Attributen *Länge Kronblatt* und *Breite Kronblatt*. In den von diesen Attributen gebildeten Unterräumen ist das Datum ein Ausreißer (Zeile/Spalte 2 von Abbildung 5.2). Die Daten gehören zum `iris`-Dataset von R [RCT12].

> (For example, there may be a large number of people below the age of 20, and a large number of people with diabetes, but very few with both.) From the perspective of an outlier detection technique a person below the age of 20 with diabetes is a very interesting record; however, it is very difficult to find such a pattern using structured search methods. [Agg01b, S. 40]

Die Schwierigkeit mit strukturierten Verfahren sehen [Agg01b] darin, dass „there are no upward or downward-closed properties in the set of dimensions [...] which are unusually sparse“ [Agg01b, S. 39].

Der von [Agg01b] entwickelte Algorithmus ist daher ein evolutionärer Algorithmus mit folgenden Eigenschaften:

- Man arbeitet nicht mit den einzelnen Daten sondern mit höherdimensionalen Quadern und betrachtet die Anzahl der Daten in den Quadern. Die Quader haben eine niedrigere Dimension als der Datenraum und sind daher Teilmengen von Unterräumen des Datenraums.

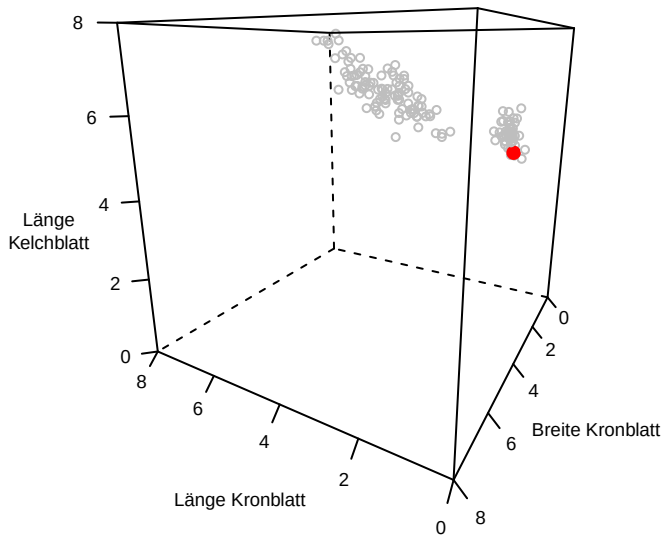

Abbildung 5.1: `iris`-Dataset ohne Attribut *Breite Kelchblatt*

- Die Quader werden erstellt, indem man den Datenraum diskretisiert. Jede Dimension wird in disjunkte Teilmengen (*Streifen*) zerlegt. Die Anzahl der Daten ist in jedem Streifen gleich. Die Streifen (und in Folge die Quader) sind daher unterschiedlich breit.
- Jeder Quader umfasst in einer Dimension nur einen Streifen. Es gibt also keine Quader, die zwei oder mehr Streifen breit sind.
- Die Unterräume entstehen durch Projektionen auf Teilmengen der Attribute.
- Alle Unterräume (und damit auch alle Quader als Teilmengen der Unterräume) haben die gleiche Dimension κ.
- Der Algorithmus sucht nach Quadern, die eine geringe, unterdurchschnittliche, Anzahl der Daten beinhalten.
- Quader ohne Daten werden nicht betrachtet.
- Die Zahl der Streifen pro Dimension, die Anzahl der Quader in der Lösung und die Dimension der Unterräume werden vom Anwender vorgegeben.

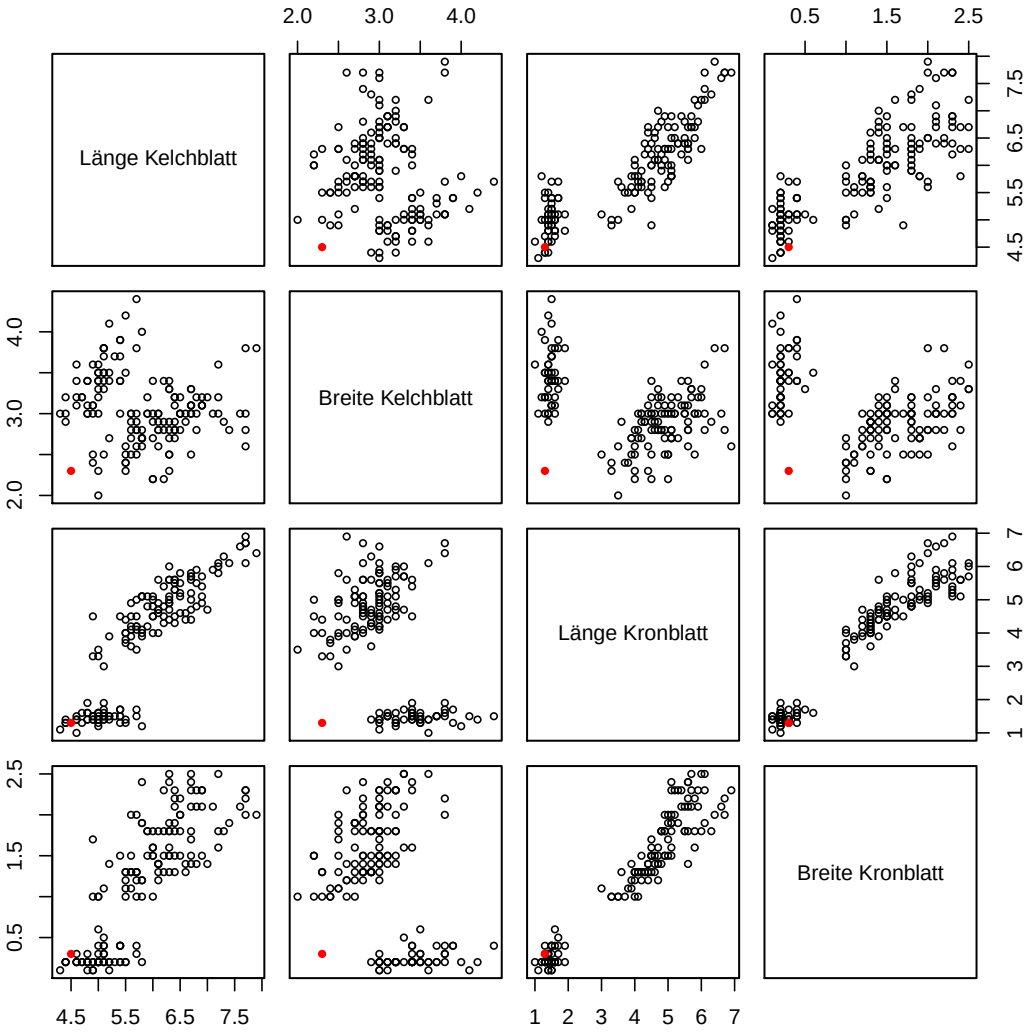

Abbildung 5.2: `iris`-Dataset, alle Attribute

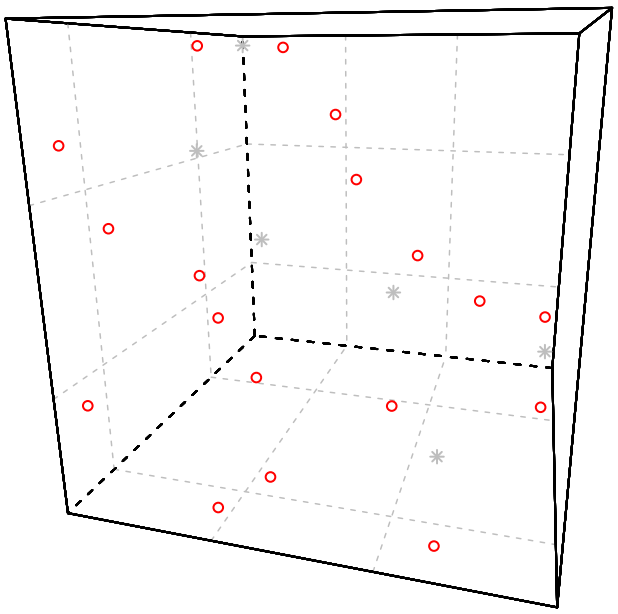

Abbildung 5.3: Konstruktion der Quader. Die grauen Punkte stellen die Daten dar, die roten Punkte sind die Projektionen der Daten.

Ziel ist es, unter den κ-dimensionalen Quadern jene zu finden, die die wenigsten Daten beinhalten, aber nicht leer sind. Das Problem lässt sich exakt mit einem Brute-Force-Algorithmus lösen. Ein Brute-Force-Algorithmus hat allerdings eine exponentielle Laufzeit von $\mathcal{O}(2^n)$ und ist daher offensichtlich nicht geeignet.

Beispiel 5.2. Beachte Abbildung 5.3: Der dreidimensionale Datenraum mit 6 Daten wird auf zweidimensionale Unterräume projeziert. In jeder Dimension werden drei Streifen erzeugt. Die Streifenbreite wurde so gewählt, dass in jedem Streifen zwei Daten (ein Drittel aller Daten) liegen.

Definition 5.3. Im folgenden bezeichnen wir mit ϕ die Anzahl der Streifen pro Dimension. Die Streifen nummerieren wir mit 1 bis ϕ. Um die Schreibweise zu vereinfachen führen wir $f := 1/\phi$ ein. Die Dimension der Quader sei κ, mit $\kappa < n$. Die Anzahl der Daten in einem Quader $\mathcal{C}$ bezeichnen wir mit $|\mathcal{C}|$.

Durch einfache kombinatorische Überlegungen erkennt man, dass es 2^n Unterräume, ϕ^n n-dimensionale Quader und $\binom{n}{k}\phi^\kappa$ κ-dimensionale Quader gibt (man wählt von n Dimensionen κ aus, an jeder der κ Stellen hat man ϕ Streifen zur Auswahl). Für die Quader verwenden wir folgende Schreibweise:

Definition 5.4. Ein Quader $\mathcal{C} \subseteq D$ mit

$$\mathcal{C} = (c_1, c_2, \ldots, c_n), \quad c_i \in \{0, 1, \ldots, \phi\}$$

beinhaltet alle Daten $x = (x_1, x_2, \ldots, x_n) \in D$ mit

$$x_i = \begin{cases} \in \mathbb{R} & \text{falls } \; c_i = 0, \\ \in \text{Streifen } c_i & \text{falls } \; c_i \neq 0. \end{cases}$$

Beispiel 5.5. Sei $n = 4$, $\phi = 10$, $\kappa = 2$. Der Quader $\mathcal{C} = (0, 3, 8, 0)$ ist eine Teilmenge des zweidimensionalen Unterraums, der durch eine Projektion des Datenraums auf die 2. und 3. Koordinate erzeugt wird. Die Elemente des Quaders $\mathcal{C}$ sind alle Punkte, deren zweite Koordinate im dritten Streifen und deren dritte Koordinate im achten Streifen liegt. Die erste und vierte Koordinate ist unerheblich.

5.1 Der Sparsity Coefficient $S(\mathcal{C})$

Der Algorithmus sucht nach Quadern, die ungewöhnlich wenige Daten beinhalten. Als Maß dafür definieren [Agg01b] den *Sparsity Coefficient*:

Definition 5.6. Der *Sparsity Coefficient* $S(\mathcal{C})$ eines k-dimensionales Quaders ist definiert durch

$$S(\mathcal{C}) := \begin{cases} \frac{|\mathcal{C}| - N \cdot f^\kappa}{\sqrt{N \cdot f^\kappa \cdot (1 - f^\kappa)}} & \text{falls } \; |\mathcal{C}| > 0, \\ \infty & \text{falls } \; |\mathcal{C}| = 0. \end{cases} \tag{5.1}$$

Hinter dieser Definition stehen folgende Überlegungen: Wir nehmen an, dass die Daten gleichverteilt sind. Die Wahrscheinlichkeit, dass ein Datum in einem bestimmten Streifen liegt ist f. Die Wahrscheinlichkeit, dass ein Datum in einem bestimmten Quader liegt ist folglich f^κ. Die Wahrscheinlichkeiten folgen einer Bernoulli-Verteilung, der Erwartungswert der Anzahl der Daten in einem Quader ist daher $N \cdot f^\kappa$, bei einer Standardabweichung von $\sqrt{N \cdot f^\kappa \cdot (1 - f^\kappa)}$. Beim Sparsity Coefficient handelt es sich also um den z-Wert von $|\mathcal{C}|$.

Je kleiner der Sparsity Coefficient ist, umso weniger Daten befinden sich im Quader. Ist der Sparsity Coefficient negativ, befinden sich im Quader unterdurchschnittlich viele Daten.

Während des Durchlaufs des Algorithmus bleiben N, f und κ konstant. Man kann daher statt des Sparsity Coefficient auch $|\mathcal{C}|$ verwenden. Der Sparsity Coefficient liefert allerdings eine Aussage über die Qualtität des Endergebnisses und man kann Endergebnisse vergleichen, die mit unterschiedlichen Parametern gewonnen wurden. Die Annahme, dass die Daten gleichverteilt sind, ist üblicherweise jedoch nicht haltbar; das relativiert die Aussage des Sparsity Coefficient etwas.

5.2 Evolutionärer Algorithmus zur Ausreißererkennung

Der Pseudocode des Algorithmus steht in den Algorithmen 1–6. Der Aufbau ist der eines evolutionären Algorithmus: Aus der Menge der vorläufig erhaltenen Lösungen wählt der Algorithmus die besten Lösungen aus (*Selektion*) und arbeitet mit dieser Auswahl weiter. Je zwei Lösungen werden zu zwei neuen Lösungen kombiniert (*Rekombination* und *Crossover*). Dabei soll zumindest eine der beiden neuen Lösungen besser sein als die beiden vorhandenen Lösungen. Der Schritt *Mutation* verändert die Lösungen geringfügig nach einem Zufallsprinzip. Während der Algorithmus läuft, werden die Lösungen immer ähnlicher. Der Algorithmus bricht ab, wenn ein bestimmter Anteil der Quader in der Lösungsmenge gleich ist.

Selektion Die Funktion Selection ordnet die Quader aufsteigend nach dem Sparsity Coefficient (der Quader mit dem kleinsten Sparsity Coefficient zuerst). Mit $\text{rang}(\mathcal{C})$ bezeichnen wir die Position des Quaders in dieser Liste, mit l die Anzahl der Quader in der Liste. Jedem Quader wird eine Wahrscheinlichkeit von $(l-\text{rang}(\mathcal{C}))\cdot 2/(l(l-1))$ zugeordnet ($l-\text{rang}(\mathcal{C})$ nimmt die Werte 0 bis $l-1$ an und $\sum_{i=0}^{l-1} i = l(l-1)/2$). Mit dieser Wahrscheinlichkeitsverteilung werden l Quader mit Zurücklegen gezogen (d. h. ein Quader kann öfter gezogen werden). Beachte, dass der Quader mit dem größten Sparsity Coefficient nie gezogen wird.

Crossover Die Quader werden paarweise zusammengefasst. Aus jedem Elternpaar $\mathcal{C}^\dagger$ und $\mathcal{C}^{\dagger\dagger}$ werden zwei neue Kindquader $\mathcal{C}^*$ und $\mathcal{C}^{**}$ erstellt, dabei muss die Dimension der Quader (d. h. die Anzahl der $c_i \neq 0$) gleich bleiben. Außerdem soll zumindest einer der beiden Kindquader von höherer Qualität sein als die beiden Elternquader. Wir betrachten die Attribute der beiden Elternquader $\mathcal{C}^\dagger = (c_1^\dagger, \ldots, c_n^\dagger)$ und $\mathcal{C}^{\dagger\dagger} = (c_1^{\dagger\dagger}, \ldots, c_n^{\dagger\dagger})$. Drei Fälle sind möglich:

1. Eine Dimension wird bei beiden Elternquader nicht verwendet: $c_i^\dagger = c_i^{\dagger\dagger} = 0$. Dann wird diese Dimension auch bei den beiden Kindquader gleich 0 gesetzt: $c_i^* = c_i^{**} = 0$.
2. Eine Dimension wird bei beiden Elternquader verwendet: $c_i^\dagger \neq 0$ und $c_i^{\dagger\dagger} \neq 0$. Sei $\widehat{\kappa} \leq \kappa$ die Anzahl der Dimensionen, bei denen diese Situation eintritt. Es gibt $2^{\widehat{\kappa}}$ Möglichkeiten diese Dimensionen neu zu kombinieren. Die Funktion berechnet von allen $2^{\widehat{\kappa}}$ Quader den Sparsity Coefficient, dabei setzt die Funktion alle anderen Dimensionen auf 0. Man wählt dann den besten Quader – mit dem niedrigsten Sparsity Coefficient – aus. Wir bezeichnen diesen Quader mit $\mathcal{C}^*$. Die Laufzeit beträgt $\mathcal{O}(2^{\widehat{\kappa}})$. Üblicherweise ist $\widehat{\kappa} \ll d$, daher ist die Laufzeit nicht zu lang.
   Beispiel 5.7. Betrachte die Elternquader $\mathcal{C}\dagger = (1,2,1,0,1)$ und $\mathcal{C}^{\dagger\dagger} = (3,4,0,0,1)$. Bei den Koordinaten 3–5 liegt Fall 1 oder Fall 3 vor. Bei den Koordinaten 1 und 2 haben wir Fall 2 und es gibt $2^2 = 4$ Möglichkeiten, diese beiden Koordinaten neu zu kombinieren: $(1,2,0,0,0)$, $(1,4,0,0,0)$, $(3,2,0,0,0)$ und $(3,4,0,0,0)$. Von diesen vier zweidimensionalen Quadern wählt der Algorithmus jenen mit dem niedrigsten Sparsity Coefficient aus.

3. Eine Dimension wird bei einem Elternquader verwendet, beim anderen nicht. Jeder Quader hat $\kappa - \widehat{\kappa}$ solcher Dimensionen, insgesamt gibt es also $2(\kappa - \widehat{\kappa})$ solcher Streifen. Dem Kindquader $\mathcal{C}^*$ aus Fall 2 fügen wir in $\kappa - \widehat{\kappa}$ Schritten $\kappa - \widehat{\kappa}$ Dimensionen hinzu. Wir fügen in jedem Schritt jene Dimension hinzu, die zu einem minimalen Sparsity Coefficient von $\mathcal{C}^*$ führt.

Der zweite Kindquader $\mathcal{C}^{**}$ bildet das Komplement zu $\mathcal{C}^*$: Wir verwenden für $\mathcal{C}^{**}$ jene Dimensionen bzw. Streifen, die wir nicht für $\mathcal{C}^*$ verwendet haben.

Mutation Die Funktion Mutation verändert mit einer Wahrscheinlichkeit p_1 den Wert eines Attributs c_i mit Wert 0 in einen Wert zwischen 1 und ϕ und zusätzlich den Wert eines Attributs $c_j, j \neq i$ mit Wert ungleich 0 in 0. Mit einer Wahrscheinlichkeit von p_2 ändert die Funktion bei einem Attribut $c_k, k \neq i, j$ mit einem Wert ungleich 0 den Wert in einen anderen Wert ungleich 0.

Abbruchkriterium Die Quader in der Lösungsmenge S werden sich mit der Zeit immer ähnlicher. [Agg01b] verwenden das De-Jong-Kriterium für die Entscheidung, ob der Algorithmus beendet werden soll. Wir betrachten die i-te Koordinate c_i der Quader in S. Sobald bei 95 % der Quader die i-ten Koordinaten c_i den selben Wert angenommen haben, haben diese Koordinaten und damit die i-te Dimension ihren Grenzwert erreicht (der Begriff Grenzwert ist hier nicht im Sinn der Analysis zu verstehen). Sobald 95 % der Dimensionen ihren Grenzwert erreicht haben, ist das Abbruchkriterium erfüllt.

Algorithmus 1 : Evolutionary Outlier Detection in Subspaces

Input : Daten D, Anzahl von Lösungen m, Dimension der Lösungen κ, Wahrscheinlichkeiten für Mutationen p_1, p_2

Output : Lösungsmenge $L = \{\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_m\}$

```
Erzeuge zufällige Startmenge S von Lösungen, |S| = m
L ← {}
while ! Abbruchkriterium(S) do
    S ← Selection(S)
    S ← Crossover(S)
    S ← Mutation(S, p1, p2)
    L ← m Quader C ∈ L ∪ S mit kleinstem Sparsity Coefficient S(C)
end
return L
```

5.3 Wahl der Parameter κ und ϕ

Die Parameter κ und ϕ bestimmen die Anzahl der Quader. Betrachtet man einen fixen κ-dimensionalen Unterraum gibt es ϕ^κ κ-dimensionale Quader in diesem Unterraum.

Algorithmus 2 : Abbruchkriterium

Input : aktuelle Lösungsmenge S

$numDimLimit \leftarrow 0$
for $i \leftarrow 1$ **to** d **do**
  if *95 % der* c_i *der* $\mathcal{C} \in S$ *sind gleich* **then**
    $numDimLimit \leftarrow numDimLimit + 1$
  end
end
return $numDimLimit > 0.95 \cdot \dim(D)$

Algorithmus 3 : Selection

Input : aktuelle Lösungsmenge S
Output : Selektion von Quadern $R \subseteq S$

$S \leftarrow \text{sort}(S)$ `// aufsteigend nach` *`Sparsity Coefficient`*
$R \leftarrow \{\}$ `// Multimenge`
for $i \leftarrow 1$ **to** $|S|$ **do**
  Wähle ein $s \in S$ mit Wahrscheinlichkeit $\mathbb{P}[s = s_i] = |S| - \text{rang}(s_i)$
  $R \leftarrow R \cup \{s\}$
end
return R

Algorithmus 4 : Crossover

Input : aktuelle Lösungsmenge S
Output : Aus S neu kombinierte Quader

$R \leftarrow \{\{s_i, s_{i+1}\} \mid s_i,\ s_{i+1} \in S,\ i = 1, 3, \ldots, m-1\}$ `// paarweise kombinieren`
foreach $p \in R$ **do**
  $p \leftarrow \text{Recombine}(p)$
end
return R

Algorithmus 5 : Recombine

Input : Elternquader $\mathcal{C}^{\dagger} = (c_1^{\dagger}, c_2^{\dagger}, \ldots, c_n^{\dagger})$, $\mathcal{C}^{\dagger\dagger} = (c_1^{\dagger\dagger}, c_2^{\dagger\dagger}, \ldots, c_n^{\dagger\dagger})$
Output : Kindquader $\mathcal{C}^{*}$, $\mathcal{C}^{**}$

$k \leftarrow \left|c_1^{\dagger} \neq 0\right|$ `// Dimension der Elternquader`
$Q \leftarrow \{i \mid c_i^{\dagger} \neq 0 \,\dot{\vee}\, c_i^{\dagger\dagger} \neq 0\}$ `//` $\dot{\vee} = \text{XOR}$
$R \leftarrow \{i \mid c_i^{\dagger} \neq 0 \wedge c_i^{\dagger\dagger} \neq 0\}$
$\mathcal{C}^{*} \leftarrow$ Jene Lösung aus den $2^{|R|}$ Möglichkeiten die Dimensionen aus R zu kombinieren mit minimalem Sparsity Coefficient.
while $|c_1^{*} \neq 0| < \kappa$ **do**
  Füge jenes Attribut aus Q zu $\mathcal{C}^{*}$ hinzu, welches $S(\mathcal{C}^{*})$ minimiert.
end
$\mathcal{C}^{**} \leftarrow \text{complement}\,(\mathcal{C}^{*})$
return $\mathcal{C}^{*}$, $\mathcal{C}^{**}$

Algorithmus 6 : Mutation

Input : aktuelle Lösungsmenge S, Wahrscheinlichkeiten für Mutationen p_1, p_2
Output : S mit mutierten Quadern

foreach $\mathcal{C} = (c_1, c_2, \ldots, c_n) \in S$ **do**
  if *Münzwurf mit Wahrscheinlichkeit* p_1 *ist Erfolg* **then**
    $i \leftarrow$ zufällig gewählter Index von $\mathcal{C}$ mit $c_i = 0$.
    $j \leftarrow$ zufällig gewählter Index von $\mathcal{C}$ mit $c_j \neq 0$.
    $c_i \leftarrow$ zufälliger Wert aus $\{1, 2, \ldots, \phi\}$
    $c_j \leftarrow 0$
  end
  if *Münzwurf mit Wahrscheinlichkeit* p_2 *ist Erfolg* **then**
    $i \leftarrow$ zufällig gewählter Index von $\mathcal{C}$ mit $c_i \neq 0$.
    $c_i \leftarrow$ zufälliger Wert aus $\{1, 2, \ldots, \phi\}$
  end
end
return S

Wählt man die Parameter zu groß, erhält man zu viele Quader. Die meisten Quader beinhalten dann nur viele Daten und es gibt keine Quader mit unterdurchschnittlich vielen Daten.

Beispiel 5.8. Sei $\phi = 10$, $\kappa = 4$, $N = 10000$. In einem beliebig gewählten Unterraum gibt es 10000 Quader. Unter der Annahme, dass die 10000 Daten gleichverteilt sind, ist der Erwartungswert für die Anzahl an Daten in einem Quader lediglich 1. Das bedeutet, es gibt keine nicht-leeren Quader mit unterdurchschnittlich vielen Daten.

Hat man andererseits zu wenig Quader, können Ausreißer öfter in einem Quader mit vielen normalen Daten liegen. Diese Ausreißer werden dann nicht gefunden.

[Agg01b] empfehlen folgendes Verfahren um die Parameter festzulegen: Im ersten Schritt wählt man einen Wert für ϕ. Der Wert sollte so groß sein, dass man die Daten eines Streifens gut interpretieren kann. Anschaulich bedeutet das, die Streifen sollen nicht zu breit sein.

Im zweiten Schritt berechnet man κ. Der Sparsity Coefficient eines leeren Quaders beträgt (Formel 5.1)

$$S(\varnothing) = -\sqrt{\frac{N}{\phi^\kappa - 1}}. \tag{5.2}$$

Wir approximieren $S(\mathcal{C})$ mit der Standardnormalverteilung. Das 0.1 %-Quartil beträgt ungefähr -3. Wir setzen in 5.2 ein und formen um:

$$\kappa = \left\lfloor \log_\phi \left(\frac{N}{3^2} + 1 \right) \right\rfloor. \tag{5.3}$$

Dieses Verfahren ist nur eine Heuristik. Man muss den Wert für ϕ schätzen (mithilfe von Erfahrung, Expertenwissen, ...) und es ist nicht garantiert, dass die Standardnormalverteilung $S(\mathcal{C})$ hinreichend gut approximiert.

5.4 Laufzeitanalyse

Beschäftigen wir uns zunächst mit den Algorithmen 2 bis 6: Algorithmus 2 hat eine Laufzeit von $n \cdot |S|$. Bei Algorithmus 3 beträgt die Laufzeit

$$N \cdot |S| + |S| \cdot \log |S| + |S|$$

für die Berechnung der Sparsity Coefficients und die Sortierung in Zeile 1 sowie für die Schleife in Zeile 3.

Bei Algorithmus 5 ist der Wert von $|R|$ und $|Q|$ entscheidend. Bei beiden Quadern sind von n Koordinaten κ ungleich 0. Wir treffen die Annahme, dass die Wahrscheinlichkeit dass eine Koordinaten c_i ungleich 0 ist, gleich κ/n ist. Die Wahrscheinlichkeit, dass bei beiden Elternquader die i-te Koordinate ungleich 0 ist, ist dann gleich $(\kappa/n)^2$. Somit gilt:

$$\begin{aligned} \mathbb{E}\left[|R|\right] &= n\,(\kappa/n)^2 = \kappa^2/n, \\ \mathbb{E}\left[|Q|\right] &= \kappa - \kappa^2/n. \end{aligned}$$

Algorithmus 2	$n \cdot \lvert S \rvert$
Algorithmus 3	$N \cdot \lvert S \rvert + \lvert S \rvert \cdot \log \lvert S \rvert + \lvert S \rvert$
Algorithmus 4	$\lvert S \rvert \cdot N \cdot \mathcal{O}(n^2)$
Algorithmus 5	$N \cdot \mathcal{O}(n^2)$
Algorithmus 6	$\lvert S \rvert$

Tabelle 5.1: Laufzeit der einzelnen Teile des Algorithmus 1

Daraus folgt eine Average-Case-Laufzeit von Zeile 4 von $N \cdot \kappa^2/n$ und von der Schleife in Zeile 5 von $N \cdot \mathbb{E}\left[\lvert Q \rvert\right]^2$. Bezeichnen wir das Verhältnis κ/n mit r und nehmen an, dass dieses Verhältnis mit wachsendem n konstant bleibt, dann gilt $N \cdot \mathbb{E}\left[\lvert Q \rvert\right]^2 = N(rn(1 - r))^2 = N \cdot \mathcal{O}(n^2)$.

Für Algorithmus 4 erhalten wir eine Laufzeit von $\lvert S \rvert \cdot N \cdot \mathcal{O}(n^2)$ und für Algorithmus 6 eine Laufzeit von $\lvert S \rvert$.

Tabelle 5.1 fasst alle Ergebnisse zusammen, die für die Laufzeit von Algorithmus 1 von Bedeutung sind.

Benennen wir die Anzahl der Schleifendurchgänge in Zeile 3 des Algorithmus 1 mit t, gehört die Laufzeit in die Klasse $\mathcal{O}(tNn^2 \lvert S \rvert \log \lvert S \rvert)$. Es existiert keine Formel um t zu berechnen (oder zumindest abzuschätzen). Die Experimente in [Agg01b] zeigen, dass die Berechnung der Sparsity Coefficients den größten Einfluss auf die Laufzeit hat. Diese Berechnung hat für $\lvert S \rvert$ Quader eine Laufzeit von $\mathcal{O}(N \lvert S \rvert)$. Deshalb ist die Anzahl der Daten N der entscheidende Faktor.

6 Ausreißererkennung mit Hilbert-Kurven

Die grundlegende Idee dieses Algorithmus ist die gleiche wie die des Algorithmus von [Agg01b]: Der hochdimensionale Datenraum wird auf einen Raum mit geringerer Dimension abgebildet. [Ang05] wählen dazu einen eindimensionalen Raum: Eine Hilbert-Kurve, die durch den Datenraum läuft.

Unsere Daten sind Elemente des Einheitswürfels, wir legen eine Hilbert-Kurve durch diesen Einheitswürfel. Die Daten bilden wir auf die Hilbert-Kurve ab und erhalten dadurch eine Reihung der Daten entlang der Kurve. Diese Reihung induziert eine Pseudometrik: Der Abstand zwischen Daten, die Nachbarn entlang der Kurve sind, ist gering. Entsprechend ist der Abstand zwischen nicht benachbarten Daten groß.

Ist der Abstand zweier Daten entlang der Hilbert-Kurve gering, ist auch der Abstand laut Minkowski-Metrik gering (Abbildung 6.1, Daten x und y). Die Umkehrung muss nicht gelten: Der Abstand entlang der Hilbert-Kurve zwischen den Daten y und z in Abbildung 6.1 ist groß, da das eine Datum am Anfang, das andere Datum am Ende der Hilbert-Kurve liegt. Der Abstand laut Minkowski-Metrik dagegen ist gering. Die Pseudometrik liefert also nur eine Abschätzung nach oben für die Minkowski-Metrik.

Um dieses Problem zu lösen, legen wir mehrere unterschiedliche Hilbert-Kurven durch den Datenraum. Jede Hilbert-Kurve verbessert die Abschätzung. Sobald bei allen Daten, die als Ausreißer in Frage kommen, die Abschätzung hinreichend gut ist, bricht der Algorithmus ab.

6.1 Hilbert-Kurven

Eine Hilbert-Kurve ist der Grenzwert der in Abbildung 6.2 dargestellten Folge von Kurven. Jede Kurve ist Bild einer Abbildung $[0,1] \to E^2$ vom Einheitsintervall in das Einheitsquadrat.

Diese Folge kann auch durch Gleichungen beschrieben werden:

$$h_0 : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \frac{1}{2}\begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 \\ 0 \end{pmatrix} \qquad h_1 : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

$$h_2 : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \frac{1}{2}\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 1 \\ 1 \end{pmatrix} \qquad h_3 : \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \mapsto \frac{1}{2}\begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \frac{1}{2}\begin{pmatrix} 2 \\ 1 \end{pmatrix}$$

Eine Hilbert-Kurve ist ein Beispiel für eine überall stetige, aber nirgends differenzierbare Kurve. Weiters ist das Jordan-Maß einer Hilbert-Kurve größer 0 (*space-filling curve*).

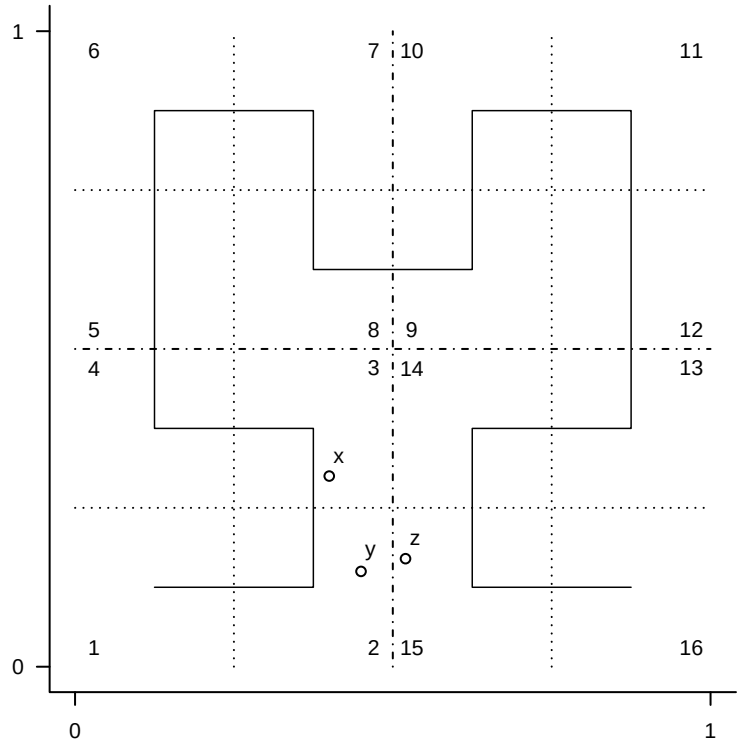

Abbildung 6.1: Hilbert-Kurve zweiter Ordnung

Ähnlich kann man Hilbert-Kurven auch in höherdimensionalen Räumen definieren. Der dreidimensionale Fall ist z. B. in [Sag94] beschrieben. Dort stehen auch Beweise für alle anderen in diesem Abschnitt gemachten Aussagen.

Der Algorithmus zur Ausreißererkennung arbeitet allerdings nicht mit der Hilbert-Kurve (dem Grenzwert der Folge) sondern mit einer Kurve der Folge.

Definition 6.1. Wir bezeichnen das κ-te Folgenglied als Hilbert-Kurve κ-ter Ordnung.

Eine Hilbert-Kurve κ-ter Ordnung im n-dimensionalen Raum zerlegt den n-dimensionalen Einheitswürfel in $2^{n\kappa}$ disjunkte Teilwürfel $\mathcal{C}_1, \ldots, \mathcal{C}_{2^{n\kappa}}$ mit

$$\mathcal{C}_1 \cup \mathcal{C}_2 \cup \ldots \cup \mathcal{C}_{2^{n\kappa}} = E^n$$

Das Urbild eines jeden Teilwürfels ist ein $1/2^{n\kappa}$ langes Intervall des Einheitsintervalls $[0, 1] \subseteq \mathbb{R}$. Die Reihenfolge dieser Intervalle bringt die Teilwürfel in eine wohldefinierte Ordnung. Das Urbild des Teilwürfels $\mathcal{C}_1$ ist das Intervall $[0, 1/2^{n\kappa})$, das Urbild des Teilwürfels $\mathcal{C}_2$ ist das Intervall $[1/2^{n\kappa}, 2/2^{n\kappa})$, usw.

Für den Algorithmus ist die Position eines bestimmten Teilwürfels innerhalb dieser Ordnung von Bedeutung. Unter *http://www.tddft.org/svn/octopus/trunk/src/grid/hilbert.c* findet man ein auf Überlegungen von John Skilling basierendes Programm, um diese Position zu berechnen.

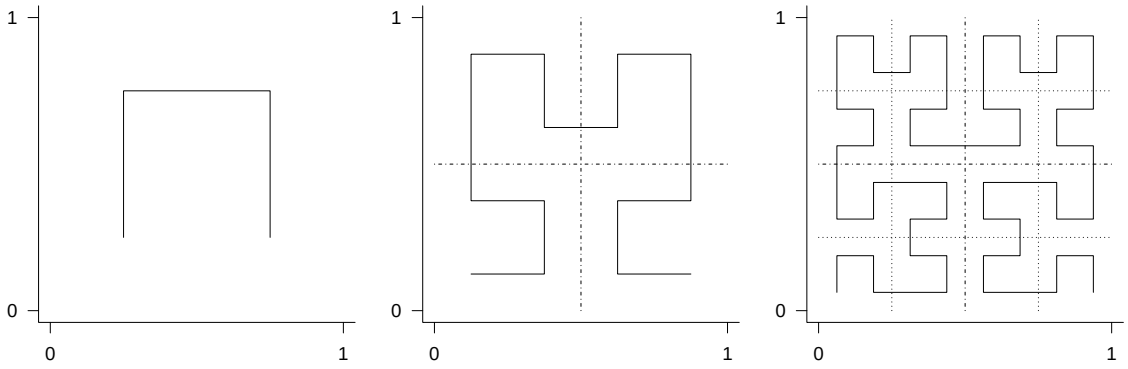

Abbildung 6.2: Die ersten drei Folgenglieder einer Hilbert-Kurven-Konstruktion

6.2 Benötigte Definitionen und Formeln

Als Datenraum dient uns in diesem Kapitel immer der n-dimensionale Einheitswürfel $E^n \subseteq \mathbb{R}^n$.

Die Definition von *Ausreißer*, die in diesem Kapitel verwendet wird, ist ähnlich der Definition 2.8 (Seite 7). Wir definieren das Gewicht eines Datums und passen die Bedeutung von $D_{\mathfrak{n}}^k$ entsprechend an:

Definition 6.2. Das *Gewicht* w^k eines Datums q ist gleich der Summe der Abstände zu den nächsten k Nachbarn:

$$w^k : \mathbb{R}^n \to \mathbb{R}_+, \quad q \mapsto \sum_{\left\{\substack{x \in D: \\ \mathrm{d}_p(q,x) < d^k(q)}\right\}} \mathrm{d}_p(q,x).$$

Definition 6.3. Ein Datum q ist ein $W_{\mathfrak{n}}^k$-*Ausreißer* wenn es maximal $\mathfrak{n}-1$ Daten $x \in D$ gibt mit $w^k(q) < w^k(x)$.

Definition 6.4. Durch unseren Datenraum E^n läuft eine Hilbert-Kurve κ-ter Ordnung. Dann bezeichnen wir mit $\mathcal{C}_1, \ldots, \mathcal{C}_{2^{n\kappa}}$ die durch diese Hilbert-Kurve induzierten halboffenen, disjunkten und geordneten Teilwürfel. Die Ordnung eines Teilwürfels ist gleich der Ordnung der Hilbert-Kurve, also κ (Abbildung 6.3). Die Seitenlänge eines Teilwürfels beträgt $r = 2^{-\kappa}$.

Als Hilbert-Wert eines Datums definieren wir die Position des Teilwürfels, in dem sich das Datum befindet:

Definition 6.5. Seien $\mathcal{C}_1, \ldots, \mathcal{C}_{2^{n\kappa}}$ die durch eine Hilbert-Kurve κ-ter Ordnung induzierten n-dimensionalen Teilwürfel. Der Hilbert-Wert h eines Datums ist die Position des Teilwürfels, in dem sich das Datum befindet:

$$\begin{aligned} h : E^n &\to [1, \ldots, 2^{n\kappa}] \subseteq \mathbb{N}, \\ x &\mapsto j \quad \text{für} \quad x \in \mathcal{C}_j. \end{aligned}$$

Mithilfe des Hilbert-Werts erhalten wir eine Ordnung $\leq_h$ auf dem Datenraum:

Definition 6.6.

$$x \leq_h y \iff h(x) \leq h(y).$$

Diese Ordnung ist reflexiv, transitiv und total. Da die Ordnung total ist, bringt sie die Daten in eine Kette, allerdings mit der Einschränkung, dass bei Daten mit dem gleichen Hilbert-Wert (also bei Daten im selben Teilwürfel) die Reihenfolge nicht festgelegt ist. Wird der Algorithmus praktisch angewendet, kann die Reihenfolge in diesem Fall beliebig, aber fest, gewählt werden.

Definition 6.7. Sei x ein Datum. Mit $h_{pred}(x, m)$ bezeichnen wir den m-ten Vorgänger von x entlang dieser Kette und mit $h_{succ}(x, m)$ den m-ten Nachfolger von x entlang dieser Kette.

Definition 6.8. Seien x und y zwei Daten. MinCube (x, y) ist die Seitenlänge des kleinsten Teilwürfels, welcher x und y beinhaltet. MaxCube (x, y) ist die Seitenlänge des größten Teilwürfels, welcher x, aber nicht y beinhaltet (Abbildung 6.4).

Definition 6.9. Seien x, y und z Daten, r die Seitenlänge eines Teilwürfels und p der Exponent der verwendeten Minkowski-Metrik.

$$\begin{aligned}
\text{MinDist} &: E^n \times [0,1] \to [0,1] \\
&\quad (x, r) \mapsto \min_{i \in \{1,\ldots,n\}} \left(\min\left(x_i \bmod r, r - x_i \bmod r\right)\right). \\
\text{MaxDist} &: E^n \times [0,1] \to [0,1] \\
&\quad (x, r) \mapsto \begin{cases} \left(\sum_{i=1}^{n} \left(\max\left(x_i \bmod r, r - x_i \bmod r\right)\right)^p\right)^{1/p} & \text{für} \quad 1 \leq p < \infty \\ \max_{i \in \{1,\ldots,n\}} \left(\max\left(x_i \bmod r, r - x_i \bmod r\right)\right) & \text{für} \quad p = \infty \end{cases}. \\
\text{BoxRadius} &: E^n \times E^n \times E^n \to [0,1] \\
&\quad (x, y, z) \mapsto \text{MinDist}\left(x,\ \min\left(\text{MaxCube}\left(x, y\right), \text{MaxCube}\left(x, z\right)\right)\right).
\end{aligned}$$

Beispiel 6.10. Beachte Abbildung 6.5: MinDist(x, r) ist der Abstand des Datums x zur nächsten Seite des Teilwürfels mit Seitenlänge r, in der sich das Datum befindet. MaxDist(x, r) ist der Radius des Umkreises, der diesen Teilwürfels umfasst. BoxRadius(x, y, z) ist die MinDist in Bezug auf jenen größtmöglichen Teilwürfel, in dem sich x, aber weder y noch z befinden (Abbildung 6.6).

Lemma 6.11. *Seien q ein Datum, a und b zwei natürliche Zahlen. Die Menge I beinhaltet die a Vorgänger und die b Nachfolger von q:*

$$I := \{h_{pred}(q, a), \ldots, h_{pred}(q, 1), h_{succ}(q, 1), \ldots, h_{succ}(q, b)\}.$$

Weiters seien

$$r := \text{BoxRadius}(q, h_{pred}(q, a + 1), h_{succ}(q, b + 1))$$

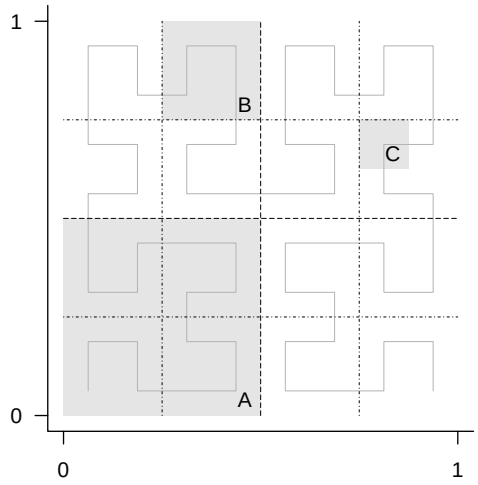

Abbildung 6.3: Drei Teilwürfel mit den Ordnungen 1 (A), 2 (B) und 3 (C).

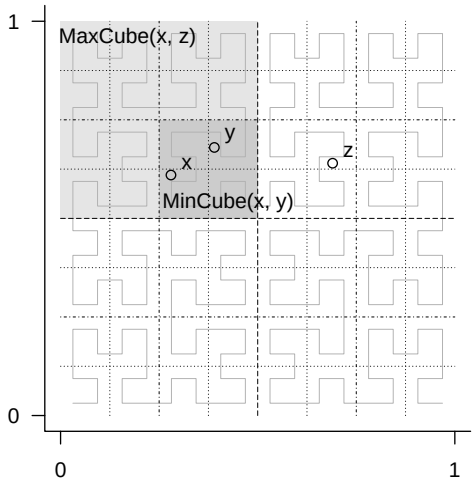

Abbildung 6.4: Teilwürfel bei der Berechnung von MaxCube und MinCube

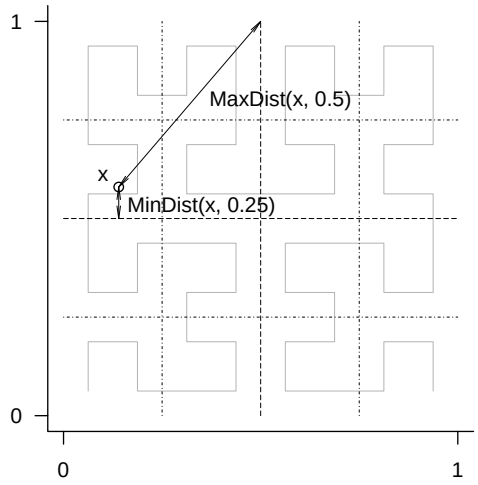

Abbildung 6.5: Berechnung von MaxDist und MinDist

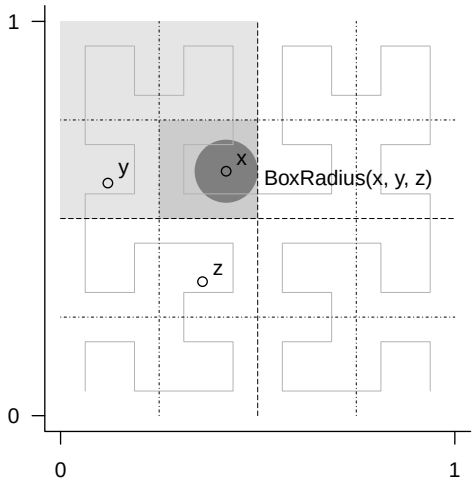

Abbildung 6.6: Berechnung von BoxRadius

und S der Durchschnitt aus I und jenen Daten, deren Abstand zu q (laut Minkowski-Metrik) kleiner oder gleich als r ist:

$$S := I \cap \{x \in D \mid \mathrm{d}_p(q, x) \leq r\}.$$

Dann gilt: Die Daten in S sind die tatsächlichen $|S|$ nächsten Nachbarn von q.

Beweis. Wir betrachten zunächst die beiden Teilwürfel $\mathrm{MaxCube}(q, h_{pred}(q, a+1))$ und $\mathrm{MaxCube}(q, h_{succ}(q, b+1))$. Den kleineren dieser beiden Teilwürfel bezeichnen wir mit

$\mathcal{C}$ (beachte: Einer der Teilwürfel ist immer Teilmenge des anderen Teilwürfels). Der Abstand von q zur nächsten Kante von $\mathcal{C}$ ist per Definition gleich r.

Sei x ein Datum, das zu den $|S|$ nächsten Nachbarn von q gehört. Dann ist der Abstand zu q kleiner gleich r. Die Kugel um q mit Radius r ist eine Teilmenge des Teilwürfels $\mathcal{C}$, nach Definition der Funktion BoxRadius. Das Datum x ist also ein Element von $\mathcal{C}$.

Aus der Stetigkeit der Hilbert-Kurve k-ter Ordnung folgt, dass die Hilbert-Kurve k-ter Ordnung an genau einer Stelle in den Teilwürfel $\mathcal{C}$ eintritt, ihn dann komplett in einem Zug durchläuft und abschließend an genau einer Stelle austritt. Daraus folgt, dass für alle Daten y, die in $\mathcal{C}$ liegen, gilt:

$$h_{pred}(q, a+1) \ <_h \ y \ <_h h_{succ}(q, b+1).$$

Die scharfen Ungleichungen gelten, weil $h_{pred}(q, b+1)$ und $h_{succ}(q, b+1)$ laut Definition von MaxCube sicher nicht in $\mathcal{C}$ liegen.

Umgekehrt ist die Menge I aber eine Obermenge der Daten z, für die gilt:

$$h_{pred}(q, a+1) \ <_h z \ <_h h_{succ}(q, b+1).$$

Daraus folgt:

$$y \in \mathcal{C} \Rightarrow y \in I.$$

Diese Implikation gilt auch für x, das damit auch ein Element von S ist. Siehe auch Abbildung 6.7. □

Korollar 6.12. *Der Abstand von x zum $(|S|+1)$-nächsten Nachbarn von x ist größer r:*

$$\mathrm{d}_p(q, d^{|S|+1}(q)) > r.$$

Die letzte Definition des Abschnitts vereinfacht die Schreibweise. Die Abbildung $V^{(j)}$ verschiebt ein Datum entlang der Hauptdiagonale. Sollte das Datum auf einer Seite aus dem Einheitswürfel hinausgeschoben werden, kommt es auf der anderen Seite wieder hinein:

Definition 6.13. Sei $x \in E^n$ ein Datum des n-dimensionalen Datenraums. Mit der $V^{(j)}$ bezeichnen wir die Transformation

$$\begin{aligned} V^{(j)} : E^n &\to E^n \\ x &\mapsto \left(x + \left(\frac{j}{n+1}, \ldots, \frac{j}{n+1}\right)\right) \bmod 1. \end{aligned}$$

6.3 Beschreibung des Algorithums

Der Algorithmus besteht neben dem Hauptprogramm (Pseudocode siehe Algorithmus 7) aus zwei Funktionen (Algorithmen 8 und 9). Die wichtigsten Variablen sind in Tabelle 6.1 beschrieben. Teilweise handelt es sich dabei um globale Variablen.

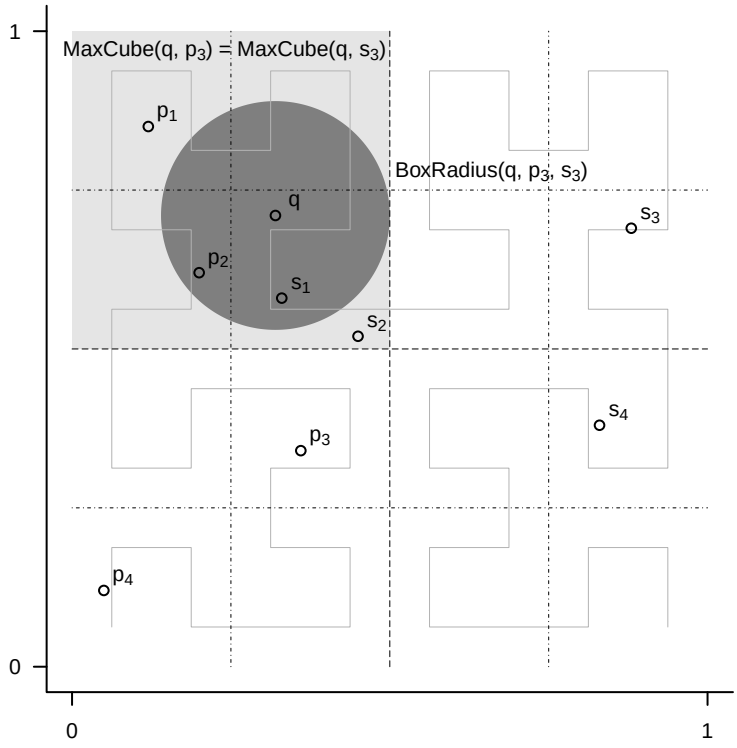

Abbildung 6.7: Anwendung von Lemma 6.11 mit $a = b = 2$ und $I = \{p_2, p_1, s_1, s_2\}$. Daraus ergibt sich $S = \{p_2, s_1\}$; p_2 und s_1 sind die zwei nächsten Nachbarn von q. Wäre ein anderes Datum näher an q befände es sich auch innerhalb des BoxRadius(q, p_3, s_3), damit auch innerhalb des gleichen Teilwürfels wie q (grau schraffiertes Rechteck) und läge dann auf der Hilbert-Kurve zwischen p_3 und s_3.

Für jedes Datum müssen einige zusätzliche Informationen gespeichert werden. Die Daten werden daher praktischerweise als Objekte mit den in Tabelle 6.2 beschriebenen Eigenschaften gespeichert.

Das Hauptprogramm initialisiert zunächst die benötigten Variablen. In den Zeilen 9–18 werden die oberen und unteren Schranken berechnet und dann die Entscheidung getroffen, ob ein Datum ein Ausreißer, ein Ausreißerkandidat oder ein normales Datum ist.

In jedem Durchgang der Schleife verschiebt die Abbildung $V^{(j)}$ die Daten entlang der Hauptdiagonale. Dadurch landen die Daten in anderen Teilwürfeln, wir erhalten eine neue Ordnung $\leq_h$ und können dadurch die Abschätzungen verbessern. Die Schleife wird solange ausgeführt, bis entweder die gewünschte Anzahl an Ausreißern gefunden wurde oder die Daten insgesamt $(n + 1)$-mal verschoben wurde. Liegt jetzt noch immer keine Lösung vor, wird in Zeile 21 die exakte Lösung berechnet.

Die Funktion Scan im Algorithmus 8 berechnet für jedes Datum die oberen und unteren Schranken. Die Abfrage in Zeile 3 entscheidet, ob das Datum überhaupt noch als Ausreißer in Frage kommen kann, ob nämlich die obere Schranke des i-ten Datums x_i

Variable	Wert
OUT	Speichert die $\mathfrak{n}$ Daten mit maximalem Wert für $upperBound$.
WLB	Speichert die $\mathfrak{n}$ Daten mit maximalem Wert für $lowerBound$.
TOP	Vereinigung von OUT und WLB, beinhaltet maximal $2\mathfrak{n}$ Daten.
$globalLowerBound$	Untere Schranke für das Gewicht des $W_{\mathfrak{n}}^k$-Ausreißer.
$numberCandidates$	Anzahl an Daten mit $upperBound \geq globalLowerBound$, also von Daten, die noch als Ausreißer in Frage kommen.
$numberOutlier$	Anzahl an Daten, die bereits als tatsächliche Ausreißer erkannt wurden.

Tabelle 6.1: verwendete Variablen der Algorithmen 7, 8 und 9

Variable	Wert
$hilbert$	Der Hilbert-Wert des Datums
$level$	Ordnung des kleinsten Teilwürfels, der sowohl das Datum, als auch den Nachfolger des Datums beinhaltet
$distance$	Abstand d_p zwischen dem Datum und einem Abfragedatum
$lowerBound$	Unter Schranke für das Gewicht des Datums
$upperBound$	Obere Schranke für das Gewicht des Datums

Tabelle 6.2: Eigenschaften der Klasse für die Daten

größer oder gleich der unteren Schranke über alle Daten ist. Falls die untere Schranke gleich der oberen Schranke ist (Zeile 4), wurde bereits der exakte Wert für das Gewicht $w^k(x_i)$ gefunden. Die Funktion muss keine Schranken mehr berechnen und kann das Datum überspringen.

Als nächstes wird in Zeile 5 und 6 der Teilwürfel, in dem die nächsten k Nachbarn des Datums x_i liegen, betrachtet. „Nächste“ bezieht sich dabei nicht auf die Minkowski-Metrik, sondern auf die Lage der Daten entlang der Hilbert-Kurve. Der Umkreisradius dieses Teilwürfels multiplizert mit k liefert eine erste obere Schranke für das Gewicht. Ist diese obere Schranke kleiner als die kleinste untere Schranke über alle Daten, kann das Datum x_i kein Ausreißer sein.

Ansonsten berechnet die Funktion InnerScan jetzt genauere obere und untere Schranken für das Gewicht unter Zuhilfenahme von Lemma 6.11. InnerScan bezieht dabei $maxcount$ Nachbarn des aktuell betrachteten Datums ein. Die Variable $maxcount$ hat zunächst den Wert $2k$ (Algorithmus 7, Zeile 14). Während der Algorithmus läuft, verringert sich die Größe $numberCandidates$, der Wert von $maxcount$ wächst dementsprechend. Im Anschluss aktualisiert die Funktion Scan die Listen OUT und WLB.

Die Funktion InnerScan fügt in einer Schleife (Zeile 6–29) den jeweils nächsten Nachbar des Datums x_i einer Liste NN hinzu. Der nächste Nachbar wird mithilfe zweier Zeiger a und b innerhalb eines kleinstmöglichen Teilwürfels gesucht. Auch hier betrachtet die Funktion die nächsten Nachbarn entlang der Hilbert-Kurve und nicht bezüglich der Minkowski-Metrik.

Es gibt zwei Möglichkeiten, um die Schleife zu verlassen:

- Falls der Abstand laut Minkowski-Metrik zu den k nächsten Nachbarn entlang der

Hilbert-Kurve kleiner ist als die globale untere Schranke *globalLowerBound* (Zeile 19). In diesem Fall ist das Datum kein Ausreißer.

- Falls der Abstand zum k-nächsten Nachbarn kleiner gleich dem Abstand zum Rand des Teilwürfels mit Ordnung $level + 1$ ist (Zeile 24). Der Algorithmus sucht den nächsten Nachbarn immer im Teilwürfel mit Ordnung *level*. Daher wurden bereits alle Daten im Teilwürfel mit Ordnung $level + 1$ betrachtet. Folglich sind die k nächsten Nachbarn bereits berücksichtigt, alle noch nicht betrachteten Daten haben einen größeren Abstand.

Die neue obere Schranke ist einfach die Summe der Abstände zu allen Daten in der Liste NN. Für die untere Schranke werden alle Daten berücksichtigt, deren Abstand zum betrachteten Datum x_i kleiner r ist. Die Summe der Abstände zu diesen Daten ist die neue untere Schranke gemäß Lemma 6.11.

Sollte der Zähler j gleich der Anzahl aller Daten minus 1 sein, wurden alle anderen Daten betrachtet. Die Liste NN enthält dann die exakten nächsten Nachbarn. Der Wert der beiden Schranken ist gleich dem exakten Wert des Gewichts w^k.

6.4 Qualität der Näherungslösung

Um mithilfe von Lemma 6.11 zu besseren Abschätzungen zu kommen, verschiebt man die Daten mit der Abbildung $V^{(j)}$ bis zu n-mal um $(1/(n+1), \ldots, 1/(n+1))$ entlang der Hauptdiagonale des Einheiteswürfels. Falls der Algorithmus 7 nach n Verschiebungen noch nicht die exakte Lösung gefunden hat, so kann man mithilfe von Satz 6.18 zumindest eine Aussage über die Abweichung der bis dahin erhaltenen Lösung machen. Satz 6.18 stammt von [Ang05] und verwendet Definition 6.15 und Lemma 6.16 von [Cha98].

Definition 6.14. Sei $L^* = \{x_1, \ldots, x_{\mathfrak{n}}\} \in D$ eine Teilmenge der Daten. Weiters sei $L = \{q_1, \ldots, q_{\mathfrak{n}}\} \subseteq D$ die Menge der $W_{\mathfrak{n}}^k$-Ausreißer und $\varepsilon \in \mathbb{R}, \varepsilon > 0$. Es gelte für $i = 1, \ldots, \mathfrak{n} - 1$:

$$w^k(x_i) \geq w^k(x_{i+1}),$$
$$w^k(q_i) \geq w^k(q_{i+1}).$$

Wir bezeichnen L^* als *ε-Näherung von L* wenn gilt:

$$\varepsilon w^k(x_i) \geq w^k(q_i), \quad i = 1, \ldots, \mathfrak{n}.$$

Im Folgenden kann die Kantenlänge r eines Würfels auch größer 1 sein.

Definition 6.15. Ein Datum $x \in D \subseteq \mathbb{R}^n$ liegt *c-zentral*, $c \in \mathbb{R}$, $0 \leq c < 0.5$, in einem Teilwürfel mit Kantenlänge r, wenn für jede Koordinate x_i gilt (Abbildung 6.8):

$$cr \leq x_i \mod r < (1 - c)r, \quad \forall i = 1, \ldots, n.$$

Algorithmus 7 : Hilbert Outlier Detection

Input : Daten D, Anzahl von Lösungen $\mathfrak{n}$, Anzahl der betrachteten Nachbarn k, Ordnung der Hilbert-Kurve h

Output : Lösungsmenge $L = \{q_1, \ldots, q_{\mathfrak{n}}\} \subseteq D$

1 `// Initialisierung der Variablen:`
2 $TOP \leftarrow \{\}$
3 $OUT \leftarrow \{\}$
4 $WLB \leftarrow \{\}$
5 $numberCandidates \leftarrow |D|$
6 $numberOutlier \leftarrow 0$
7 $globalLowerBound \leftarrow 0$
8 $j \leftarrow 0$
9 **while** $j \leq \dim(D) \wedge numberOutlier < \mathfrak{n}$ **do**
10 $\quad D_{trans} \leftarrow V^{(j)}(D)$ `// verschiebe alle Daten`
11 $\quad$ Berechne Hilbert-Werte aller Daten in D_{trans}
12 $\quad$ Sortiere Daten nach ihren Hilbert-Werten
13 $\quad numberCandidates \leftarrow |\{x \in D : x.upperBound > globalLowerBound\}|$
14 $\quad \text{Scan}\,(D_{trans}, \frac{k|D|}{numberCandidates})$
15 $\quad numberOutlier \leftarrow |\{x \in OUT \ : \ x.lowerBound = x.upperBound \wedge x.upperBound > globalLowerBound\}|$
16 $\quad TOP \leftarrow OUT \cup WLB$
17 $\quad j \leftarrow j + 1$
18 **end**
19 **if** $numberOutlier < \mathfrak{n}$ **then**
20 $\quad D_{trans} \leftarrow V^{(\dim(D))}(D)$ `// verschiebe alle Daten`
21 $\quad \text{Scan}\,(D_{trans}, |D|\,)$
22 **end**
23 $L \leftarrow OUT$

Algorithmus 8 : Scan

Input : Daten D, Suchbereich s
Output : D, OUT, WLB, $globalLowerBound$

```
1  for i ← 1 to |D| do
2      // x_i ∈ D
3      if x_i.upperBound ≥ globalLowerBound then
4          if x_i.lowerBound < x_i.upperBound then
5              Suche kleinsten Teilwürfel C, in dem die nächsten k Nachbarn (bezogen
                auf die Hilbert-Kurve) von x_i liegen.
6              upperBoundEstimator ← k · Umkreisradius(C)
7              if upperBoundEstimator < globalLowerBound then
8                  x_i.upperBound ← upperBoundEstimator
9              else
10                 maxcount ← min(2s, |D|)
11                 if x_i ∈ TOP then
12                     maxcount ← |D|
13                 end
14                 InnerScan(D, i, maxcount)
                                  // berechnet newLowerBound und newUpperBound
15                 if newLowerBound > x_i.lowerBound then
16                     x_i.lowerBound ← newLowerBound
17                 end
18                 if newUpperBound < x_i.upperBound then
19                     x_i.upperBound ← newUpperBound
20                 end
21             end
22         end
23     end
24     if |OUT| < n ∨ min_{y∈OUT}(y.upperBound) < x_i.upperBound then
25         Entferne gegebenenfalls y mit minimaler upperBound aus WLB
26         Füge x_i zu OUT hinzu
27     end
28     if |WLB| < n ∨ min_{y∈WLB}(y.lowerBound) < x_i.upperBound then
29         Entferne gegebenenfalls y mit minimaler upperBound aus WLB
30         Füge x_i zu WLB hinzu
31     end
32     globalLowerBound ← max(globalLowerBound, min(WLB))
33 end
```

Algorithmus 9 : InnerScan

Input : Daten D, Index des Abfragedatums i, $maxcount$

Output : Schranken $newLowerBound$ und $newUpperBound$

// Initialisierung der Variablen:
$NN \leftarrow \{\}$ // Next Neighbors
$a \leftarrow i$; $b \leftarrow i$
$levela \leftarrow h$; $levelb \leftarrow h$; $level \leftarrow h$
$j \leftarrow 0$
while $j < maxcount$ **do**
  $j \leftarrow j + 1$
  if $x_{a-1}.level > x_b.level$ **then**
    $a \leftarrow a - 1$; $levela \leftarrow \min(levela, x_a.level)$; $c \leftarrow a$
  else
    $levelb \leftarrow \min(levelb, x_b.level)$; $b \leftarrow b + 1$; $c \leftarrow b$
  end
  $x_c.distance \leftarrow d_p(p, x_c)$
  if $|NN| < k \vee \exists y \in NN \ : \ x.distance < y.distance$ **then**
    Entferne gegebenenfalls y mit maximaler $y.distance$ aus NN
    Füge x_c zu NN hinzu
  end
  if $|NN| = k$ **then**
    if $\sum_{y \in NN} y.distance < globalLowerBound$ **then**
      Exit while
    else if $\max(levela, levelb) < level$ **then**
      $level \leftarrow \max(levela, levelb)$
      $\delta \leftarrow \text{MinDist}\left(p, 2^{-(level+1)}\right)$
      if $\delta \geq \max_{y \in NN}(y.distance)$ **then**
        Exit while
      end
    end
  end
end
if $j = |D| - 1$ **then**
  $newLowerBound \leftarrow \sum_{y \in NN} y.distance$
  $newUpperBound \leftarrow newLowerBound$
else
  $r \leftarrow \text{BoxRadius}\left(V^{(j)}(x_i), V^{(j)}(x_{a-1}), V^{(j)}(x_{b+1})\right)$
  $newLowerBound \leftarrow \sum_{y \in NN, y.distance \leq r} y.distance$ // lt. Lemma 6.11
  $newUpperBound \leftarrow \sum_{y \in NN} y.distance$
end

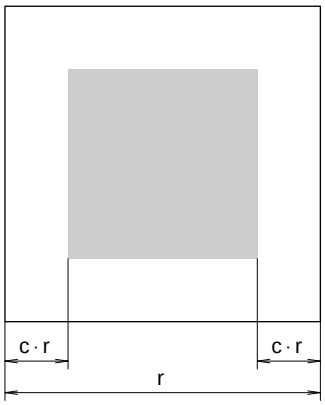

Abbildung 6.8: Daten innerhalb des grauen Gebiets liegen c-zentral.

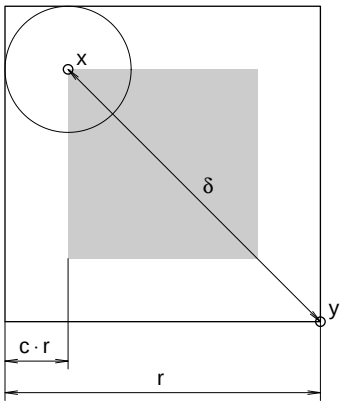

Abbildung 6.9: Abschätzung des Abstands zu einem c-zentralem Datum x.

Da $cr < r$ gilt, ist die Ungleichung in dieser Definition äquivalent zu

$$2cr \leq (x_i + cr) \mod r.$$

Lemma 6.16. *Sei die Dimension n des Datenraums $D \subseteq \mathbb{R}^n$ eine gerade Zahl. Dann existiert für jedes Datum $x \in D$ und $r = 2^{-l}$ eine Verschiebung $V^{(j)}$, $j = 0, \ldots, n$, sodass $V^{(j)}(x) \in \mathcal{C}$ $\left(\frac{1}{2n+2}\right)$-zentral im Teilwürfel $\mathcal{C}$ mit Kantenlänge r liegt.*

Sollte die Dimension unseres Datenraums ungerade sein, können wir einfach eine weitere Koordinate mit $x_i = 0$ hinzufügen.

Beweis. Wir nehmen an, dass $V^{(j)}(x)$ für kein j $\left(\frac{1}{2n+2}\right)$-zentral ist. D. h. für jedes j existiert eine Koordinate $i(j)$ mit

$$\left(x_{i(j)} + \frac{j}{n+1} + \frac{r}{2n+2}\right) \mod r < \frac{r}{n+1}.$$

Wir multiplizieren mit $(n+1)2^l$:

$$\left((n+1)2^l x_{i(j)} + 2^l j + \frac{1}{2}\right) \mod (n+1) < 1.$$

Wir haben n Koordinaten und $n+1$ Verschiebungen, daher müssen zwei Verschiebungen j und j' existieren mit $j \neq j'$ und $i(j) = i(j')$. Sei $z := (n+1)2^l x_{i(j)} + \frac{1}{2}$, so gilt

$$\begin{aligned}
&\left(z + 2^l j\right) \mod (n+1) < 1, \\
&\left(z + 2^l j'\right) \mod (n+1) < 1 \\
&\Rightarrow 2^l j \equiv 2^l j' \mod (n+1).
\end{aligned}$$

$n+1$ ist ungerade, daher sind 2^l und $n+1$ teilerfremd, woraus $j = j'$ folgt, im Widerspruch zur Annahme. □

Lemma 6.17. *Sei x ein Datum, wobei nach der n-ten Verschiebung um $V^{(1)}$ (in Algorithmus 7, Zeile 19) $x.upperBound > globalLowerBound$ gilt. Wir definieren:*

$$\varepsilon_n := 2\sqrt[p]{\sum_{i=1}^{n}(2n+1)^p}.$$

Dann folgt:

$$x.upperBound \leq k\varepsilon_n w^k(x)$$

nach der n-ten Verschiebung.

Beweis. Wie in Definition 2.8 bezeichnen wir mit $d^k(x)$ den Abstand von x zu dessen k-nächstem Nachbarn. Wir betrachten Würfel, für deren Seitenlänge $r = 2^{-l}$ gilt:

$$\frac{r}{4n+4} \leq d^k(x) < \frac{r}{2n+2} \tag{6.1}$$

Diese Ungleichungen definieren r eindeutig. Die Seitenlänge r kann in diesem Fall größer 1 sein.

Aufgrund von Lemma 6.16 existiert eine Verschiebung, sodass $V^{(j)}(x)$ $\frac{1}{2n+2}$-zentral in einem Würfel $\mathcal{C}$ ist. Wir wollen zeigen, dass für diese Verschiebung die Behauptung gilt.

Einige Überlegungen: Für den Abstand δ zwischen x und einem beliebigen anderem Datum in $\mathcal{C}$ gilt (Abbildung 6.9):

$$\begin{aligned}\delta &\leq \sqrt[p]{\sum_{i=1}^{n}\left(r - \frac{r}{2n+2}\right)^p} \\ &= \sqrt[p]{\sum_{i=1}^{n}\left(\frac{2n+1}{2n+2}r\right)^p}.\end{aligned}$$

Da $x.upperBound$ während des Algorithmus monoton fällt, gilt $x.upperBound > globalLowerBound$ während des gesamten Algorithmus. Die Bedingung in Zeile 3 von Algorithmus 8 ist daher immer erfüllt.

Betrachten wir die Zeile 4 von Algorithmus 8. Falls $x.lowerBound = x.upperBound$ ist, ist $x.upperBound = w^k(x) \leq k\varepsilon_n w^k(x)$, da sowohl k als auch ε_n größer oder gleich 1 sind. $x.upperBound$ wird in Folge nicht mehr geändert, daher gilt die Behauptung auch am Ende des Algorithmus 7.

Sei nun $x.lowerBound < x.upperBound$. Aufgrund von Ungleichung 6.1 sind die k-nächsten Nachbarn Elemente von $\mathcal{C}$. Der in Zeile 5 gefundene Würfel ist somit eine Teilmenge des Würfels $\mathcal{C}$. Wird $x.upperBound$ in Zeile 8 neu gesetzt, ist der neue Wert daher kleiner $k\delta$. Das gleiche Argument gilt auch, falls die Funktion InnerScan eine genauere Abschätzung berechnet.

Es gelten daher die Ungleichungen

$$\begin{aligned} x.upperBound &\leq k\delta \\ &\leq kr \sqrt[p]{\sum_{i=1}^{n} \left(\frac{2n+1}{2n+2}\right)^p} \\ &\leq k(4n+4)d^k(x) \sqrt[p]{\sum_{i=1}^{n} \left(\frac{2n+1}{2n+2}\right)^p} \\ &\leq k\varepsilon_n d^k(x) \\ &\leq k\varepsilon_n w^k(x). \end{aligned}$$

Wieder aufgrund der monotonen Schrumpfung von $x.upperBound$ gilt diese Ungleichung auch am Ende des Algorithmus.

□

Satz 6.18. *Bezeichne $OUT^* = \{x_1, \ldots, x_{\mathfrak{n}}\}$ die Menge OUT am Ende des ersten Teils von Algorithmus 7, Zeile 18. OUT^* ist eine $k\varepsilon_n$-Näherung der tatsächlichen Lösung $L = \{q_1, \ldots, q_{\mathfrak{n}}\}$.*

Beweis. O. B. d. A. sei $x_1.upperBound \geq \ldots \geq x_{\mathfrak{n}}.upperBound$. Dann gilt $x_i.upperBound \geq w^k(q_i)$ für $i = 1, \ldots, \mathfrak{n}$. Andernfalls wäre statt x_i das Datum q_i in die Menge OUT^* aufgenommen worden. Aus Lemma 6.17 folgt:

$$k\varepsilon_n w^k(x_i) \geq x_i.upperBound \geq w^k(q_i).$$

Wir betrachten jetzt jene Permutation π, die OUT^* nach Gewicht sortiert

$$w^k(x_{\pi(1)}) \geq \ldots \geq w^k(x_{\pi(\mathfrak{n})}),$$

und zeigen, dass $k\varepsilon_n w^k(x_{\pi(i)}) \geq w^k(q_i)$ gilt. Es folgt eine Fallunterscheidung:

- $\pi(i) < i$:
$$k\varepsilon_n w^k(x_{\pi(i)}) \geq k\varepsilon_n w^k(x_i) \geq w^k(q_i).$$
- $\pi(i) \geq i$: In diesem Falle existiert ein $x_j, j = \{1, \ldots, i\}$, mit $w^k(x_{\pi(i)}) \geq w^k(x_j)$ ($x_{\pi(i)}$ wanderte nach oben, daher muss zumindest ein Datum nach unten gewandert sein). Weiters:
$$k\varepsilon_n w^k(x_{\pi(i)}) \geq k\varepsilon_n w^k(x_j) \geq w^k(q_j) \geq w^k(q_i).$$

□

6.5 Laufzeitanalyse

Die Ordnung der Hilbert-Kurve κ sei für die Laufzeitanalyse konstant. Ausgangspunkt für die Laufzeitanalyse ist der Algorithmus 8, *Scan*. Die Laufzeit der Berechnung von *upperBoundEstimator* beträgt $\mathcal{O}(k)$ für Zeile 5 und $\mathcal{O}(n)$ für die Berechnung des Umkreisradius (Zeile 6), in Summe $\mathcal{O}(k+n)$.

Die Laufzeit eines Schleifendurchgangs in Algorithmus 9, *InnerScan*, Zeile 6, beträgt $\mathcal{O}(n)$ für Zeile 13 und $\mathcal{O}(\log k)$, um NN zu aktualisieren. Dabei setzen wir eine geeignete Datenstruktur für NN voraus. Die Schleife wird bei den maximal $2\mathfrak{n}$ Daten in TOP maximal N mal durchlaufen. Bei den restlichen $\mathcal{O}(N)$ Daten wird die Schleif maximal k mal durchlaufen. Zusammengefasst erhalten wir $\mathcal{O}\left(N\left(2\mathfrak{n}+k\right)\left(n+\log k\right)\right)$.

Die Berechnung von *BoxRadius*, *newLowerBound* und *newUpperBound* (Zeile 34–36) erfolgt in $\mathcal{O}(n+2k)$. Für die Aktualisierung von OUT und WLB können wir erneut $\mathcal{O}(2\log\mathfrak{n})$ annehmen.

Somit ergibt sich als Laufzeit für *Scan*:

$$
\begin{aligned}
&\mathcal{O}\left(N(k+n)+N(n+2k+2\log\mathfrak{n})+N(2\mathfrak{n}+k)\left(n+\log k\right)\right)\\
&=\mathcal{O}\left(N\left(2n+3k+2\log\mathfrak{n}+(2\mathfrak{n}+k)\left(n+\log k\right)\right)\right)\\
&\leq\mathcal{O}\left(N\left(n+k+\mathfrak{n}+(\mathfrak{n}+k)\left(n+\log k\right)\right)\right)\\
&\leq\mathcal{O}\left(N\left(n+(\mathfrak{n}+k)\left(n+\log k\right)\right)\right)\\
&\leq\mathcal{O}\left(N\left(n+\log k+(\mathfrak{n}+k)\left(n+\log k\right)\right)\right)\\
&=\mathcal{O}\left(N(\mathfrak{n}+k)\left(n+\log k\right)\right).
\end{aligned}
$$

Die Berechnung und Sortierung der Hilbert-Werte (Algorithmus 7, Zeilen 11 und 12) hat eine Laufzeit von $\mathcal{O}(nN\log N)$ (siehe [Ang05]). Die Autoren von [Ang05] treffen weiters die Annahmen $\mathcal{O}(\mathfrak{n})=\mathcal{O}(k)$ und $\mathcal{O}(n)\geq\mathcal{O}(\log k)$. Als Endresultat ergibt sich für die Laufzeit von *HilbertOutlierDetection* somit

$$
\begin{aligned}
&\mathcal{O}\left(nN\log N\right)+\mathcal{O}\left(N(\mathfrak{n}+k)\left(n+\log k\right)\right)\\
&\leq\mathcal{O}\left(N\left(n\log N+4\mathfrak{n}n\right)\right)\\
&\leq\mathcal{O}\left(nN\left(\log N+\mathfrak{n}\right)\right).
\end{aligned}
$$

7 Angle-Based Outlier Detection

[Kri08] stellten 2008 die Idee vor, Ausreißer nicht mit einer Metrik, sondern mit Winkeln zu finden (*Angle-Based Outlier Detection*, *ABOD*). Wir betrachten ein Abfragedatum q und berechnen von q aus die Winkel zwischen allen Paaren von anderen Daten. Wenn q in der Nähe von anderen Daten liegt (q entspricht dann einem normalen Datum) nehmen die Winkel viele verschiedene Werte zwischen 0 und 2π an. Wenn q weit entfernt von den anderen Daten ist (q entspricht dann einem Ausreißer) nehmen alle Winkel ungefähr denselben Wert an (siehe Abbildungen 7.1 und 7.2).

Entscheidend sind nicht die Werte der Winkel sondern die Streuung dieser Werte: Bei Ausreißern ist die Streuung gering, bei normalen Daten ist sie hoch. Bei Daten am Rand der normalen Daten liegt die Streuung dazwischen.

Die Dimension des Datenraums beeinflusst die Winkel weniger als die Abstände. Außerdem verwendet dieses Verfahren keine Parameter, die ein Anwender im Vorhinein setzen muss.

Definition 7.1. Im folgenden sei $\langle\ ,\ \rangle$ das kanonische Skalarprodukt

$$\langle\ ,\ \rangle :\ \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}, \quad \langle x, y\rangle \mapsto \sum_{i=1}^{n} x_i y_i$$

und $x \otimes y$ das dyadische Produkt:

$$\otimes :\ \mathbb{R}^m \times \mathbb{R}^n \to \mathbb{R}^{m\times n}, \quad x \otimes y \mapsto A \quad \text{mit} \quad A_{ij} = x_i y_j.$$

Mit $\|\ \|_F$ bezeichnen wir die Frobeniusnorm einer Matrix

$$\|\ \|_F :\ \mathbb{R}^{m\times n} \to \mathbb{R}^0_+, \quad A \mapsto \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} \left|a_{ij}^2\right|}.$$

7.1 Der Angle-Based Outlier Factor

Definition 7.2 (ABOF)**.** Sei $D \subseteq \mathbb{R}^n$ ein Datenraum und $q \in D$ ein Abfragedatum. Der *Angle-Based Outlier Factor (ABOF)* ist definiert als die Stichprobenvarianz der Kosinusse der Winkel zwischen allen Paaren von Vektoren $\{x-q, y-q\},\ x \in D\backslash\{q\},\ y \in D\backslash\{q, x\}$. Die Kosinusse der Winkel werden dabei mit dem Produkt der Abstände $\mathrm{d}\,(q, x)$

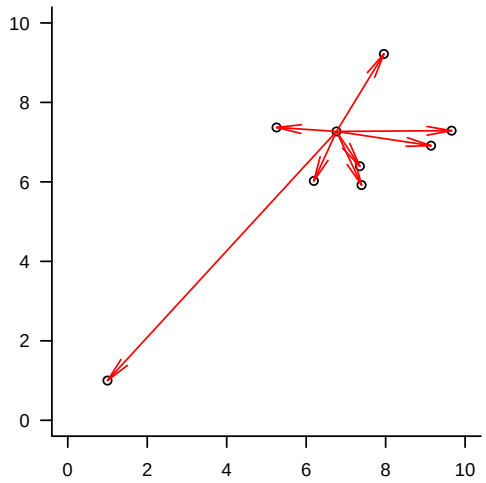

Abbildung 7.1: Winkel zu anderen Daten bei einem normalen Datum

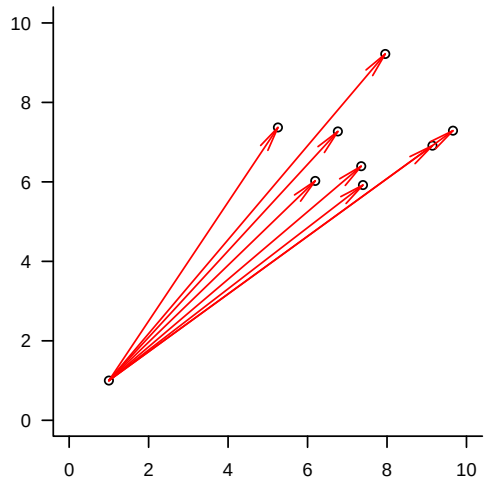

Abbildung 7.2: Winkel zu anderen Daten bei einem Ausreißer

und $\mathrm{d}\,(q,y)$ gewichtet:

$$ABOF(q) := \mathbb{V}_{\{x,y\}} \left[\frac{\langle x-q, y-q \rangle}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)} \right]$$
$$= \frac{\sum\limits_{\{x,y\}} \frac{1}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)} \left(\frac{\langle x-q,\ y-q \rangle}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)} \right)^2}{\sum\limits_{\{x,y\}} \frac{1}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)}} - \left(\frac{\sum\limits_{\{x,y\}} \frac{1}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)} \frac{\langle x-q,\ y-q \rangle}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)}}{\sum\limits_{\{x,y\}} \frac{1}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)}} \right)^2$$

mit

$$x \in D \setminus \{q\},\ y \in D \setminus \{q, x\}.$$

Je kleiner der ABOF eines Datums ist, desto eher ist dieses Datum ein Ausreißer. Durch die Gewichtung mit den Abständen fließen Winkel zwischen nahen Daten stärker in das Ergebnis ein, als Winkel zwischen fernen Daten. Die dafür verwendete Metrik hat nur einen geringen Einfluss auf den ABOF. Daher leidet das Verfahren kaum unter dem Fluch der Dimensionalität, obwohl es eine Metrik verwendet.

Beispiel 7.3. Betrachte Abbildung 7.3: Bei Daten am Rand des Feldes ist der ABOF kleiner, bei Daten, die in der Mitte liegen, ist der ABOF größer. Es wurde die euklidische Metrik verwendet.

7.2 FastABOD

Möchte man den ABOF für alle Daten eines Datenraums D berechnen, muss man alle Tripel der Daten betrachten. Man erhält eine Laufzeit von $\mathcal{O}(N^3)$, daher ist das Verfah-

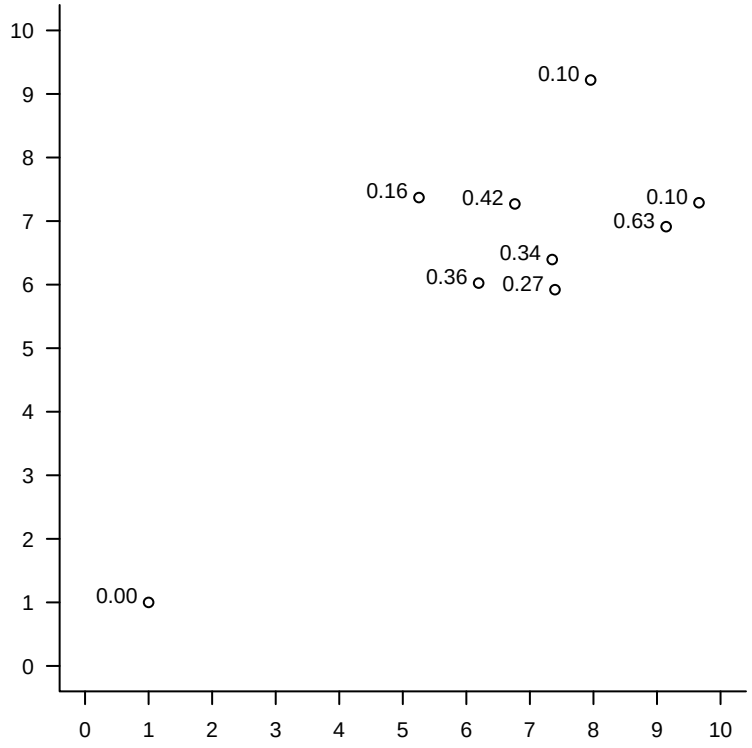

Abbildung 7.3: ABOF von Beispieldaten unter Verwendung der euklidischen Metrik.

ren für praktische Anwendungen nicht geeignet. Wir müssen nach Möglichkeiten suchen, die die Laufzeit verringern.

Als erste Verbesserung (*FastABOD*) betrachten [Kri08] nicht alle Paare von Daten sondern nur Paare von Daten, die in der Nähe des Abfragedatums liegen. Im ersten Schritt sucht man für jedes Datum die k nächsten Nachbarn. Im zweiten Schritt berechnet man den *approximate Angle-Based Outlier Factor (approxABOF)*:

Definition 7.4. Sei $D \subseteq \mathbb{R}^n$ ein Datenraum, $q \in D$ ein Abfragedatum und $N_k(q) \subseteq D \setminus \{q\}$ die Menge der k nächsten Nachbarn von q. Der *approximate Angle-Based Outlier Factor (approxABOF)* ist definiert als:

$$\text{approxABOF}(q) := \mathbb{V}_{\{x,y\}} \left[\frac{\langle y - q, z - q \rangle}{\text{d}\,(q,x)\ \ \text{d}\,(q,y)} \right]$$

mit

$$x \in N_k(q),\ y \in N_k(q) \setminus \{x\}.$$

Man betrachtet beim approxABOF also nur die Winkel zwischen Paaren in $N_k(q)$. Die Suche nach den nächsten Nachbarn für alle Daten hat eine Laufzeit $\mathcal{O}(N^2)$, die Berechnung des approxABOF für alle Daten hat somit eine Laufzeit von $\mathcal{O}(N^2 + N \cdot k^2)$.

[Kri08] stellen allerdings fest, dass der approxABOF für den Fluch der Dimensionalität anfällig ist (bei der Berechnung von $N_k(q)$). Die Qualität der Ergebnisse nimmt mit steigender Dimensionalität ab. Der approxABOF von Ausreißern ist zwar üblicherweise klein. Es gibt aber manchmal normale Daten, die ebenfalls kleine Werte haben.

7.3 LB-ABOD

Bei FastABOD betrachtet man nur Daten, die zur Menge der nächsten Nachbarn gehören, alle anderen Daten ignoriert man. Um die Qualität der Ergebnisse zu verbessern, nimmt man bei *lower bound ABOD (LB-ABOD)* auch die weiter entfernten Daten mit einer Abschätzung in die Berechnung auf. Man erhält den *lower bound ABOF (LB-ABOF)* als Abschätzung für den ABOF.

Wir stellen folgende Anforderungen an den LB-ABOF:

- Die Abschätzung soll möglichst konservativ sein.
- Es soll gelten: $\text{ABOF}(q) - \text{LB-ABOF}(q) > 0 \quad \forall q \in D$.
- Die Berechnung soll effizient sein.

$$ABOF(q) = \frac{\sum\limits_{\{x,y\}\in M_A} \frac{1}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)}\ \left(\frac{\langle x-q,\ y-q\rangle}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)}\right)^2}{\sum\limits_{\{x,y\}\in M_A} \frac{1}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)}} - \left(\frac{\sum\limits_{\{x,y\}\in M_A} \frac{1}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)}\ \frac{\langle x-q,\ y-q\rangle}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)}}{\sum\limits_{\{x,y\}\in M_A} \frac{1}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)}}\right)^2$$

$$= \frac{\sum\limits_{\{x,y\}\in M_B} \frac{1}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)}\ \left(\frac{\langle x-q,\ y-q\rangle}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)}\right)^2 + R_1}{\sum\limits_{\{x,y\}\in M_A} \frac{1}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)}}$$

$$- \left(\frac{\sum\limits_{\{x,y\}\in M_B} \frac{1}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)}\ \frac{\langle x-q,\ y-q\rangle}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)} + R_2}{\sum\limits_{\{x,y\}\in M_A} \frac{1}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)}}\right)^2$$

mit

$$R_1 = \sum_{\{x,y\}\in M_C} \frac{1}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)}\ \left(\frac{\langle x-q,\ y-q\rangle}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)}\right)^2,$$

$$R_2 = \sum_{\{x,y\}\in M_C} \frac{1}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)}\ \frac{\langle x-q,\ y-q\rangle}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)},$$

und

$$M_A := \{\{x,y\} \mid x \in D \setminus \{q\} \wedge y \in D \setminus \{q,x\}\},$$
$$M_B := \{\{x,y\} \mid x \in N_k(q) \wedge y \in N_k(q) \setminus \{q\}\},$$
$$M_C := \{\{x,y\} \mid \{x,y\} \in M_A \wedge \{x,y\} \notin M_B\}$$
$$(\Rightarrow\ M_C = M_A \setminus M_B).$$

Wir versuchen R_1 und R_2 durch Abschätzungen zu ersetzen, die Abschätzungen sollen konservativ sein. Das erreichen wir, indem wir für R_1 eine möglichst kleine Abschätzung finden und für R_2 eine möglichst große.

R_1 ist minimal wenn alle Winkel $\pi/2$ sind, dann sind alle Skalarprodukte 0 und es gilt $R_1 = 0$. R_2 ist maximal, wenn alle Skalarprodukte 1 sind (alle Winkel 0). Wir müssen zusätzlich noch die Gewichte berücksichtigen:

$$R_2 \leq \sum_{\{x,y\}\in M_C} \frac{1}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)}\ \frac{1}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)}$$
$$= \sum_{\{x,y\}\in M_C} \left(\frac{1}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)}\right)^2$$
$$= \sum_{\{x,y\}\in M_A} \left(\frac{1}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)}\right)^2 - \sum_{\{x,y\}\in M_B} \left(\frac{1}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)}\right)^2.$$

Wir formen die Summe um:

$$\begin{aligned}
\sum_{\{x,y\}\in M_A} \frac{1}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)} &= \frac{1}{2} \sum_{x\in M_A} \sum_{y\in M_A, y\neq x} \frac{1}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)} \\
&= \frac{1}{2} \sum_{x\in M_A} \frac{1}{\mathrm{d}\,(q,x)} \sum_{y\in M_A, y\neq x} \frac{1}{\mathrm{d}\,(q,y)} \\
&= \frac{1}{2} \sum_{x\in M_A} \frac{1}{\mathrm{d}\,(q,x)} \left(\sum_{y\in M_A} \frac{1}{\mathrm{d}\,(q,y)} - \frac{1}{\mathrm{d}\,(q,x)} \right) \\
&= \frac{1}{2} \left[\sum_{x\in M_A} \frac{1}{\mathrm{d}\,(q,x)} \sum_{y\in M_A} \frac{1}{\mathrm{d}\,(q,y)} - \sum_{x\in M_A} \left(\frac{1}{\mathrm{d}\,(q,x)} \right)^2 \right] \\
&= \frac{1}{2} \left[\left(\sum_{x\in M_A} \frac{1}{\mathrm{d}\,(q,x)} \right)^2 - \sum_{x\in M_A} \left(\frac{1}{\mathrm{d}\,(q,x)} \right)^2 . \right]
\end{aligned}$$

Analog gilt

$$\sum_{\{x,y\}\in M_A} \left(\frac{1}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)} \right)^2 = \frac{1}{2} \left[\left(\sum_{x\in M_A} \left(\frac{1}{\mathrm{d}\,(q,x)} \right)^2 \right)^2 - \sum_{x\in M_A} \left(\frac{1}{\mathrm{d}\,(q,x)} \right)^4 \right].$$

Die Berechnungen der rechten Seite hat eine Laufzeit von $\mathcal{O}(N)$.

Wir definieren daher

$$\begin{aligned}
R_{2,app} := \frac{1}{2} &\left[\left(\sum_{x\in M_A} \left(\frac{1}{\mathrm{d}\,(q,x)} \right)^2 \right)^2 - \sum_{x\in M_A} \left(\frac{1}{\mathrm{d}\,(q,x)} \right)^4 \right] \\
&- \frac{1}{2} \left[\left(\sum_{x\in M_B} \left(\frac{1}{\mathrm{d}\,(q,x)} \right)^2 \right)^2 - \sum_{x\in M_B} \left(\frac{1}{\mathrm{d}\,(q,x)} \right)^4 \right]
\end{aligned}$$

und den LB-ABOF als:

Definition 7.5.

$$\begin{aligned}
\text{LB-ABOF}(q) := &\frac{\sum_{\{x,y\}\in M_B} \frac{1}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)} \left(\frac{\langle x-q,\ y-q\rangle}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)} \right)^2}{\sum_{\{x,y\}\in M_A} \frac{1}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)}} \\
&- \left(\frac{\sum_{\{x,y\}\in M_B} \frac{1}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)} \frac{\langle x-q,\ y-q\rangle}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)} + R_{2,app}}{\sum_{\{x,y\}\in M_A} \frac{1}{\mathrm{d}\,(q,x)\ \mathrm{d}\,(q,y)}} \right)^2 .
\end{aligned}$$

Die Summen über M_A in den Nennern kann man mit der Umformung oben ebenfalls in linearer Laufzeit berechnen.

Der LB-ABOD-Algorithmus berechnet zunächst den LB-ABOF von allen Daten und sortiert sie nach ihrem LB-ABOF. Von den l Daten mit dem niedrigsten LB-ABOF wird der ABOF berechnet, diese Daten werden in einer Ergebnisliste gespeichert. Der Algorithmus läuft jetzt der Reihe nach durch die übrigen $N - l$ Daten. Sollte der ABOF einer dieser übrigen Daten größer sein als der kleinste ABOF in der Ergebnisliste, wird dieses eine Datum der Ergebnisliste hinzugfügt, das andere aus der Ergebnisliste gelöscht. Algorithmus 10 stellt LB-ABOD in Pseudocode dar.

Algorithmus 10 : Lower-Bound-Angle-Based-Outlier-Detection

Input : Daten D, Anzahl der Rückgabewerte l
Output : *ResultList*, ABOF(q) $\forall q \in ResultList$

```
1  foreach q ∈ D do
2  |  Berechne LB-ABOF(q)
3  end
4  L ← sort(D)                                   // aufsteigend nach LB-ABOF
5  ResultList ← L[1, l]
6  L ← L[l + 1, length(L)]
7  Berechne ABOF(q)  ∀q ∈ ResultList
8  Berechne ABOF(L[1])
9  while ABOF(L[1]) > ABOF(ResultList[l]) do
10 |  Entferne ResultList[l] aus ResultList
11 |  Füge L[1] in ResultList[l] ein
12 |  Entferne L[1] aus L
13 |  Berechne ABOF(L[1])
14 end
```

Betrachten wir die Laufzeit von Algorithmus 10: Die Laufzeit der Schleife in den Zeilen 1–3 beträgt $\mathcal{O}(N^2 + N \cdot k^2)$ (N^2 für das Finden der k nächsten Nachbarn, $N \cdot k^2$ für das Berechnen der LB-ABOF). Die Laufzeit von Zeile 4 beträgt $\mathcal{O}(N \log N)$ und von Zeile 9–14 $\mathcal{O}(i \cdot N^2)$. Mit i bezeichnen wir die Anzahl der Durchläufe der Schleife in Zeile 9–14. Unter der Annahme, dass $i \ll N$ gilt, beträgt die Laufzeit des gesamten Algorithmus somit $\mathcal{O}(N^2)$.

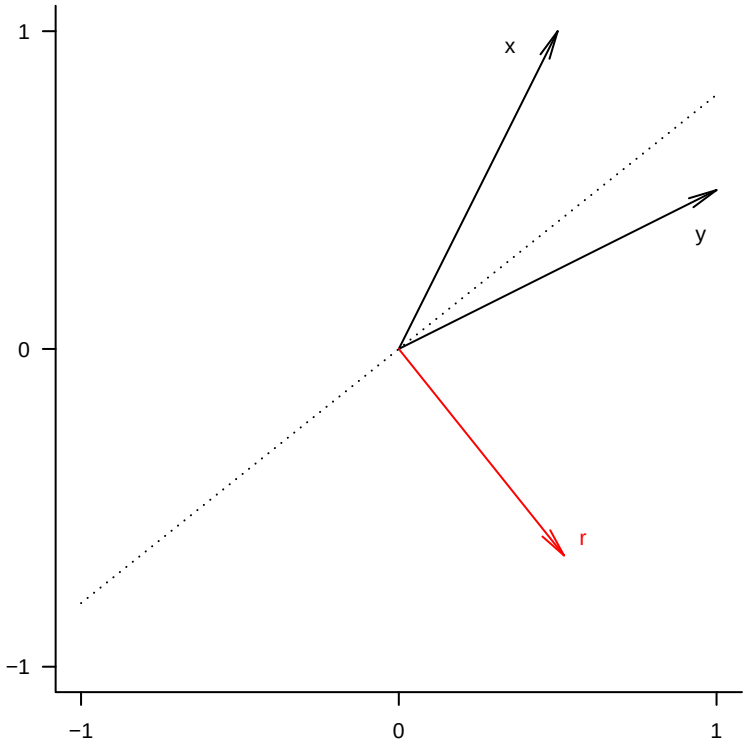

Abbildung 7.4: Winkel und Zufallsvektor r

7.4 Monte-Carlo-Algorithmus mit Laufzeit $N \log^2(N)$

Die Laufzeit von LB-ABOD wächst quadratisch. Das ist um eine Größenordnung schneller als der grundlegende ABOD-Algorithmus, kann für große Datenmengen allerdings noch immer zu langsam sein. [Pha12] stellten 2012 einen Monte-Carlo-Algorithmus mit Laufzeit $N \log^2(N)$ vor.

Der Algorithmus benutzt folgende Eigenheit (siehe Abbildung 7.4): Gegeben seien zwei Vektoren x und y, der Winkel zwischen den Vektoren beträgt φ. Wir erzeugen einen Vektor r, der in eine zufällig gewählte Richtung zeigt, und bilden die Hyperebene, zu der r orthogonal ist. Die Hyperebene teilt den Raum in zwei Halbräume. Die Wahrscheinlichkeit, dass x im linken Halbraum und y im rechten Halbraum liegt, beträgt $\varphi/(2\pi)$. Dieser anschaulich offensichtliche Sachverhalt (Satz 7.7) wird in [Goe95] formal hergeleitet.

Definition 7.6. Sei $D \subseteq \mathbb{R}^n$ die Menge aller Daten und $q \in D$ ein Abfragedatum, $x \in D \backslash \{q\}$ und $y \in D \backslash \{q, x\}$ zwei andere Daten. Weiters seien $r_1, r_2, \ldots, r_t \in \mathbb{R}^n$ zufällig gewählte Vektoren, deren Koordinaten unabhängig voneinander standardnormalverteilt

sind. Wir definieren für $i = 1, \ldots, t$ die Zufallsvariablen

$$X_{xqy}^{(i)} := \begin{cases} 1 & \text{für } x \cdot r_i < q \cdot r_i < y \cdot r_i, \\ 0 & \text{sonst.} \end{cases} \tag{7.1}$$

Die $X_{xqy}^{(i)}$ sind unabhängig. $X_{xqy}^{(i)}$ ist genau dann eins, wenn $(x - q) \cdot r_i < 0$ und $0 < (y - q) \cdot r_i$ gilt.

Satz 7.7. *Sei* $\varphi_{xqy} = \arccos\left(\frac{\langle x-q, y-q\rangle}{\mathrm{d}(q,x)\ \mathrm{d}(q,y)}\right)$. *Es gilt:*

$$\mathbb{P}[X_{xqy}^{(i)} = 1] = \frac{\varphi_{xqy}}{2\pi}. \tag{7.2}$$

Beweis. Siehe [Goe95, S. 1121]. □

Mit den $X_{xqy}^{(i)}$ können wir daher den Winkel φ_{xqy} schätzen:

Definition 7.8.

$$\varphi_{xqy} = 2\pi \mathbb{E} X_{xqy},$$
$$\Theta_{xqy} := \frac{2\pi}{t} \sum_{i=1}^{t} X_{xqy}^{(i)}.$$

Um die Berechnung zu vereinfachen, definieren wir noch:

Definition 7.9.

$$L_q^{(i)} := \{x \in D \setminus \{q\} \mid (x - q) \cdot r_i < 0\},$$
$$R_q^{(i)} := \{x \in D \setminus \{q\} \mid 0 < (x - q) \cdot r_i\}.$$

Mit $L_q^{(i)}$ und $R_q^{(i)}$ bezeichnen wir also die Menge der Daten in den beiden unterschiedlichen Halbräumen, die durch den Zufallsvektor r_i erzeugt wurden.

[Pha12] arbeiten bei ihrem Algorithmus mit der normalen Varianz. Anders als [Kri08] in Definition 7.2 verwenden sie keine Gewichtsfaktoren:

Definition 7.10.

$$\text{linearABOF} : \begin{cases} \mathbb{R}^n \to \mathbb{R} \\ q \mapsto \mathbb{V}[\varphi_{xqy}] = \overline{\varphi_{xqy}^2} - \overline{\varphi}_{xqy}^2 \end{cases}.$$

Die beiden Werte $\overline{\varphi_{xqy}^2}$ und $\overline{\varphi}_{xqy}^2$ werden aber nicht exakt berechnet, sondern geschätzt. Für $\overline{\varphi}_{xqy}$ verwenden wir die Schätzfunktion $S_1(q)$:

$$S_1(q) := \frac{2}{(N-1)(N-2)} \sum_{\substack{\{x,y\} \\ x \in D\setminus\{q\} \\ y \in D\setminus\{q,x\}}} \Theta_{xqy}$$
$$= \frac{2}{(N-1)(N-2)} \sum_{\substack{\{x,y\} \\ x \in D\setminus\{q\} \\ y \in D\setminus\{q,x\}}} \sum_{i=1}^{t} \frac{2\pi}{t} X_{xqy}^{(i)}.$$

Es gilt $X_{xqy}^{(i)} = 1 \iff x \in L_q^{(i)} \wedge y \in R_q^{(i)}$. Letzteres ist in $\left|L_q^{(i)}\right| \left|R_q^{(i)}\right|$ Kombinationen gegeben. Wir erhalten aber die selbe Anzahl an Kombinationen, wenn wir den Zufallsvektor r_i um π drehen. Verwenden wir $\left|L_q^{(i)}\right| \left|R_q^{(i)}\right|$ für unsere Schätzfunktion gilt daher

$$S_1(q) = \frac{2\pi}{t(N-1)(N-2)} \sum_{i=1}^{t} \left|L_q^{(i)}\right| \left|R_q^{(i)}\right|. \tag{7.3}$$

Um $\overline{\varphi_{xqy}^2}$ zu schätzen, verwenden wir die Schätzfunktion $S_2(q)$. Um $S_2(q)$ zu berechnen, betrachten wir zunächst die restlichen Daten $x_1, x_2, \ldots, x_{N-1} \in D \setminus \{q\}$ und die beiden „Indikatorvektoren“ $u^{(i)}$ und $v^{(i)}$:

Definition 7.11.

$$u_q^{(i)} := (u_{q,1}^{(i)},\ u_{q,2}^{(i)}, \ldots,\ u_{q,N-1}^{(i)}),$$
$$v_q^{(i)} := (v_{q,1}^{(i)},\ v_{q,2}^{(i)}, \ldots,\ v_{q,N-1}^{(i)}),$$

wobei

$$u_{q,j}^{(i)} := \begin{cases} 1 & \text{falls } x_j \in L_q^{(i)} \\ 0 & \text{sonst} \end{cases},$$
$$v_{q,j}^{(i)} := \begin{cases} 1 & \text{falls } x_j \in R_q^{(i)} \\ 0 & \text{sonst} \end{cases},$$
$$A_q := \sum_{i=1}^{t} u_q^{(i)} \otimes v_q^{(i)}.$$

Die Vektoren $u_q^{(i)}$ und $v_q^{(i)}$ zeigen für ein Abfragedatum q und einen Zufallsvektor r_i an, ob ein bestimmtes Datum x_j im linken (bzw. rechten) Halbraum liegt oder nicht. Die Einträge der Matrix A_q zählen, wie oft bei einem bestimmten Datumspaar ein Datum im linken Halbraum und das andere Datum im rechten Halbraum liegt. Man beachte, dass $u_{q,j}^{(i)} \otimes v_{q,k}^{(i)} = 1 \iff X_{xqy}^{(i)} = 1$ (mit $x = x_j$ und $y = x_k$). Mit A_{xqy} bezeichnen wir den Eintrag in der Zeile von A_q, in der die Information über x steht, und in der Spalte, in der die Information über y steht.

$$
\begin{aligned}
A_{xqy}^2 &= \left(\sum_{i=1}^{t} X_{xqy}^{(i)}\right)^2, \\
\mathbb{E}[A_{xyz}^2] &= \sum_{i=1}^{t} \mathbb{E}\left[\left(X_{xqy}^{(i)}\right)^2\right] + \sum_{i=1}^{t}\sum_{\substack{j=1\\ j\neq i}}^{t} \mathbb{E}[X_{xqy}^{(i)}]\mathbb{E}[X_{xqy}^{(j)}] \\
&= t\frac{\Theta_{xqy}}{2\pi} + t(t-1)\left(\frac{\Theta_{xqy}}{2\pi}\right)^2.
\end{aligned}
$$

Bei der letzten Gleichung beachte man, dass die $X_{xqy}^{(i)}$ nur die Werte 0 oder 1 annehmen; daher ist $\left(X_{xqy}^{(i)}\right)^2 = X_{xqy}^{(i)}$. Wir erhalten eine Schätzfunktion für Θ_{xqy}^2:

$$
\Theta_{xqy}^2 = \frac{4\pi^2}{t(t-1)}\left(\mathbb{E}[A_{xqy}^2] - t\frac{\Theta_{xqy}}{2\pi}\right).
$$

$$
\begin{aligned}
S_2'(q) &= \frac{2}{(N-1)(N-2)} \sum_{\substack{\{x,y\}\\ x\in D\setminus\{q\}\\ y\in D\setminus\{q,x\}}} \Theta_{xqy}^2 \\
&= \frac{4\pi^2}{(N-1)(N-2)t(t-1)} \sum_{\substack{\{x,y\}\\ x\in D\setminus\{q\}\\ y\in D\setminus\{q,x\}}} 2\left(\mathbb{E}[A_{xqy}^2] - t\frac{\Theta_{xqy}}{2\pi}\right) \\
&= \frac{4\pi^2}{(N-1)(N-2)t(t-1)} \left(\sum_{\substack{\{x,y\}\\ x\in D\setminus\{q\}\\ y\in D\setminus\{q,x\}}} 2\mathbb{E}[A_{xqy}^2] - \sum_{\substack{\{x,y\}\\ x\in D\setminus\{q\}\\ y\in D\setminus\{q,x\}}} t\frac{\Theta_{xqy}}{\pi}\right) \\
&= \frac{4\pi^2}{(N-1)(N-2)t(t-1)} \left(\|A_q\|_F^2 - \frac{t}{2\pi}(N-1)(N-2)S_1(q)\right) \\
&= \frac{4\pi^2}{(N-1)(N-2)t(t-1)}\|A_q\|_F^2 - \frac{2\pi}{t-1}S_1(q). \qquad (7.4)
\end{aligned}
$$

Bezüglich der Umformung von $\mathbb{E}[A_{xqy}^2]$ zu $\|A_q\|_F^2$ beachte, dass in die Einträge über der Hauptdiagonale von A_q sich auf $X_{xqy}^{(i)}$ beziehen, die Einträge unter der Hauptdiagonale auf $X_{yqx}^{(i)}$.

Die Erwartungswerte der A_{xqy}^2 können wir ohne Kenntnis der Winkel nicht berechnen, daher schätzen wir die Erwartungswerte mit der Frobeniusnorm. Die Berechnung der Frobeniusnorm $\|\ \|_F$ von A_q hat allerdings eine Laufzeit von $\mathcal{O}(N^2)$. Der folgende Satz 7.14 ([Ind08]) liefern eine Schätzfunktion, die mit Laufzeit $\mathcal{O}(N)$ berechnet werden kann:

Definition 7.12. Ein Zufallsvektor $X = (X_1, \ldots, X_N) : \Omega \to \{-1, 1\}^N$ heißt *vierfach unabhängig*, wenn für alle $i_1, i_2, i_3, i_4 \in \{1, \ldots, N\}$ (i_1, i_2, i_3, i_4 paarweise verschieden) und alle $b_1, b_2, b_3, b_4 \in \{-1, 1\}$ gilt

$$\mathbb{P}\left[X_{i_1} = b_1, X_{i_2} = b_2, X_{i_3} = b_3, X_{i_4} = b_4\right] = 1/16.$$

Nach Noga Alon, Yossi Matias und Mario Szegedy ([Alo96]) ist der AMS Sketch benannt:

Definition 7.13. Seien $u, v \in \mathbb{R}^n$ sowie $s^{(1)}$ und $s^{(2)}$ zwei vierfach unabhängige Vektoren mit $s^{(1)}, s^{(2)} \in \{\pm 1\}^n$. Der *AMS Sketch* des dyadischen Produkts $u \otimes v$ ist definiert als

$$\mathrm{AMS} : \begin{cases} \mathbb{R}^{n \times n} \to \mathbb{R} \\ u \otimes v \mapsto \sum_{i=1}^n \sum_{j=1}^n s_i^{(1)} s_j^{(2)} u_i v_j = \left(\sum_{i=1}^n s_i^{(1)} u_i\right)\left(\sum_{j=1}^n s_j^{(2)} v_j\right) \end{cases}.$$

Satz 7.14. *Seien u und v wie in Definition 7.13. Dann gilt:*

$$\mathbb{E}\left[\mathrm{AMS}^2\left(u \otimes v\right)\right] = \|u \otimes v\|_F^2,$$

d. h. der Erwartungswert von $\mathrm{AMS}^2(u \otimes v)$ ist gleich der quadrierten Frobeniusnorm des dyadischen Produkts $u \otimes v$. Für die Varianz gilt die Abschätzung

$$\mathbb{V}\left[\mathrm{AMS}^2\left(u \otimes v\right)\right] < 8\,\mathbb{E}\left[\mathrm{AMS}^2\left(u \otimes v\right)\right]^2.$$

Beweis. Die Vektoren $s^{(1)}$ und $s^{(2)}$ sind vierfach unabhängig, daher sind die Einträge $s_i^{(1)}$ und $s_j^{(2)}$ für beliebige Indizes i und j unabhängig. Somit gilt:

$$\begin{aligned}
\mathbb{E}\left[\mathrm{AMS}^2\left(u \otimes v\right)\right] &= \mathbb{E}\left[\sum_{i=1}^{n-1}\sum_{j=1}^{n-1}\left(s_i^{(1)} s_j^{(2)} u_i v_j\right)^2\right] \\
&= \sum_{i=1}^{n-1}\sum_{j=1}^{n-1}\mathbb{E}\left[\left(s_i^{(1)}\right)^2\right]\mathbb{E}\left[\left(s_j^{(2)}\right)^2\right](u_i v_j)^2 \\
&= \sum_{i=1}^{n-1}\sum_{j=1}^{n-1}(u_i v_j)^2 \\
&= \|u \otimes v\|_F^2.
\end{aligned}$$

Die Ungleichung zwischen Varianz und Erwartungswert wird in [Bra10] bewiesen. □

In unserem Fall folgt daraus (mit $A := u \otimes v$):

$$\mathbb{E}[\mathrm{AMS}^2\left(u \otimes v\right)] = \|A\|_F^2.$$

Unsere Matrix A_q selbst ist allerdings nicht das Ergebnis eines dyadischen Produkts, sondern eine Summe von Ergebnissen von dyadischen Produkten. Wir können den Satz daher zunächst nicht anwenden. Es gilt allerdings auch:

Satz 7.15. *Seien $u^{(k)} \in \mathbb{R}^n$ und $v^{(k)} \in \mathbb{R}^n$, $1 \leq k \leq t$. Dann gilt:*

$$\mathbb{E}\left[\left(\sum_{k=1}^{t} \mathrm{AMS}\left(u^{(k)} \otimes v^{(k)}\right)\right)^2\right] = \left\|\sum_{k=1}^{t} u^{(k)} \otimes v^{(k)}\right\|_F^2,$$

Beweis.

$$\begin{aligned}\mathbb{E}\left[\left(\sum_{k=1}^{t} \mathrm{AMS}\left(u^{(k)} \otimes v^{(k)}\right)\right)^2\right] &= \mathbb{E}\left[\left(\sum_{k=1}^{t}\sum_{i=1}^{n-1}\sum_{j=1}^{n-1} s_i^{(1)} s_j^{(2)} u_i^{(k)} v_j^{(k)}\right)^2\right] \\ &= \mathbb{E}\left[\left(\sum_{i=1}^{n-1}\sum_{j=1}^{n-1} s_i^{(1)} s_j^{(2)} \sum_{k=1}^{t} u_i^{(k)} v_j^{(k)}\right)^2\right].\end{aligned}$$

Wir multiplizieren die Klammer aus und fassen die quadratischen Terme in einer eigenen Summe zusammen:

$$\begin{aligned}&= \mathbb{E}\left[\sum_{i=1}^{n-1}\sum_{j=1}^{n-1}\left(s_i^{(1)} s_j^{(2)} \sum_{k=1}^{t} u_i v_j\right)^2 + \right. \\ &\qquad \left. \sum_{g=1}^{n-1}\sum_{h=1}^{n-1} \underbrace{\sum_{i=1}^{n-1}\sum_{j=1}^{n-1}}_{i\neq g \vee j \neq h} s_g^{(1)} s_h^{(2)} \left(\sum_{k=1}^{t} u_g^{(k)} v_h^{(k)}\right) s_i^{(1)} s_j^{(2)} \left(\sum_{k=1}^{t} u_i^{(k)} v_j^{(k)}\right)\right] \\ &= \sum_{i=1}^{n-1}\sum_{j=1}^{n-1} \mathbb{E}\left[\left(s_i^{(1)}\right)^2\right] \mathbb{E}\left[\left(s_j^{(2)}\right)^2\right] \left(\sum_{k=1}^{t} u_i v_j\right)^2 + \\ &\qquad \sum_{g=1}^{n-1}\sum_{h=1}^{n-1} \underbrace{\sum_{i=1}^{n-1}\sum_{j=1}^{n-1}}_{i\neq g \vee j \neq h} \mathbb{E}\left[s_g^{(1)} s_h^{(2)} s_i^{(1)} s_j^{(2)}\right] \left(\sum_{k=1}^{t} u_g^{(k)} v_h^{(k)}\right)\left(\sum_{k=1}^{t} u_i^{(k)} v_j^{(k)}\right).\end{aligned}$$

Betrachten wir den Ausdruck $\mathbb{E}\left[s_g^{(1)} s_h^{(2)} s_i^{(1)} s_j^{(2)}\right]$. Da entweder $i \neq g$ oder $j \neq h$ gilt, sind zumindest zwei der Zufallsvariablen unabhängig. Für diese unabhängigen Zufallsvariablen ist der Erwartungswert 0 und daher auch die gesamte Vierfachsumme:

$$\begin{aligned}&= \sum_{i=1}^{n-1}\sum_{j=1}^{n-1}\left(\sum_{k=1}^{t} u_i v_j\right)^2 + 0 \\ &= \left\|\sum_{k=1}^{t} u^{(k)} \otimes v^{(k)}\right\|_F^2.\end{aligned}$$

□

Algorithmus 11 : Near-linear Angle-Based Outlier Detection

Input : Daten D, Anzahl der Iterationen t, w_1, w_2
Output : linearABOF(q) $\forall q \in D$

1 $N \leftarrow |D|$;
2 $R[1,t] \leftarrow$ erzeuge t Zufallsvektoren;
3 **for** $i \leftarrow 1$ **to** t **do**
4 | **foreach** $q \in D$ **do**
5 | | $Q_i[q] \leftarrow q \cdot R[i]$;
6 | **end**
7 | $L_i \leftarrow$ IndicesOf(sort$(Q_i[q], ascending)$);
8 **end**
9 **for** $i \leftarrow 1$ **to** t **do**
10 | **foreach** $q \in D$ **do**
11 | | $S1[q] \leftarrow S_1(q)$; // $S_1(q)$ lt. Formel 7.3, mithilfe von L_i
12 | **end**
13 **end**
14 **for** $i \leftarrow 1$ **to** w_2 **do**
15 | **foreach** $q \in D$ **do**
16 | | $Y_i[q] \leftarrow \frac{1}{w_1} \sum_{k=1}^{w_1} S_2(q)$; // $S_2(q)$ lt. Formel 7.5, mithilfe von L_i
17 | **end**
18 **end**
19 **foreach** $q \in D$ **do**
20 | $S2[q] \leftarrow$ median$(Y_i[q])$;
21 | $S2[q] \leftarrow \frac{4\pi^2}{(N-1)(N-2)t(t-1)} S2[q] - \frac{2\pi}{t-1} S1[q]$;
22 | linearABOF$[q] \leftarrow S2[q] - (S1[q])^2$;
23 **end**

Wir können also $\|A_q\|_F^2$ schätzen, indem wir w_1 Zufallsvektoren $s^{(1)}$ und $s^{(2)}$ erzeugen, mithilfe der Funktion AMS w_1 Näherungswerte berechnen und von diesen Näherungswerten den Mittelwert als Schätzer verwenden:

$$S_2(q) = \frac{4\pi^2}{(N-1)(N-2)t(t-1)} \left(\frac{1}{w_1} \sum_{j=1}^{w_1} \left(\sum_{i=1}^{t} \text{AMS}(u_q^{(i)} \otimes v_q^{(i)}) \right)^2 \right) - \frac{2\pi}{t-1} S_1(q). \quad (7.5)$$

Algorithmus 11 stellt das Verfahren in Pseudocode dar. In der Arbeit von [Pha12] wird der Code detaillierter beschrieben, insbesondere wie man die Daten effizient mithilfe der Listen L_i, $i = 1, \ldots t$ speichern und die Werte der Schätzer berechnen kann.

Zunächst erzeugt der Algorithmus t Zufallsvektoren und speichert diese Vektoren in der $t \times n$-Matrix R. In der Schleife von Zeile 3 bis Zeile 8 erzeugen wir t Listen L_i. In den Listen L_i stehen die Indizes der einzelnen Punkte sortiert nach dem Skalarprodukt $x \cdot r_i$.

Mithilfe der Listen L_i können wir die Größen $\left|L_q^{(i)}\right|$ und $\left|R_q^{(i)}\right|$ und damit $S_1(q)$ in

Berechnung L	$\mathcal{O}(t(N + N\log(N)))$	Zeilen 5 und 7
Berechnung $S1$	$\mathcal{O}(tN)$	Zeile 11
Berechnung S_2	$\mathcal{O}(tN)$	Zeile 16
Berechnung Y	$\mathcal{O}(w_2 w_1 S_2)$	Zeile 16
Berechnung $S2$	$\mathcal{O}(Nw_2)$	Zeilen 19–23

Tabelle 7.1: Laufzeit der einzelner Zeilen von Algorithmus 11

linearer Zeit berechnen (in Zeile 11): Hat eine bestimmte Liste z. B. 100 Einträge, so ist für den 20. Eintrag $\left|L_q^{(i)}\right| = 19$ und $\left|R_q^{(i)}\right| = 80$.

Genauso können wir die Schätzfunktion $S_2[q]$ mit L_i in linearer Zeit berechnen: Bezogen auf z. B. den 20. Eintrag in einer Liste sind die Einträge von $u_q^{(i)}$ für die 19 Daten davor 1 und für die 80 Daten danach 0. Denn für die 19 Einträge davor gilt $x \cdot r_i < q \cdot r_i$, für die 80 Daten danach entsprechend $q \cdot r_i < x \cdot r_i$. Bei $v_q^{(i)}$ ist es umgekehrt. Die Terme $\left(\sum_{i=1}^{n} s_i^{(1)} u_i\right)$ und $\left(\sum_{j=1}^{n} s_j^{(2)} v_j\right)$ aus Definition 7.13 berechnen wir, indem wir in der von L_i gegebenen Reihenfolge durch die Daten wandern und zufällig $+1$ oder -1 zu einer laufenden Summe addieren.

Zusammengefasst ergibt sich (siehe Tabelle 7.1)

$$\mathcal{O}(tN + tN\log(N) + tN + tNw_1w_2 + Nw_2) = \mathcal{O}(tN\log(N) + N(t + tw_1w_2 + w_2)).$$

[Pha12] arbeiten mit $t = N\log(N)$ um eine bestimmte Genauigkeit der Nährung garantieren zu können (mit einer gewissen Wahrscheinlichkeit). Die Laufzeit des Algorithmus beträgt dann

$$\mathcal{O}(N\log^2(N) + Nw_1w_2\log(N) + N\log(N) + Nw_2).$$

Abhängig von der Wahl von w_1 und w_2 gilt $\log(N) \leq w_1w_2$ oder $\log(N) \geq w_1w_2$. Die Laufzeit des Algorithmus beträgt also $\mathcal{O}(N\log^2(N))$ oder $\mathcal{O}(Nw_1w_2\log(N))$.

In [Pha12] werden die Sätze 7.17 und 7.18 hergeleitet, die bei einer hinreichenden Anzahl von Iterationen eine bestimmte Genauigkeit der Schätzfunktionen S_1 und S_2' garantieren. Für die Beweise verwenden wir folgende Version der Chernoff-Abschätzung [Dub09, S. 23]:

Satz 7.16. *Sei $X = \sum_{i=1}^{t} X^{(i)}$ eine Summe von unabhängigen Zufallsvariablen mit Werten im Intervall* [0,1]. *Für jede positive, reelle Zahl $\varepsilon \in \mathbb{R}_+$ gilt*

$$\mathbb{P}\left[|X - \mathbb{E}X| > \varepsilon\right] \leq 2e^{-2\varepsilon^2/t}.$$

Satz 7.17. *Sei $N \geq 3$ die Anzahl der vorhandenen Daten, $\varepsilon \in \mathbb{R}_+$ eine positive, reelle Zahl, $S_1(q)$ ist definiert wie in Gleichung 7.3. Dann gilt für $t > \varepsilon^{-2}\log(N)$ die Abschätzung*

$$\mathbb{P}\left[\left|\frac{S_1(q)}{2\pi} - \frac{\varphi}{2\pi}\right| > \varepsilon\right] \leq 2/N^2.$$

Beweis. Laut Gleichung 7.3 gilt:

$$\frac{tS_1(q)}{2\pi} = \sum_{i=1}^{t} \frac{\left|L_q^{(i)}\right| \left|R_q^{(i)}\right|}{(N-1)(N-2)}.$$

Falls $\left|L_q^{(i)}\right| = N-1$, ist $\left|R_q^{(i)}\right| = 0$. Falls $\left|L_q^{(i)}\right| \leq N-2$, ist $\left|L_q^{(i)}\right|/(N-2) \leq 1$. Für $\left|R_q^{(i)}\right|$ gilt das Gleiche, daher sind die einzelnen Summanden ≤ 1. Wir können also Satz 7.16 auf die Gleichung anwenden:

$$\begin{aligned}
\mathbb{P}\left[\left|\frac{tS_1(q)}{2\pi} - \mathbb{E}\left[\frac{tS_1(q)}{2\pi}\right]\right| > \delta\right] &\leq 2e^{-2\delta^2/t}, \\
\mathbb{P}\left[\left|\frac{tS_1(q)}{2\pi} - \frac{t\varphi}{2\pi}\right| > t\varepsilon\right] &\leq 2e^{-2(t\varepsilon)^2/t}, \\
\mathbb{P}\left[\left|\frac{S_1(q)}{2\pi} - \frac{\varphi}{2\pi}\right| > \varepsilon\right] &\leq 2e^{-2t\varepsilon^2}.
\end{aligned}$$

Wählen wir $t > \varepsilon^{-2}\log(N)$ folgt

$$\mathbb{P}\left[\left|\frac{S_1(q)}{2\pi} - \frac{\varphi}{2\pi}\right| > \varepsilon\right] \leq 2/N^2.$$

□

Satz 7.18. *Sei $N \geq 3$ die Anzahl der vorhandenen Daten, $\varepsilon \in \mathbb{R}_+$ eine positive Zahl, $S_2'(q)$ ist definiert wie in Gleichung 7.4. Dann gilt für $t > 16\varepsilon^{-2}\pi^4\log(N)$ die Abschätzung*

$$\mathbb{P}\left[\left|S_2'(q) - \mathbb{E}\left[S_2'(q)\right]\right| > \varepsilon\right] \leq 2/N^2.$$

Beweis. Wir analysieren zunächst den Summanden $\frac{4\pi^2}{(N-1)(N-2)t(t-1)}\|A_q\|_F^2$ von $S_2'(q)$. Dieser Summand weicht um mehr als eine positive Zahl $\varepsilon \in \mathbb{R}_+$ vom Erwartungswert ab, falls $\|A_q\|_F^2$ um mehr als $(N-1)(N-2)t(t-1)/(4\pi^2)$ abweicht. Wir zeigen daher, dass jeder Eintrag von A_{xqy}^2 mit einer bestimmten Wahrscheinlichkeit um nicht mehr als $\varepsilon t^2/(4\pi^2)$ abweicht. Wir bemerken noch, dass die Einträge der Hauptdiagonale von $\|A_q\|_F^2$ alle 0 sind und nur die verbleibenden $(N-1)(N-2)$ Einträge von A_{xqy} einen Beitrag liefern können.

Für einen einzelnen Eintrag von A_q gilt (Definition 7.11)

$$A_{xqy} = \sum_{i=1}^{t} X_{xqy}^{(i)},$$

wir können daher Satz 7.16 darauf anwenden:

$$\mathbb{P}\left[\left|A_{xqy} - \mathbb{E}\left[A_{xqy}\right]\right| > \frac{t\varepsilon}{4\pi^2}\right] \leq 2e^{-2\left(\frac{t\varepsilon}{4\pi^2}\right)^2/t}.$$

Wählen wir $t > 16\varepsilon^{-2}\pi^4 \log(N)$ gilt

$$\mathbb{P}\left[\left|A_{xqy} - \frac{t\varphi}{2\pi}\right| > \frac{t\varepsilon}{4\pi^2}\right] \leq 2/N^2. \tag{7.6}$$

Weiters gilt $A_{xqy} \leq \sum_{i=2}^{t} 1 = t$; daher weicht A_{xqy}^2 maximal um $\varepsilon t^2/(4\pi^2)$ vom Erwartungswert ab. Verwenden wir in der Gleichung des Satzes statt ε die Schranke $\varepsilon/2$ und in Satz 7.17 statt ε die Schranke $(t-1)\varepsilon/(4\pi)$, folgt die Behauptung. □

8 Implementierung und Vergleich der Algorithmen

Die in dieser Arbeit vorgestellten Algorithmen wurden hauptsächlich in R [RCT12] implementiert. Programmteile, die für die Laufzeit wesentlich sind, wurden in C programmiert. Berechnet wurden die Ergebnisse auf einem Intel Core i3-2100T-Prozessor mit 2.50 MHz und 4 GiB RAM.

8.1 Arrhythmia-Datensatz

Dieser Datensatz stammt vom *UCI Machine Learning Repository* ([Lic13], [https://archive.ics.uci.edu/ml/datasets/Arrhythmia]). Er beinhaltet die Untersuchungsergebnisse mittels Elektrokardiogramm von 452 Personen, die auf Anzeichen von Herzrhythmusstörungen (Arrhythmie) untersucht wurden. Die Daten von 245 Personen weisen keine Anzeichen auf. Die Daten von 185 Personen weisen eine von vierzehn möglichen Arten von Herzrhythmusstörung auf. Bei 22 Personen konnte keine Diagnose gestellt werden.

Jedes Datum besitzt 280 Attribute. 60 Attribute sind nominalskaliert und wurden daher vor der Analyse entfernt. Auffallend ist, dass bei ungefähr 40 % der verbleibenden 220 Attributen das jeweilige Attribut lediglich vier oder weniger unterschiedliche Werte annimmt. Vor der Ausreißeranalyse werden die Daten mittels

$$x_{*j} \mapsto \frac{x_{*j} - \min\limits_{i\in\{1,\dots,|D|\}} x_{ij}}{\max\limits_{i\in\{1,\dots,|D|\}} x_{ij} - \min\limits_{i\in\{1,\dots,|D|\}} x_{ij}}, \qquad j = 1,\dots,n \tag{8.1}$$

in den Einheitswürfel abgebildet.

Für unsere Analyse der Algorithmen betrachten wir alle Daten der 245 Personen ohne Herzrhythmusstörung. Um Ausreißer hinzuzufügen verwenden wir folgende Methode: Wir fügen bis zu 10 zufällig gewählte Personen mit Herzrhythmusstörung hinzu (Datensätze 1, 178, 2, 9, 3, 6, 4, 148, 5 und 79).

8.1.1 Laufzeit

Wir beschäftigen uns zunächst mit der Laufzeit der Algorithmen, abhängig von den Parametern. Die Ergebnisse sind in den Tabellen 8.2 bis 8.4 und in den Abbildungen 8.1 bis 8.4 dargestellt. Da die Größenordnungen der Laufzeiten unterschiedlich sind, sind in den Abbildungen die relativen Werte aufgetragen. Die Grundwerte ergeben sich aus dem jeweils kleinsten Parameterwert. Die fixen Werte der restlichen Parameter sind in Tabelle 8.1 aufgelistet.

Beim Algorithmus Hilbert OD spielt die Anzahl der Durchläufe der Funktion Scan eine wesentliche Rolle für die Laufzeit. Daher wurde diese Zahl gemeinsam mit der durchschnittlichen Laufzeit eines Durchlaufs von Scan in die Tabellen aufgenommen. In den Abbildungen sind die durchschnittlichen Laufzeiten eines Durchlaufs aufgetragen.

Die Änderungen der Laufzeiten mit den Parametern entsprechen nur ungefähr den theoretisch erwarteten Änderungen. Aufgrund der geringen Datenmenge sind teilweise konkrete Details in den Implementierungen der Algorithmen von größerer Bedeutung. Die Laufzeit von Hilbert OD gehört z. B. der Klasse $\mathcal{O}(N \log N)$ an, da die N Hilbertwerte sortiert werden müssen. Bei nur wenigen hundert Daten dauert das Sortieren aber um ein Vielfaches kürzer als das Berechnen der Hilbertwerte.

Die Laufzeit des Algorithmus ABOD wächst wie erwartet schnell mit der Anzahl der Daten. Bei nur 100 Daten ist der absolute Wert zwar noch klein, ist die Datenmenge aber nur etwas größer, ist der Algorithmus praktisch nicht mehr verwendbar.

Die Laufzeit des Algorithmus Hilbert OD ändert sich ungefähr nur linear, und nicht mit $\mathcal{O}(N \log N)$, mit der Anzahl der Daten und der Dimension, ist aber bereits bei den wenigen Daten absolut weitaus größer als bei den anderen Algorithmen. Die Ursache dafür ist die schlechte Implementierung der Berechnung des Abstands in den Zeilen 14–17 im Algorithmus InnerScan (S. 48).

Evolutionary OD		Hilbert OD		near-linear ABOD	
Parameter	Wert	Parameter	Wert	Parameter	Wert
m	50	$\mathfrak{n}$	3	t	100
ϕ	7	k	4	w_1	50
κ	2	h	3	w_2	50
p_1	0.1				
p_2	0.1				

Tabelle 8.1: Fix gesetzte Parameter, Laufzeitanalyse, Datensatz *Arrhythmia*

Anzahl Daten	Evolutionary OD	Hilbert OD	ABOD	near-linear ABOD
10	0.95	$1.72 \cdot 3$	0.01	0.15
20	1.20	$3.58 \cdot 3$	0.02	0.19
30	1.19	$3.30 \cdot 4$	0.06	0.26
40	1.10	$4.62 \cdot 4$	0.13	0.30
50	1.22	$5.38 \cdot 4$	0.24	0.34
60	1.15	$7.03 \cdot 4$	0.41	0.38
70	1.19	$8.33 \cdot 4$	0.64	0.41
80	1.24	$14.37 \cdot 3$	0.95	0.45
90	1.35	$15.96 \cdot 3$	1.34	0.48
100	1.14	$18.41 \cdot 3$	1.83	0.52

Tabelle 8.2: Laufzeit der Algorithmen in Sekunden, bei Hilbert OD inkl. Anzahl der Durchläufe der Funktion Scan. Datensatz *Arrhythmia*, Dimension $n = 100$.

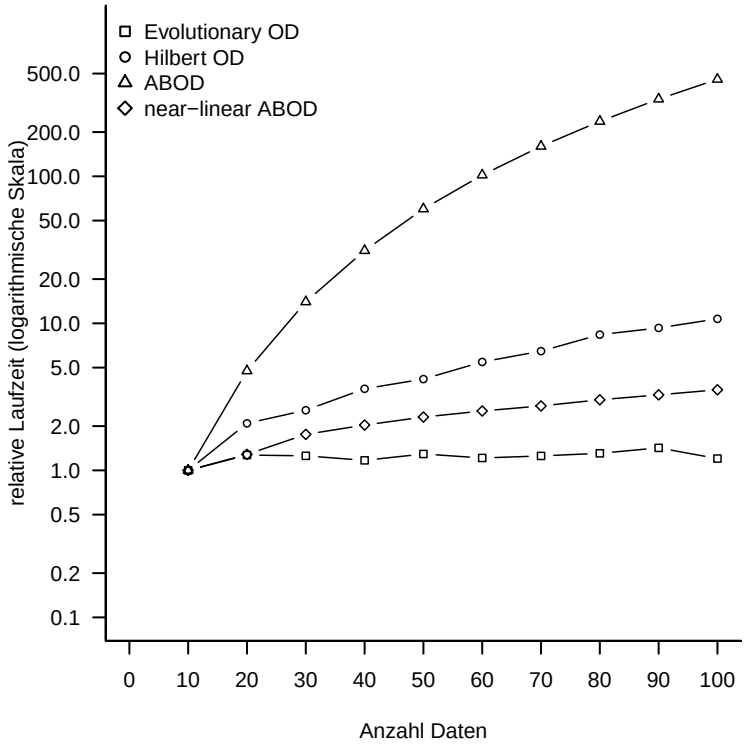

Abbildung 8.1: Laufzeit der Algorithmen, bei Hilbert OD die durchschnittliche Laufzeit der Funktion Scan. Datensatz *Arrhythmia*, Dimension $n = 100$.

Dimensionen	Evolutionary OD	Hilbert OD	ABOD	near-linear ABOD
10	0.87	$2.65 \cdot 3$	0.24	0.54
20	0.79	$4.07 \cdot 4$	0.41	0.54
30	0.84	$6.46 \cdot 4$	0.60	0.57
40	0.90	$8.01 \cdot 4$	0.77	0.58
50	0.90	$9.17 \cdot 3$	0.97	0.67
60	1.01	$12.04 \cdot 3$	1.15	0.57
70	1.00	$11.95 \cdot 3$	1.35	0.62
80	1.12	$3.99 \cdot 3$	1.49	0.57
90	1.27	$16.21 \cdot 3$	1.66	0.54
100	1.18	$19.49 \cdot 3$	1.83	0.56
110	1.35	$18.52 \cdot 3$	1.98	0.57
120	1.38	$16.36 \cdot 4$	2.16	0.59
130	1.19	$22.86 \cdot 3$	2.31	0.61
140	1.48	$28.24 \cdot 3$	2.48	0.57
150	1.56	$23.05 \cdot 4$	2.66	0.57
160	1.48	$24.54 \cdot 4$	2.83	0.57
170	1.62	$28.69 \cdot 4$	2.99	0.62
180	1.68	$30.82 \cdot 4$	3.19	0.59
190	1.81	$30.44 \cdot 4$	3.63	0.60
200	1.80	$32.35 \cdot 4$	3.91	0.57

Tabelle 8.3: Laufzeit der Algorithmen in Sekunden, bei Hilbert OD inkl. Anzahl der Durchläufe der Funktion Scan. Datensatz *Arrhythmia*, Anzahl der Daten $N = 100$.

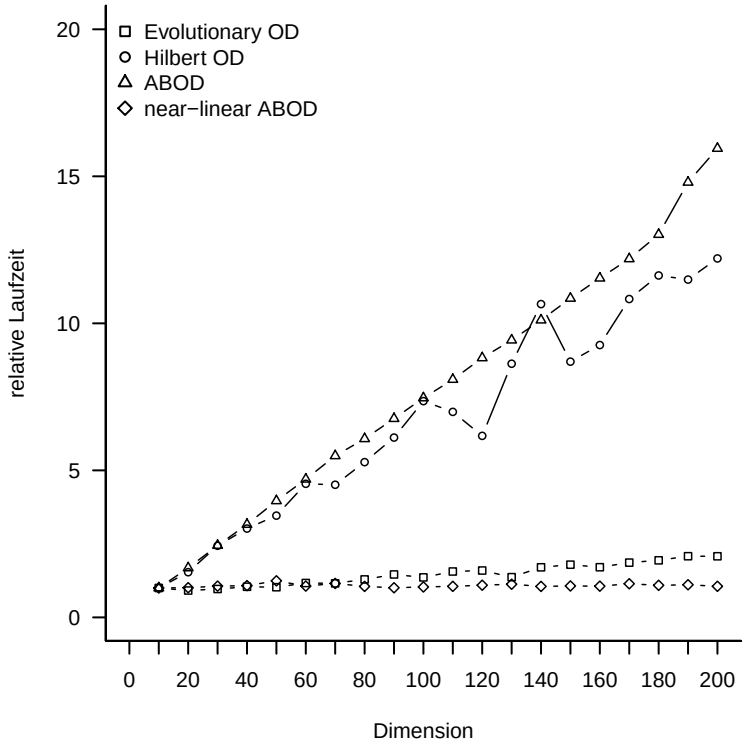

Abbildung 8.2: Laufzeit der Algorithmen, bei Hilbert OD die durchschnittliche Laufzeit der Funktion Scan. Datensatz *Arrhythmia*, Anzahl der Daten $N = 100$.

	Evolutionary OD			Hilbert OD	
Wert	m	ϕ	κ	k	h
1	0.03	8.02	8.48	$10.08 \cdot 5$	$16.63 \cdot 3$
2	0.21	8.31	10.06	$14.19 \cdot 3$	$16.82 \cdot 3$
3	0.16	8.61	9.21	$13.43 \cdot 4$	$18.17 \cdot 3$
4	0.61	8.37	4.87	$17.36 \cdot 3$	$19.28 \cdot 3$
5	0.46	10.27	1.97	$19.75 \cdot 3$	$20.53 \cdot 3$
6	5.67	9.28	6.84	$19.88 \cdot 3$	$21.25 \cdot 3$
7	6.42	10.32		$21.21 \cdot 3$	
8	7.28	10.54		$23.08 \cdot 3$	
9	8.78	10.39		$26.33 \cdot 3$	
10	9.65			$26.35 \cdot 3$	

Tabelle 8.4: Laufzeit der Algorithmen in Sekunden, bei Hilbert OD inkl. Anzahl der Durchläufe der Funktion Scan. Datensatz *Arrhythmia*.

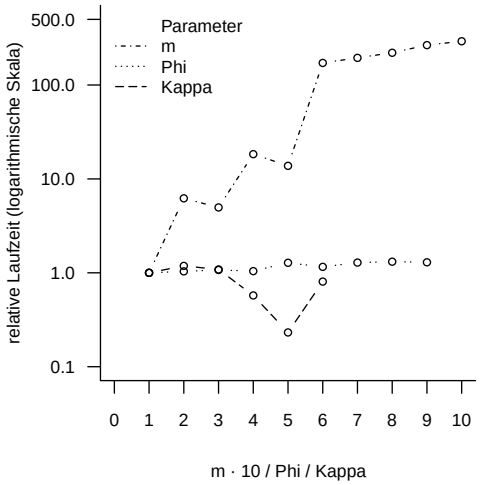

Abbildung 8.3: Laufzeit Evolutionary OD in Sekunden in Abhängigkeit der Parameter m, ϕ und κ. Datensatz *Arrhythmia*, Anzahl der Daten $N = 100$, Dimension $n = 100$.

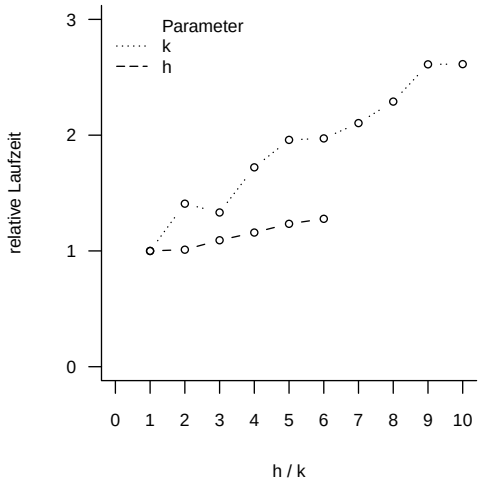

Abbildung 8.4: Laufzeit Hilbert OD in Sekunden in Abhängigkeit der Parameter k und h. Datensatz *Arrhythmia*, Anzahl der Daten $N = 100$, Dimension $n = 100$.

Evolutionary OD		Hilbert OD		near-linear ABOD	
Parameter	Wert	Parameter	Wert	Parameter	Wert
m	100	$\mathfrak{n}$	50	t	100
ϕ	5	k	5	w_1	50
κ	2	h	3	w_2	50
p_1	0.1				
p_2	0.1				

Tabelle 8.5: Fix gesetzte Parameter, Analyse der Qualität, Datensatz *Arrhythmia*

8.1.2 Qualität der Resultate

Für die Berechnungen in diesem Abschnitt wurden, wie zu Beginn des Abschnitts 8.1 beschrieben, alle 255 Daten und alle 220 Dimensionen verwendet. Die für die jeweiligen Algorithmen spezifischen Parameter sind in Tabelle 8.5 aufgelistet.

Um die Qualität der Resultate bewerten zu können, verwenden wir den Recallwert aus [Kri08]:

Definition 8.1. Gegeben sei die Liste der Daten $(d_1, d_2, \ldots, d_N), d_i \in D$, absteigend geordnet nach einer Bewertung (Score), die bestimmt, inwieweit ein Datum ein Ausreißer ist (z. B. der ABOF). Die Liste der Ausreißer $(q_1, q_2, \ldots, q_{\mathfrak{n}})$ sei nach der gleichen Bewertung geordnet und eine Teilauswahl der Liste $(d_1, d_2, \ldots, d_N)$. Welche Daten Ausreißer

Datensatz-Id	Evolutionary OD	Hilbert OD	ABOD	near-linear ABOD
1	112	–	57	71
178	96	24	18	8
2	191	–	188	224
9	150	37	73	39
3	135	7	7	13
6	63	–	170	191
4	56	–	27	41
148	18	1	1	1
5	32	2	2	2
79	97	26	30	33

Tabelle 8.6: Position der Ausreißer, Datensatz *Arrhythmia*.

sind, muss a-priori bekannt sein. Der *Recallwert* eines Ausreißers $q_i = d_j$ ist definiert als:

$$\mathrm{rec}\,(q_i) := \frac{i}{j}.$$

Abbildung 8.5 stellt die Recallwerte der Algorithmen dar. Der Algorithmus Evolutionary OD liefert keine Bewertung, sondern nur eine Klassifizierung (Label), ob ein Datum ein Ausreißer ist oder nicht. Für diesen Algorithmus kann man daher keine Recallwerte berechnen.

Um diese Einschränkung zu umgehen, wählen wir folgende Vorgehensweise: Wir starten den Algorithmus 100-mal und zählen für jedes Datum, wie oft das Datum in einem schwach besetzten Quader liegt. Je größer diese Anzahl ist, desto eher bewerten wir das Datum als Ausreißer. Ein Quader gilt für diese Analyse als schwach besetzt, wenn er weniger als 10 Daten enthält.

Der Recallwert von 1 für die ersten beiden Ausreißer bei den Algorithmen Hilbert OD, ABOD und near-linear ABOD zeigt dass diese drei Algorithmen zwei Ausreißer an die ersten beiden Stellen reihen. Der dritte Ausreißer kommt an Stelle 7 bzw. 8 (Recallwerte 3/7 bzw. 3/8). In Tabelle 8.6 stehen die Positionen der Ausreißer, die Daten wurden nach der Anzahl der Quader, d^k, ABOF bzw. linearABOF geordnet. Der Algorithmus Hilbert OD liefert nur die 50 Daten mit der höchsten Bewertung, unter diesen befinden sich 6 der 10 Ausreißer.

Sowohl Abbildung 8.5 als auch Tabelle 8.6 zeigen, dass die gefundenen Lösungen von Hilbert OD, ABOD und near-linear ABOD ausgesprochen ähnlich sind. Um die Ähnlichkeit der Ergebnisse zweier Algorithmen zu beurteilen, berechnen wir das Verhältnis

$$\frac{|A_{alg1} \cap A_{alg2}|}{m}, \quad A_{alg1}, A_{alg2} \subseteq D, \quad |A_{alg1}| = |A_{alg2}| = m,$$

wobei die Mengen A_{alg1} und A_{alg2} jene Daten beinhalten, die die Algorithmen $alg1$ und $alg2$ als Top-m-Ausreißer liefern. Abbildungen 8.6 und 8.7 stellen diese Verhältnisse für verschiedene Werte von m zwischen 10 und 50 dar. Ungefähr 80–90 % der Daten werden von Hilbert OD, ABOD und near-linear ABOD an eine ähnliche Position gereiht.

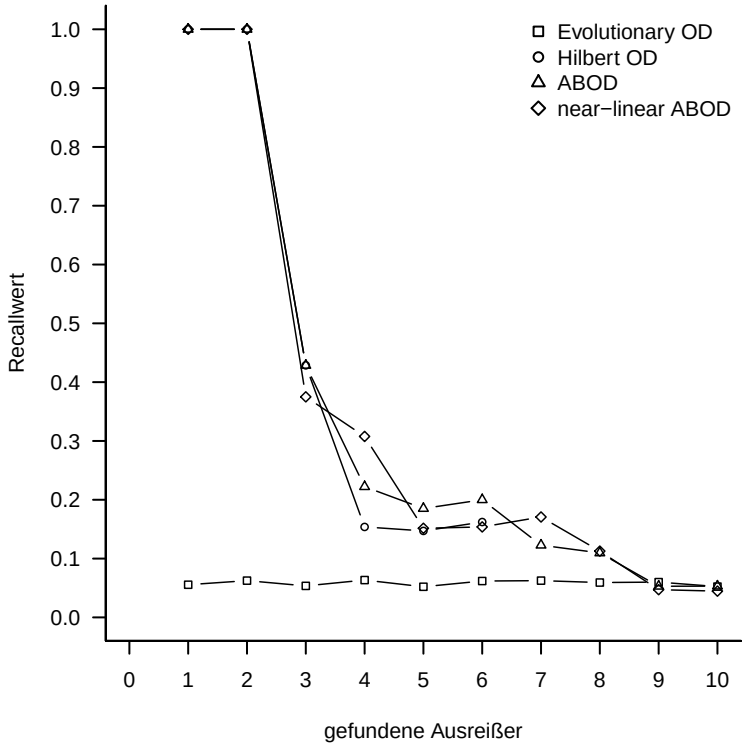

Abbildung 8.5: Recallwerte der Algorithmen, Datensatz *Arrhythmia*, Anzahl der Daten $N = 255$, Dimensionen $n = 220$.

Im Gegensatz dazu steht der Algorithmus Evolutionary OD. Dieser Algorithmus liefert weitaus schlechtere Ergebnisse. Alle Ausreißer wurden an hintere Stellen gereiht. Für eine nähere Analyse sind fünf Ergebnisse detaillierter in Tabelle 8.7 zusammengefasst.

Betrachtet man die Ergebnisse von zwei Durchläufen dieses Algorithmus, unterscheiden sich bereits diese beiden Ergebnisse völlig voneinander. In den fünf Ergebnisse befinden sich 95 unterschiedliche Daten. Die meisten, 69, befinden sich in lediglich einem der fünf Ergebnisse. 23 Daten wurden in 2 Durchläufen als Ausreißer klassifiziert, 2 Daten in 3 Durchläufen und ein Datum in 4 Durchläufen.

Unter den Daten, die Evolutionary OD als Ausreißer vorschlägt, befinden sich nur relativ wenige tatsächliche Ausreißer, teilweise auch überhaupt keine. Mangels eines Scores gibt es keine Möglichkeit, innerhalb der zurückgelieferten Daten eine Reihung vorzunehmen. Die Ergebnisse unterscheiden sich deutlich von denen der drei anderen Algorithmen, lediglich 10–20 % der Daten mit dem höchsten Score stimmen überein, der Wert steigt später auf ca. 40 % (siehe Abbildung 8.7).

Zusammenfassend ergibt sich folgendes Bild: Das Resultat von Evolutionary OD ist kaum brauchbar und weicht stark vom Rest ab. Die Resultate der drei anderen Algorithmen sind besser und sehr ähnlich. Die beiden Daten 5 und 148 wurden immer mit dem höchsten Score versehen und eindeutig als Ausreißer erkannt. Der Ausreißer 3 erhielt

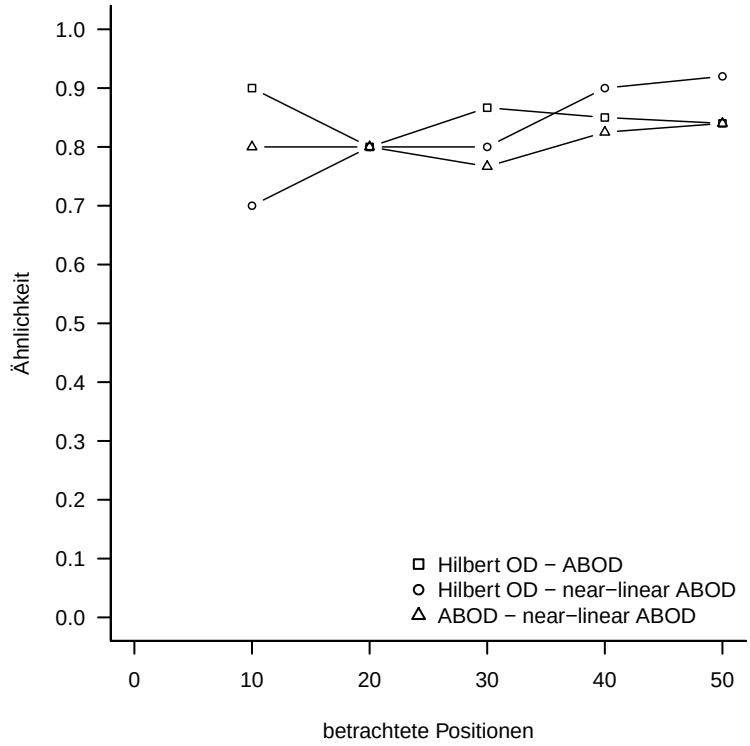

Abbildung 8.6: Ähnlichkeit der Algorithmen, Datensatz *Arrhythmia*, Anzahl der Daten $N = 255$, Dimensionen $n = 220$.

ebenfalls von allen drei Algorithmen ein hohes Score. Bei den anderen Ausreißern stellt sich die Frage, ob man sie mit den vorhanden Daten überhaupt als Ausreißer erkennen kann. Eventuell ist dazu zusätzliches Expertenwissen notwendig.

Bei ähnlichen Resultaten ist near-linear ABOD mit 1.3 s deutlich schneller als ABOD (62 s) und Hilbert OD (1617 s). Insgesamt kann near-linear ABOD für dieses Problem als der am besten geeignete Algorithmus betrachtet werde.

	Lauf 1	Lauf 2	Lauf 3	Lauf 4	Lauf 5
gefundene Quader	13	27	27	32	26
Quader mit < 10 Daten	7	21	19	24	18
Anzahl Daten in Quadern mit < 10 Daten	26	34	15	31	19
Anzahl der Ausreißer	3	0	0	3	2
Datensatz-Id der Ausreißer	2, 3, 6	–	–	2, 3, 4	148, 5
Ähnlichkeit mit Hilbert OD	0.19	0.21	0.13	0.23	0.21
Ähnlichkeit mit ABOD	0.23	0.18	0.13	0.19	0.26
Ähnlichkeit mit near-linear ABOD	0.15	0.15	0.13	0.19	0.21

Tabelle 8.7: Ergebnisse Evolutionary OD, Datensatz *Arrhythmia*.

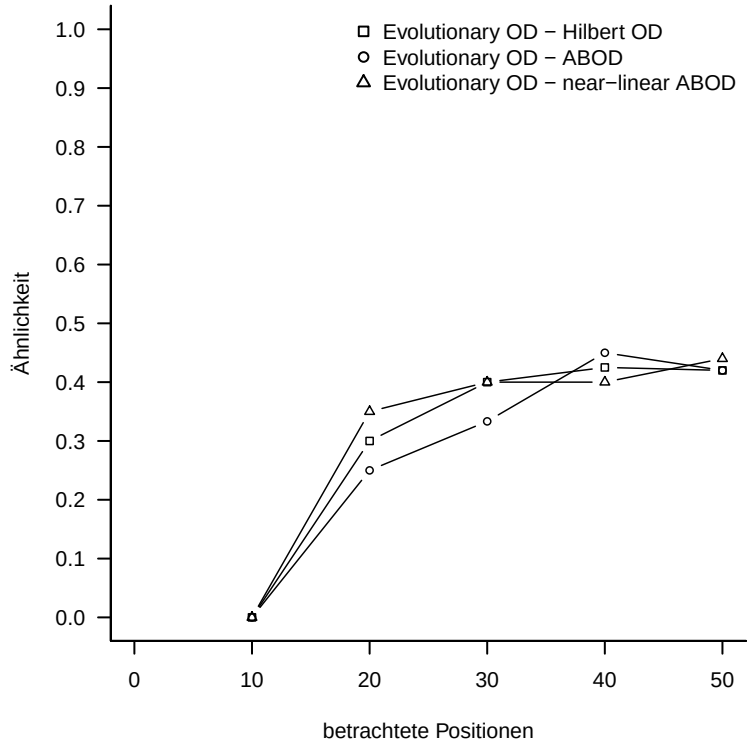

Abbildung 8.7: Ähnlichkeit zw. Evolutionary OD und den anderen Algorithmen, Datensatz *Arrhythmia*, Anzahl der Daten $N = 255$, Dimensionen $n = 220$.

8.2 Datensatz Bag-of-words von Stack Exchange

Ein Bag-of-words-Modell bietet eine Möglichkeit, Dokumente für weitere Analysen als Vektoren darzustellen. Gegeben sei ein Korpus aus Dokumenten (Texte). Die Dokumente bestehen aus Termen (Wörtern). Insgesamt gibt es n verschiedene Terme in allen Dokumenten des Korpus. Um den bag of words zu erhalten, zählen wir die Terme in den einzelnen Dokumenten und speichern diese absoluten Häufigkeiten als n-dimensionalen Vektor ab. Diese Vektoren bilden den Datenraum.

Beispiel 8.2. Gegeben seien zwei Dokumente:

$$d_1 : \text{„Der Ball liegt in der Wiese.“}$$
$$d_2 : \text{„Die Wiese ist feucht.“}$$

Die Menge aller Terme ist dann

$$\{\text{Ball}, \text{der}, \text{die}, \text{feucht}, \text{in}, \text{ist}, \text{liegt}, \text{Wiese}\}. \tag{8.2}$$

Evolutionary OD		Hilbert OD		near-linear ABOD	
Parameter	Wert	Parameter	Wert	Parameter	Wert
m	100	$\mathfrak{n}$	50	t	500
ϕ	5	k	10	w_1	100
κ	2	h	4	w_2	100
p_1	0.1				
p_2	0.1				

Tabelle 8.8: Fix gesetzte Parameter, Datensatz *bag-of-words*.

Das Bag-of-words-Modell bildet die beiden Dokumente auf die Vektoren

$$(1, 2, 0, 0, 1, 0, 1, 1) \text{ und}$$
$$(0, 0, 1, 1, 0, 1, 0, 1)$$

ab.

Unsere Dokumente stammen von Stack Exchange [https://stackexchange.com], einer Internetseite bei der Benutzer Fragen stellen und Fragen anderer Benutzer beantworten können. Fragen und Antworten sind in eine Vielzahl von Themenbereichen gegliedert. Für unsere Analyse verwenden wir Fragen aus den Bereichen Data Science [https://datascience.stackexchange.com], Law [https://law.stackexchange.com] und German Language [https://german.stackexchange.com]. Die Daten können vom Internet Archive unter https://archive.org/details/stackexchange heruntergeladen werden. Wir verwenden die Daten mit Stand 14.3.2017.

Unser Korpus besteht aus 100 Fragen (ohne Antworten). 94 Fragen stammen aus dem Bereich Data Science, 3 aus dem Bereich Law und 3 aus dem Bereich German Language. Die Data-Science- und Law-Fragen sind in Englisch gestellt. Bei den Fragen aus German Language wird teilweise Deutsch, teilweise Englisch verwendet. Wir haben explizit Fragen auf Deutsch in den Korpus aufgenommen.

Die Fragen wurden anschließend transformiert:

- Alle Buchstaben wurden in Kleinbuchstaben umgewandelt.
- Ziffern, Satzzeichen und HTML-Tags wurden entfernt.
- Als Trennzeichen von einzelnen Termen wurde das Leerzeichen definiert.
- Die Datensätze wurden mit Formel 8.1, Abschnitt 8.1, normalisiert. Die Attribute der Vektoren geben daher die relativen Häufigkeiten an.

Es wurden keine Stoppwörter (Wörter ohne Bedeutung, z. B. Artikel) entfernt und keine Wörter in die Stammform umgewandelt (z. B. „liegen“ statt „liegt“).

Der Korpus besitzt nach der Transformation 2392 Terme, der Datenraum somit 2392 Dimensionen. Die Fragen haben im Durchschnitt 109 Terme, die meisten Attribute der einzelnen Datensätze haben daher den Wert 0.

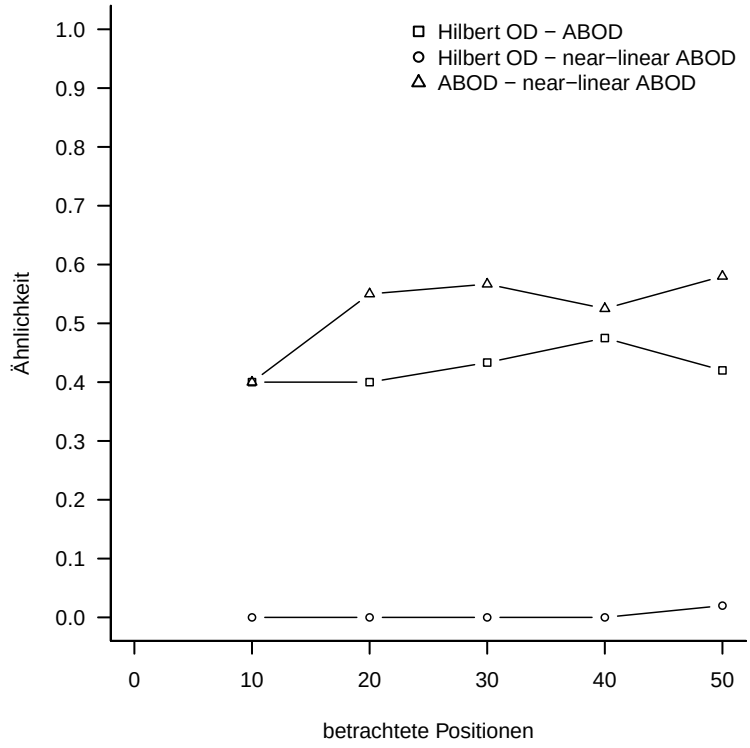

Abbildung 8.8: Ähnlichkeit der Algorithmen, Datensatz *bag-of-words*, Anzahl der Daten $N = 100$, Dimensionen $n = 2392$.

Die verwendeten Parameter stehen in Tabelle 8.8. Die Recallwerte und die Ähnlichkeiten der Algorithmen sieht man in den Abbildungen 8.8, 8.9 und 8.11. Die Algorithmen Hilbert OD und near-linear ABOD liefern relativ ähnliche Ergebnisse wie der Algorithmus ABOD. Daten, die sowohl von Hilbert OD als auch von ABOD als Ausreißer klassifiziert werden, werden von near-linear ABOD als normale Daten klassifiziert und umgekehrt. Daher ergeben sich die Ähnlichkeitswerte von ca. 0.5 und 0.

Betrachten wir Abbildung 8.9 sehen wir dass keiner der drei Algorithmen in der Lage ist, die Ausreißer zu finden. In Tabelle 8.9 sind die Positionen, an die die sechs Ausreißer gesetzt wurden, eingetragen – kein Ausreißer wurde an eine niedrige Position gereiht. Bei den Ausreißern aus der Kategorie *Law* mag das noch einleuchten: Die Dokumente enthalten nur wenige Fachbegriffe aus den jeweiligen Bereichen. Bei den in einer anderen Sprache geschriebenen Ausreißern der Kategorie *German Language* ist dieser Umstand erstaunlich.

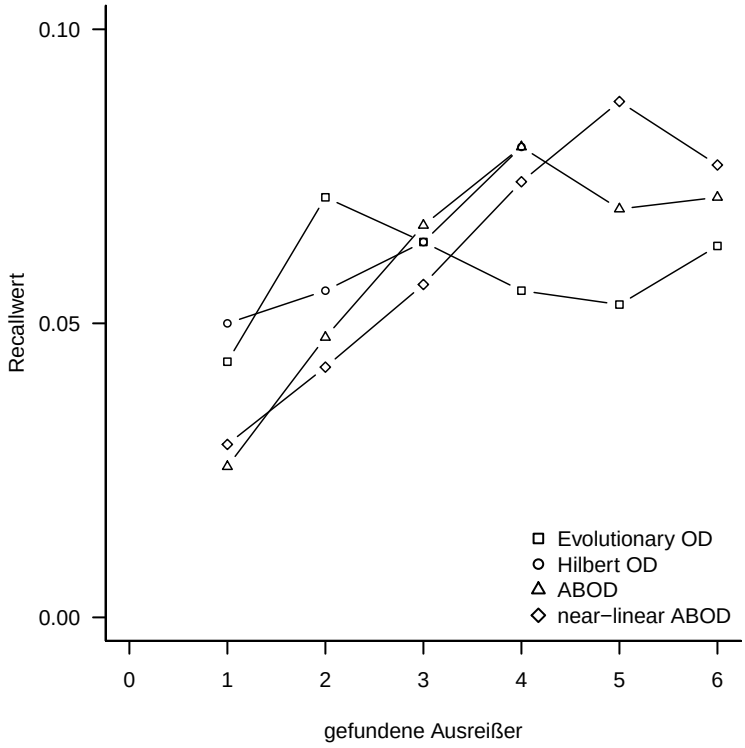

Abbildung 8.9: Recallwerte der Algorithmen, Datensatz *bag-of-words*, Anzahl der Daten $N = 100$, Dimensionen $n = 2392$.

Ausreißer	Evolutionary OD	Hilbert OD	ABOD	near-linear ABOD
German Language 1	72	50	50	53
German Language 2	94	–	39	34
German Language 3	95	–	45	47
Law 1	47	47	84	54
Law 2	28	36	72	57
Law 3	23	20	42	78

Tabelle 8.9: Position der Ausreißer, Datensatz *bag-of-words*

Der Grund für diese Reihungen scheint darin zu liegen, dass die meisten Attribute 0 oder 1 sind. D. h. die meisten Terme kommen nur in genau einem Dokument vor. Oder anders ausgedrückt: Alle Dokumente sind einzigartig und können damit als Ausreißer gewertet werden (umgekehrt betrachtet: keine Dokumente). Das gilt sowohl für die in deutsch als auch für die in englisch geschriebenen Dokumente, daher können die Algorithmen die beiden Klassen nicht unterscheiden.

Je länger ein Dokument ist, desto mehr Attribute haben den Wert 1, je kürzer desto mehr den Wert 0. Das könnte die Begründung für den aus Abbildung 8.10 ersichtlichen

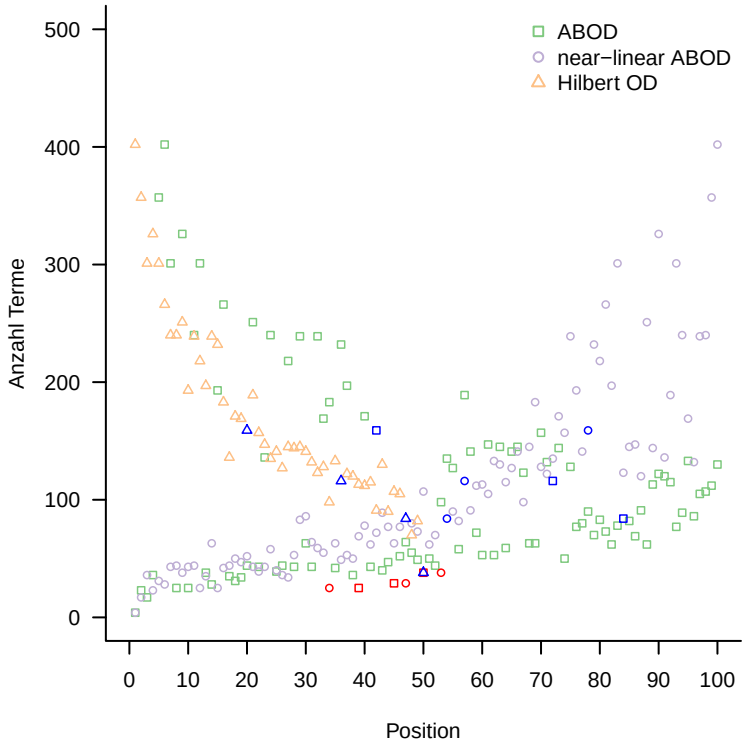

Abbildung 8.10: Die blauen Punkte gehören zur Klasse *Law*, die roten Punkte zur Klasse *German Language*. Die restlichen Punkte stellen normalen Daten (Klasse *Data Science*) dar. Jedes Datum ist für jeden Algorithmus (insgesamt dreimal) eingezeichnet.

Zusammenhang sein. In Abbildung 8.10 ist die Anzahl der Terme über der Position der Daten aufgetragen. Man sieht deutlich dass Hilbert OD jene Dokumente mit den meisten Termen, near-linear ABOD jene Dokumente mit den wenigsten Termen und ABOD jene Dokumente mit den meisten und wenigsten Termen an eine niedrige Position reiht bzw. als mögliche Ausreißer klassifizieren.

Um auch mit dem Algorithmus Evolutionary OD eine Reihung der Daten zu erhalten, haben wir den Algorithmus wie im Abschnitt 8.1.2 hundertmal gestartet und für jedes Datum gezählt, wie oft das Datum in einem schwach besetzten Quader vorkommt. Schwach besetze Quader sind dabei Quader mit weniger als 5 Daten.

Die Ergebnisse finden sich bei denen der anderen Algorithmen (Tabelle 8.9 und Abbildung 8.9) und haben die gleiche, geringe, Qualität wie die drei anderen Algorithmen. Im Gegensatz zu den anderen Algorithmen gibt es bei Evolutionary OD keinen offensichtlichen Zusammenhang zwischen der Länge eines Dokuments und dessen Klassifizierung als Ausreißer.

Abbildung 8.11 stellt die Ähnlichkeit der Ergebnisse zwischen Evolutionary OD und den anderen Algorithmen dar. Wir sehen ein ähnliches Phänomen wie in Abbildung 8.8: Die Ähnlichkeit zu Hilbert OD ist gut, ungefähr 0.5, die zu den beiden ABOD-Varianten

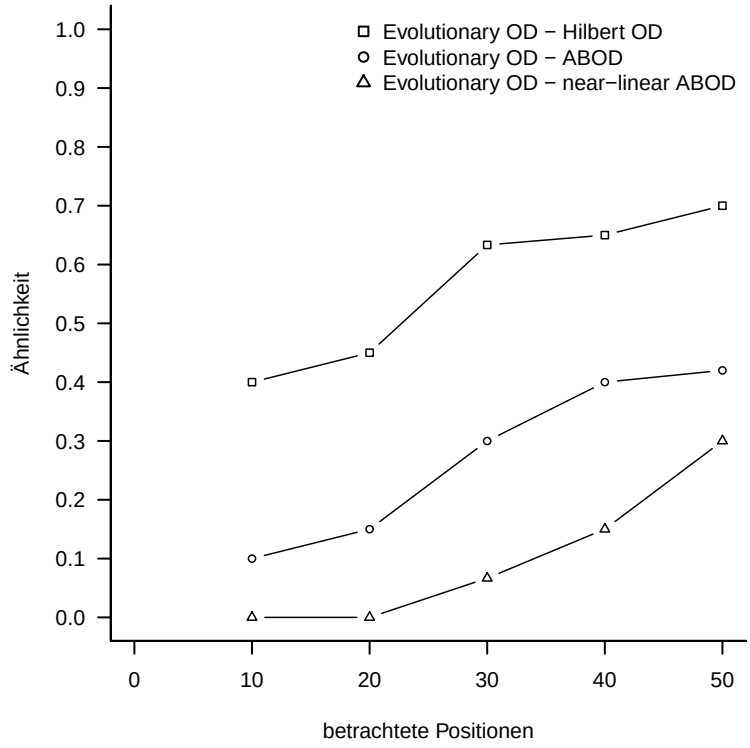

Abbildung 8.11: Ähnlichkeit zw. Evolutionary OD und den anderen Algorithmen, Datensatz *bag-of-words*.

schlechter.

Betrachten wir die Vereinigung der Mengen der Ausreißer mit den jeweils 10 höchsten Bewertungen der vier Algorithmen, so enthält diese Menge 28 verschiedene Dokumente (bei einem theoretischen Maximum von 40). Jeder der vier Algorithmen klassifiziert andere Dokumente als Ausreißer. Das unterstützt die weiter oben formulierte These: Die Attribute der sechs Ausreißer unterscheiden sich in ihrer Struktur nicht von den normalen Daten. Die Algorithmen können diese sechs Daten daher nicht als Ausreißer erkennen.

Die Laufzeit von ABDO betrug 46 s, near-linear ABOD ist mit 7 s schneller. Die Laufzeit von Evolutionary OD beträgt durchschnittlich 105 s. Mit Abstand am längsten benötigte Hilbert OD um ein Ergebnis zu ermitteln: 133 min. Dieser extrem hohe Wert ist allerdings der schlechten Implementierung geschuldet, nicht dem Algorithmus selbst.

8.3 Grammatical-Facial-Expressions-Datensatz

Der Datensatz *Grammatical Facial Expressions* (GFE) stammt ebenfalls vom *UCI Machine Learning Repository* ([Lic13], [Fre14], [https://archive.ics.uci.edu/ml/datasets/Grammatical+Facial+Expressions]). Mithilfe von Videoaufnahmen wurde die Mimik ei-

Evolutionary OD		Hilbert OD	
Parameter	Wert	Parameter	Wert
m	100	$\mathfrak{n}$	100
ϕ	5	k	20
κ	3	h	4
p_1	0.1		
p_2	0.1		
FastABOD		near-linear ABOD	
Parameter	Wert	Parameter	Wert
k	20	t	500
		w_1	100
		w_2	100

Tabelle 8.10: Fix gesetzte Parameter, Datensatz *GFE*.

ner Versuchsperson aufgenommen. Jedes Datum besteht aus den x-, y- und z-Koordinaten von 100 Punkten im Gesicht. Die Mienen sind Teil der Brasilianischen Gebärdensprache. Bestimmte Gesichtsausdrücke haben dabei eine bestimmte grammatikalische Bedeutung. Jedes Datum gehört zu einer grammatikalischen Bedeutung. Man unterscheidet weiters zwischen Daten, bei denen die Person die Miene mit der Bedeutung aufsetzt, oder die Miene nicht aufsetzt.

Unsere normalen Daten gehören zu den Klassen *Conditional Clause* (1359 Daten) und *Doubt Question* (821 Daten). In beiden Fällen nehmen wir jene Daten, in denen die Person die Mienen mit den Bedeutungen Conditional Clause und Doubt Question nicht aufsetzt.

Unsere Ausreißer stammen aus den Klassen *Affirmative* und *Emphasis*. Aus diesen beiden Klassen nehmen wir jeweils 10 Daten mit und 10 Daten ohne entsprechende Mimik. Wir bezeichnen die Ausreißer mit $\mathit{Affirmative}_{+1}, \mathit{Affirmative}_{+2}, \ldots, \mathit{Affirmative}_{+10}$, $\mathit{Affirmative}_{-1}, \ldots, \mathit{Affirmative}_{-10}$ und analog mit $\mathit{Emphasis}_{+1}, \ldots, \mathit{Emphasis}_{-1}, \ldots$ Insgesamt arbeiten wir mit 2180 normale Daten und 40 Ausreißer. Aufgrund der großen Anzahl an Daten verwenden wir statt dem Algorithmus ABOD den Algorithmus FastABOD. Wir haben FastABOD dem Algorithmus LB-ABOD vorgezogen, da ersterer bessere Resultate liefert.

Jedes Datum besteht aus 100 Punkten im Gesicht zu je 3 Koordinaten. Unser Datenraum hat somit 300 Dimensionen.

Die verwendeten Parameter stehen in Tabelle 8.10, die Ergebnisse in Tabelle 8.12, die Recallwerte sind in Abbildung 8.12 dargestellt. Wir betrachten nur die 100 Daten mit der höchsten Bewertung, auch wenn die Algorithmen FastABOD und near-linear ABOD für alle Daten eine Bewertung berechnen.

Hilbert OD liefert relativ gute Ergebnisse, die Ergebnisse von near-linear ABOD sind ähnlich, die Ausreißer erhalten allerdings eine schlechtere Bewertung. Die Recallwerte von FastABOD sind noch schlechter, FastABOD findet darüber hinaus andere Ausreißer als Hilbert OD und near-linear ABOD. Abbildung 8.13 bestätigt das.

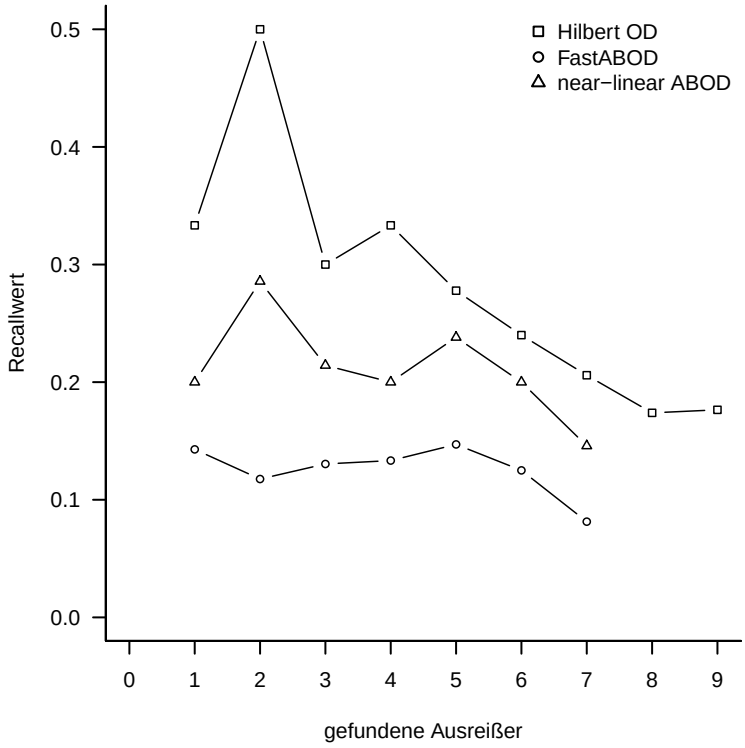

Abbildung 8.12: Recallwerte der Algorithmen, Datensatz *GFE*, Anzahl der Daten $N = 2220$, Dimensionen $n = 300$.

Evolutionary OD liefert auch für diesen Datensatz die schlechtesten Resultate. Wegen der langen Laufzeit wurde der Algorithmus nur zehnmal (anstatt hundertmal wie bei den beiden anderen Datensätzen) gestartet. Die Ausreißer $\mathit{Affirmative}_{+2}$, $\mathit{Affirmative}_{+5}$ und $\mathit{Affirmative}_{-1}$ befanden sich in drei Durchläufen in einem schwach besetzten Quader (ebenso wie weitere 32 Daten, daher die gemeinsame Reihung an Position 33–68 in Tabelle 8.12). Für die Ausreißer $\mathit{Affirmative}_{+8}$ und $\mathit{Affirmative}_{+9}$ traf dies in zwei Durchläufen zu. Aufgrund dieser groben Reihung wurden für Evolutionary OD keine Recall- und Ähnlichkeitswerte berechnet.

Es wurde auch mit dem Algorithmus LB-ABOD nach Ausreißern gesucht. Unter den Daten mit den 100 kleinsten ABOF-Werten befand sich kein einziger Ausreißer. Die Abschätzung LB-ABOF ist für eine Vielzahl normaler Daten kleiner als bei den Ausreißern, daher werden keine Ausreißer in die Ergebnisliste aufgenommen. Berechnet man den ABOF der 40 Ausreißer explizit, und vergleicht diese Werte mit den ABOF-Werten der Ergebnisse von LB-ABOD, stellt man fest, dass bei 25 Ausreißern der ABOF kleiner ist als der kleinste ABOF der LB-ABOD-Ergebnisse. Unter diesen 25 Ausreißern befinden sich auch alle Ausreißer die Hilbert OD und near-linear ABOD fanden. Man muss jedenfalls die Frage stellen, wann der LB-ABOF eine brauchbare Abschätzung ist.

Wie bei den beiden anderen Datensätzen gilt auch hier: Ohne Expertenwissen zu Rate

zu ziehen liefert kein Algorithmus ein Ergebnis, welches man unmittelbar verwenden kann, sondern lediglich Vorschläge, die man noch näher betrachten muss. Alternativ kann man versuchen, die Daten vor der Ausreißererkennung passend zu selektieren.

Abschließend geben wir noch in Tabelle 8.11 die Laufzeiten der Algorithmen an. Die extrem lange Laufzeit von Hilbert OD liegt auch hier an der schlechten Implementierung.

FastABOD	4.6 s
Evolutionary OD (durchschnittlich)	203 s
near-linear ABOD	248 s
Hilbert OD	853 min

Tabelle 8.11: Laufzeit der Algorithmen, Datensatz *GFE*

Ausreißer	Evolutionary OD	Hilbert OD	FastABOD	near-linear ABOD
$Affirmative_{+2}$	33–68	–	–	–
$Affirmative_{+5}$	33–68	–	–	–
$Affirmative_{+7}$	–	–	34	–
$Affirmative_{+8}$	69–128	–	–	–
$Affirmative_{+9}$	69–128	–	48	–
$Affirmative_{-1}$	33–68	4	–	7
$Affirmative_{-2}$	–	25	–	14
$Affirmative_{-3}$	–	51	–	21
$Affirmative_{-4}$	–	–	86	–
$Affirmative_{-6}$	–	–	17	–
$Affirmative_{-8}$	–	–	7	–
$Emphasis_{+9}$	–	–	23	–
$Emphasis_{-1}$	–	3	–	5
$Emphasis_{-2}$	–	18	–	48
$Emphasis_{-3}$	–	10	–	20
$Emphasis_{-4}$	–	12	–	30
$Emphasis_{-5}$	–	34	–	–
$Emphasis_{-8}$	–	46	–	–
$Emphasis_{-10}$	–	–	30	–
Anzahl	5	9	7	7

Tabelle 8.12: Position der Ausreißer unter den 100 ersten Daten, Datensatz *GFE*.

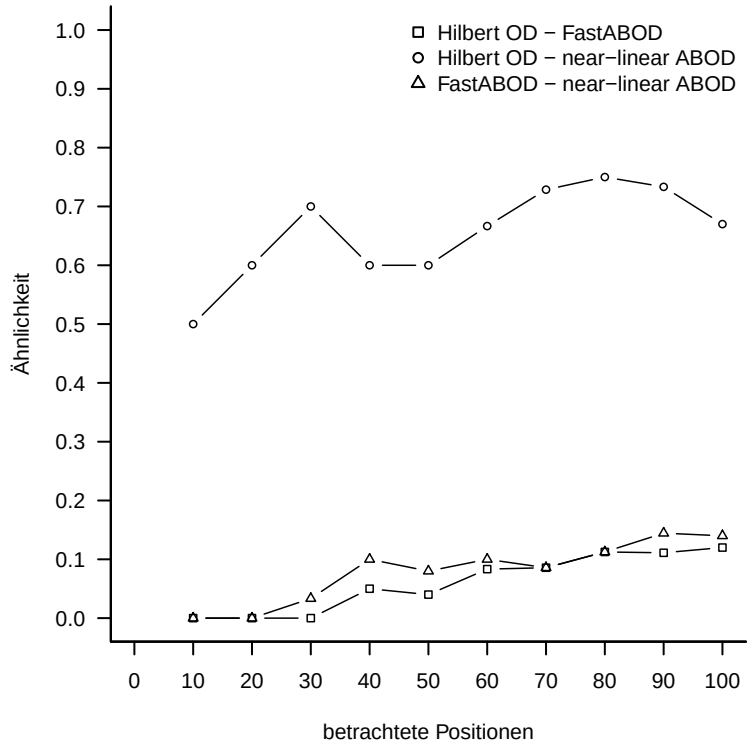

Abbildung 8.13: Ähnlichkeit der Algorithmen, Datensatz *GFE*, Anzahl der Daten $N = 2220$, Dimensionen $n = 300$.

9 Resümee

Ziel dieser Arbeit ist es, die Ergebnisse von Algorithmen zur Ausreißererkennung, die speziell für höherdimensionale Räume entwickelt wurden, zu vergleichen. Im vorangehenden Kapitel 8 suchten wir in drei Datensätzen nach Ausreißern und präsentierten die teilweise unterschiedlichen Resultate.

Einige Erkenntnisse daraus sind: Die Algorithmen Hilbert OD, ABOD und near-linear ABOD liefern vergleichbare Ergebnisse. D. h. unter den Daten mit den höchsten Bewertungen befinden sich ähnlich viele Ausreißer. Near-linear ABOD kann als Monte-Carlo-Algorithmus keine exakte Lösung garantieren, liefert aber in der Praxis gleich gute Ergebnisse wie ABOD. Dabei ist er deutlich schneller.

Die Laufzeit wächst bei diesen drei Algorithmen höchstens linear mit der Dimension und bei Hilbert OD und near-linear ABOD logarithmisch-linear mit der Anzahl der Daten. Beschränken wir unsere Überlegungen rein auf die Laufzeit, so sind Hilbert OD und near-linear ABOD sowohl für hohe Dimensionen als auch für große Datenmengen geeignet.

Absolut bewertet sind die Ergebnisse jedoch schlecht. Die Recallwerte bewegen sich, von Ausnahmen abgesehen, im Bereich von 0.2–0.3, bei dem Datensatz Bag-of-words lediglich im Bereich von 0.05. Die Daten, die die Algorithmen als Ausreißer vorschlagen, müssen in allen Fällen durch einen Experten überprüft werden. Weiters muss man sich bewusst sein, dass eine Reihe von Ausreißern unentdeckt bleibt. Genauso wie die Anzahl an falsch positiv, ist auch die Anzahl an falsch negativ klassifizierten Daten hoch.

Die oberflächliche Auswahl der Daten und Datensätze relativiert allerdings die schlechten Ergebnisse. Weder die normalen Daten noch die Ausreißer wurden mit Expertenhilfe ausgewählt. Es ist daher nicht garantiert, dass die Daten die wir als Ausreißer hinzufügten, sich von den normalen Daten nennenswert unterscheiden. Für den Datensatz Bag-of-words gilt das bereits in Abschnitt 8.2 gesagte: Alle Daten sind einzigartig, es sind keine ausgezeichneten Ausreißer vorhanden. Die Struktur des Datensatzes lässt keine brauchbaren Resultate zu.

Der Algorithmus Evolutionary OD nimmt eine Sonderstellung ein. Bei keinem Datensatz lieferte dieser Algorithmus ein brauchbares Ergebnis. Eine mögliche Erklärung dafür ist: Die Daten in einem Quader mit unterdurchschnittlich vielen Daten sind oft keine Ausreißer, eventuell liegen sie nur am Rand der normalen Daten. Da der Algorithmus ein evolutionärer Algorithmus ist, hängt es vom Zufall ab, ob speziell die Quader mit Ausreißern gefunden werden.

[Agg01b] untersuchen in ihrer Arbeit ebenfalls den Datensatz Arrhythmia und kommen zu besseren Ergebnissen. Sie verwenden alle 14 Klassen (insgesamt 452 Daten) und definieren die Daten der Klassen mit nur wenigen Daten als Ausreißer. Möglicherweise befinden sich in diesem Fall mehr Ausreißer in dünner besetzten Quadern und

können leichter gefunden werden. [Agg01b] schlüsseln leider nicht auf, ob die Daten bestimmter Ausreißer-Klassen öfter gefunden wurden. Für ihre Arbeit verwenden sie noch weitere Datensätze, dabei betrachten sie lediglich, ob die gefundenen Quader unterdurchschnittlich viele Daten enthalten. Ob die Daten dieser Quader besondere Eigenschaften aufweisen, wird nicht analysiert.

Hilbert OD arbeitet mit einer Minkowski-Metrik um die Daten zu ordnen, ABOD mit Winkeln. Hilbert OD unterliegt dadurch zwar dem Fluch der Dimensionalität, die Qualität der Resultate ist aber trotzdem gleich, ABOD ist Hilbert OD nicht überlegen. Damit stellen sich folgende Fragen: Inwieweit ist ABOD besser geeignet dem Fluch der Dimensionalität zu entkommen? Die Aussage einer Minkowski-Metrik wird mit wachsender Dimension geringer, aber ab wann ist die Aussage so gering, dass sie wertlos ist? Liefern andere Algorithmen die mit einer Minkowski-Metrik arbeiten und deshalb für höherdimensionale Räume nicht verwendet werden, vielleicht gleichfalls Resultate von ähnlicher Qualität?

Abschließend kann man sagen, dass die Algorithmen Hilbert OD und die verschiedenen ABOD-Varianten gleichwertig sind. Ob für ein konkretes Problem der eine oder andere Algorithmus besser geeignet ist, muss man für den jeweiligen Einzelfall gesondert entscheiden. Dabei können eventuell auch Fragen der Implementierung (z. B. Speicherverbrauch, Parallelisierung) eine Rolle spielen. Dem Algorithmus Evolutionary OD sind beide Algorithmen eindeutig überlegen.

A Formelzeichen, Symbole und Abkürzungen

Formelzeichen	Bedeutung
$\lvert\ \rvert$	Betrag einer reellen Zahl oder Mächtigkeit einer Menge
$x \otimes y$	dyadisches Produkt
$\lVert\ \rVert_F$	Frobeniusnorm
$\langle\ ,\ \rangle$	kanonisches Skalarprodukt
$\dot{\vee}$	Kontravalenz, exklusives Oder
$\rightarrow_{f.\,s.}$	Konvergenz fast sicher
$\rightarrow_p$	Konvergenz in Wahrscheinlichkeit
$\leq_h$	Ordnung, durch Hilbert-Wert induziert
$\lVert\ \rVert_p$	p-Norm
ABOD	*Angle-Based Outlier Detection*
ABOF(q)	*Angle-Based Outlier Factor* von q
AMS	*AMS Sketch*
approxABOF(q)	*approximate Angle-Based Outlier Factor* von q
C	reelle Konstante
$\mathcal{C}$	Quader oder Würfel
$\mathcal{C}^\dagger, \mathcal{C}^{\dagger\dagger}$	Elternquader
$\mathcal{C}^*, \mathcal{C}^{**}$	Kindquader
D	Menge der Daten, Datenraum
$D^k_{\mathfrak{n}}$	Menge der $D^k_{\mathfrak{n}}$-Ausreißer
$\mathrm{d}\,(x, y)$	Metrik
$\mathrm{d}_p(x, y)$	Metrik, abgeleitet von einer p-Norm
$d^k(q)$	Abstand von q zum k-nächsten Nachbarn von q
DB()	Menge der Distance-Based-Ausreißer
$\dim(V)$	Dimension des Vektorraums V
d_{min}	minimaler Abstand zwischen einem Abfragedatum und allen anderen Daten
d_{min}	maximaler Abstand zwischen einem Abfragedatum und allen anderen Daten
E^n	n-dimensionaler Einheiteswürfel
$\mathbb{E}$	Erwartungswert
Evolutionary OD	*Evolutionary Outlier Detection in Subspaces*

Formelzeichen	Bedeutung
F	Wahrscheinlichkeitsverteilung
FastABOD	*fast Angle-Based Outlier Detection*
$\mathrm{fdm}\,(x, y)$	*fractional distance metric*
f	Kehrwert der Anzahl der Streifen $= 1/\phi$
GFE	Datensatz *Grammatical Facial Expressions*
Hilbert OD	*Hilbert Outlier Detection*
$h(q)$	Hilbert-Wert von q
k	Anzahl der betrachteten Nachbarn bei $d^k(q)$, $D^k_{\mathfrak{n}}$, $N_k(q)$, w^k, $W^k_{\mathfrak{n}}$
κ	Dimension eines Quaders, Ordnung einer Hilbert-Kurve
LB-ABOF(q)	*lower bound Angle-Based Outlier Factor* von q
LB-ABOD	*lower bound Angle-Based Outlier Detection*
linearABOF(q)	*near-linear Angle-Based Outlier Factor* von q
max	Maximum
min	Minimum
N	Anzahl der Daten, $\lvert D\rvert$
$N_k(q)$	Menge der k nächsten Nachbarn von q
$\mathcal{N}$	Normalverteilung
n	Dimension des Datenraums D
$\mathfrak{n}$	Anzahl an zu suchenden Ausreißern, Anzahl der betrachteten Daten bei $D^k_{\mathfrak{n}}$ und $W^k_{\mathfrak{n}}$
$\mathcal{O}$	Landau-Symbol
$\mathbb{P}$	Wahrscheinlichkeitsmaß
p	Parameter der p-Norm
Q	Zufallsvektor dessen Realisation ein Abfragedatum ist
q	Abfragedatum
$\mathbb{R}$	Menge der reellen Zahlen
$\mathbb{R}_+$	Menge der positiven reellen Zahlen > 0
$\mathbb{R}^0_+$	Menge der nichtnegativen reellen Zahlen ≥ 0
r_i	Zufallsvektor, $1 \leq i \leq t$
S	Schätzer
$\mathfrak{S}$	σ-Algebra eines Wahrscheinlichkeitsraums
t	Anzahl der Zufallsvektoren
$W^k_{\mathfrak{n}}$	Menge der $W^k_{\mathfrak{n}}$-Ausreißer
w^k	Gewicht eines Datums in Bezug auf die nächsten k Nachbarn
X, Y	Zufallsgrößen, Zufallsvariablen, Zufallsvektoren
x, y	Daten, Vektoren
Θ_{xqy}	Schätzer für φ_{xqy}
φ_{xqy}	Winkel zwischen den Vektoren $\vec{qx}$ und $\vec{qy}$
ϕ	Anzahl der Streifen
Ω	Ereignisraum eines Wahrscheinlichkeitsraums
ω	Element des Ereignisraums Ω

Literaturverzeichnis

[Agg01a] Aggarwal, Charu C., Alexander Hinneburg und Daniel A. Keim (2001): *On the Surprising Behavior of Distance Metrics in High Dimensional Space*. In: Van den Bussche, Jan und Victor Vianu (Hrsg.): *Proceedings of the 8th International Conference on Database Theory (ICDT)*, S. 420–434. London. Berlin u. a. : Springer.

[Agg01b] Aggarwal, Charu C. und Philip S. Yu (2001): *Outlier Detection for High Dimensional Data*. In: Sellis, Timos und Sharad Mehrotra (Hrsg.): *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, S. 37–46. Santa Barbara. New York: ACM.

[Agg13] Aggarwal, Charu C. (2013): *Outlier Analysis*. New York u. a. : Springer.

[Alo96] Alon, Nogai, Matias, Yossi und Szegedy, Mario: *The space complexity of approximating the frequency moments*. In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of computing*, S. 20–29. Philadelphia. New York: ACM.

[Ang05] Angiulli Fabrizio und Clara Pizzuti (2005): *Outlier Mining in Large High-Dimensional Data Sets*. In: *IEEE Transactions on Knowledge and Data Engineering* 17, S. 203–215.

[Bar94] Barnett, Vic und Toby Lewis (1994): *Outliers in Statistical Data*. 3. Aufl. Chichester: John Wiley & Sons.

[Bey99] Beyer, Kevin, Jonathan Goldstein, Raghu Ramakrishnan und Uri Shaft (1999): *When Is „Nearest Neighbor“ Meaningful?*. In: Beeri, Catriel und Peter Buneman (Hrsg.): *Proceedings of the 7th International Conference on Database Theory*, S. 217–235. Jerusalem. Berlin u. a. : Springer.

[Bra10] Braverman, Kai-Min Chung, Zhenming Liu, Michael Mitzenmacher und Rafail Ostrovsky (2010): *AMS Without 4-Wise Independence on Product Domains*. In: Marion, Jean-Yves und Thomas Schwentick (Hrsg.): *Proceedings of the 27th Annual Symposium on the Theoretical Aspects of Computer Science*, S. 119–130. Nancy. o. O.: Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik.

[Cha98] Chan Timothy M. (1998): *Approximate Nearest Neighbor Queries Revisited*. In: *Discrete & Computational Geometry* 20, S. 359–373.

[Dub09] Dubhashi, Devdatt P. und Alessandro Panconesi (2009): *Concentration of Measure for the Analysis of Randomised Algorithms*. Cambridge u. a.: Cambridge University Press.

[Fre14] Freitas, Fernando de Almeida, Sarajane M. Peres, Clodoaldo A. M. Lima und Felipe V. Barbosa (2014): *Grammatical Facial Expressions Recognition with Machine Learning*. In: *Proceedings of the 27th Florida Artificial Intelligence Research Society Conference (FLAIRS)*, S. 180–185. Palo Alto: The AAAI Press.

[Gru69] Grubbs, Frank E. (1969): *Procedures for Detecting Outlying Observations in Samples*. In: *Technometrics* 11, S. 1–21.

[Goe95] Goemans, Michel X. und David P. Williamson (1995): *Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming*. In: *Journal of the ACM* 42, S. 1115–1145. New York: ACM.

[Han12] Han, Jiawei, Micheline Kamber und Jian Pei (2012): *Data Mining – Concepts and Techniques*. 3. Aufl. Waltham: Morgan Kaufmann.

[Haw80] Hawkins, Douglas M. (1980): *Identifications of Outliers*. London: Chapman und Hall.

[Hin00] Hinneburg, Alexander, Charu C. Aggarwal und Daniel A. Keim (2000): *What is the nearest neighbor in high dimensional spaces?*. In: Abbadi, Amr El, Michael L. Brodie, Sharma Chakravarthy, Umeshwar Dayal, Nabil Kamel, Gunter Schlageter und Kyu-Young Whang (Hrsg.): *Proceedings of the 26th International Conference on Very Large Data Bases*, S. 506–515. Cairo. San Francisco: Morgan Kaufmann.

[Hod04] Hodge, Victoria J. und Jim Austin (2004): *A Survey of Outlier Detection Methodologies*. In: *Artificial Intelligence Review* 22, S. 85–126. Kluwer.

[Ind08] Indyk, Piotr und Andrew McGregor: *Declaring Independence via the Sketching of Sketches*. In: *Proceedings of the Nineteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, S. 737–745. San Francisco. Philadelphia: Society for Industrial and Applied Mathematics.

[Kno98] Knorr, Edwin M. und Raymond T. Ng: *Algorithms for Mining Distance-Based Outliers in Large Datasets*. In: Gupta, Ashish, Oded Shmueli und Jennifer Widom (Hrsg.): *Proceedings of 24th International Conference on Very Large Data Bases*, S. 392–403. New York. San Francisco: Morgan Kaufmann.

[Kri08] Kriegel, Hans-Peter, Matthias Schubert und Arthur Zimek: *Angle-Based Outlier Detection in High-dimensional Data*. In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, S. 444–452. Las Vegas. New York: ACM.

[Lic13] Lichman, M. (2013): *UCI Machine Learning Repository*. Irvine, CA: University of California, School of Information and Computer Science. [http://archive.ics.uci.edu/ml]

[Pha12] Pham, Ninh und Rasmus Pagh (2012): *A Near-linear Time-Approximation Algorithm for Angle-based Outlier Detection in High-dimensional Data*. In: *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, S. 877–885. Peking. New York: ACM.

[Ram00] Ramaswamy, Sridhar, Rajeev Rastogi und Kyuseok Shim (2000): *Efficient Algorithms for Mining Outliers from Large Data Sets*. In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, S. 427–438. Dallas. New York: ACM.

[RCT12] R Core Team (2012): *R: A language and environment for statistical computing* [http://www.R-project.org]. Vienna: R Foundation for Statistical Computing.

[Sag94] Sagan, Hans (1994): *Space-Filling Curves*. New York: Springer.

[Sch11] Schmidt, Volker (2011): *Elementare Wahrscheinlichkeitsrechnung und Statistik*, Wintersemester 2010/11, Institut für Stochastik, Universität Ulm, unveröffentlicht.

[Sin12] Singh, Karanjit und Shuchita Upadhyaya (2012): *Outlier Detection: Applications and Techniques*. In: *International Journal of Computer Science Issues* 9, S. 308–323.