INAUGURAL-DISSERTATION

zur
Erlangung der Doktorwürde
der
Naturwissenschaftlich-Mathematischen Gesamtfakultät
der
Ruprecht-Karls-Universität Heidelberg

vorgelegt von
M.Sc. Maximilian Hoecker
aus
Ludwigshafen am Rhein

Tag der mündlichen Prüfung: .....

Clustering von großen hochdimensionalen und unsicheren Datensätzen in der Astronomie

Betreuer:
Prof. Dr. Vincent Heuveline
Dr. habil. Marcel Kunze

Zusammenfassung

Ein ständiges Wachstum der Datenmengen ist in vielen IT-affinen Bereichen gegeben. Wissenschaftliche und insbesondere astronomische Datensätze weisen komplexe Eigenschaften wie Unsicherheiten, eine hohen Anzahl an Dimensionen sowie die enorme Anzahl an Dateninstanzen auf. Beispielsweise besitzen astronomische Datensätze mehrere Millionen Dateninstanzen mit jeweils mehreren tausend Dimensionen, die sich durch die Anzahl unabhängiger Eigenschaften bzw. Komponenten widerspiegeln. Diese Größenordnungen bzgl. der Dimensionen und Datenmengen in Kombination mit Unsicherheiten zeigen, dass automatisierte Analysen der Datensätze in akzeptabler Analysezeit und damit akzeptabler Berechnungskomplexität notwendig sind.

Mit Clustering Verfahren existiert eine mögliche Analysemethodik zur Untersuchung von Ähnlichkeiten innerhalb eines Datensatzes. Aktuelle Verfahren integrieren jedoch nur einzelne Aspekte der komplexen Datensätze im Verfahren, mit einer teilweise nicht-linearen Berechnungskomplexität im Hinblick auf eine steigende Anzahl an Dateninstanzen sowie Dimensionen.

Diese Dissertation skizziert die einzelnen Herausforderungen der Prozessierung komplexer Daten in einem Clustering Verfahren. Darüber hinaus präsentiert die Arbeit einen neuartigen parametrisierbaren Ansatz zur Verarbeitung großer und komplexer Datensätze, genannt Fractal Similarity Measures, der die Datenmengen in log-linearer Analysezeit prozessiert. Durch das ebenfalls vorgestellte sogenannte unsichere Sortierungsverfahren für hochdimensionale Daten, stellt die dafür notwendigen Initialisierungsverfahren Gitter bereit. Mit Hilfe des neuen Konzepts des fraktalen Ähnlichkeitsmaßes bzw. dem fraktalen Informationswert analysiert das Verfahren die möglichen Cluster sowie die Dateninstanzen auf Ähnlichkeiten.

Zur Demonstration der Funktionalität und Effizienz des Algorithmus evaluiert diese Arbeit das Verfahren mit Hilfe eines synthetischen und eines reellen Datensatzes aus der Astronomie. Die Prozessierung des reellen Datensatzes setzt eine Vergleichbarkeit der gegebenen Spektraldaten voraus, weshalb ein weiteres Verfahren zur Vorprozessierung von Spektraldaten auf Basis des Hadoop-Rahmenwerks vorgestellt wird. Die Dissertation stellt darüber hinaus Ergebnisse des Clustering-Vorgangs des reellen Datensatzes vor, die mit manuell erstellten Ergebnissen von Domänennexperten qualitativ vergleichbar sind.

Abstract

A constant growth is currently present in many IT-affine areas. Scientific and especially astronomical datasets show complex properties e.g. uncertainties, a high number of dimensions as well as a high number of data instances. For example in astronomy, datasets consist of millions of data instances with thousands of dimensions, which are represented by the independent properties of the data. These magnitudes in combination with uncertainties require an automated data analysis within an acceptable timeframe.

Clustering methods represent a possible unsupervised approach of analysis to explore similarities within a dataset. Current methods just integrate single aspects of complex datasets with non-linear processing complexity w.r.t. the number of dimensions and data instances.

This dissertation describes the challenges mentioned of processing complex datasets using a clustering method. Moreover, this work presents a new configurable approach to analyze complex datasets called Fractal Similarity Measures which has a log-linear processing complexity. Using the so called uncertain sorting method for high dimensional data, an bootstrapping approach is able to provide meshs. This method analyzes the clusters and data instances w.r.t. similarities using the new concept of fractal information value and fractal similarity measure. Additionally, the method is able to be adapted by some parameters and considers properties of complex datasets.

To demonstrate the algorithm functionality, this work evaluates the fractal similarity measures method using a synthetic and a real astronomical dataset. To establish a comparability of real spectra, an additional method for preprocessing spectra based on hadoop is presented which provides a preprocessed version of the real dataset. Furthermore, this dissertation provides clustering results of the real dataset which are qualitatively comparable to manually created results of domain experts.

Danksagung

Mein besonderer Dank gilt Herrn Prof. Dr. Vincent Heuveline für das entgegen gebrachte Vertrauen auch in den Aufgabenstellungen neben dieser Dissertation sowie für die fruchtbaren Diskussionen der letzten Jahre.

Ich möchte mich ebenfalls ganz besonders bei meinem Mentor Herrn Dr. habil. Marcel Kunze für die allzeit gewährte und geschätzte Beratung und Unterstützung bedanken.

Weiterhin möchte ich meinen Dank an das Engineering Mathematics and Computing Lab und die AG Data Mining and Uncertainty Quantification (DMQ) richten, die eine motivierende Arbeitsatmosphäre ermöglichten.

Darüber hinaus bedanke ich mich bei Herrn Dr. Kai Lars Polsterer und der AG Astroinformatik (AIN) für die vielen Fachgespräche.

Ich danke meinen Kollegen des Universitätsrechenzentrums Heidelberg für gemeinsamen Projekte, die neben der Dissertation eine Abwechselung darstellten. Ich danke der Klaus-Tschira-Stiftung für die finanzielle Unterstützung dieser Arbeit.

Schließlich danke ich meiner Familie und insbesondere meiner Freundin Anja Uhrig für die außerordentliche Unterstützung in den arbeitsintensiven Phasen und das Korrekturlesen dieser Arbeit.

Inhaltsverzeichnis

Zusammenfassung **i**

Abstract **ii**

Danksagung **iii**

1 Einleitung **1**
1.1 Clustering von komplexen Datensätzen 2
1.2 Stand von Wissenschaft und Technik 3
1.2.1 Clustering Terminologie 3
1.2.2 Clustering in der Astronomie 3
1.3 Herangehensweise 4
1.3.1 Wissenschaftliche Beiträge 4
1.3.2 Aufbau dieser Arbeit 5

2 Clustering Verfahren **6**
2.1 Herausforderungen beim hochdimensionalen Clustering 6
2.1.1 Unterräume 7
2.1.2 Ähnlichkeitsdefinition 8
2.1.3 Ergebnisvalidierung 10
2.2 Herausforderungen im Clustering unsicherer Daten 12
2.3 Kategorien 14
2.3.1 Dichtebasierte Verfahren 15
2.3.2 Wahrscheinlichkeitsbasierte Verfahren 15
2.3.3 Partitionierende Verfahren 16
2.3.4 Hierarchische Verfahren 20
2.3.5 Gridbasierte Verfahren 26
2.3.6 Fraktale Verfahren 29
2.3.7 Zusammenfassung 30

3 Datenvorverarbeitung **32**
3.1 Beschreibung des Datensatzes 33
3.1.1 Datenerfassung der Spektrographen 34
3.1.2 Kalibrierung der Messungen 35

3.2 Der Clustering Datensatz 35
3.3 Hadoop 38
3.3.1 MapReduce 39
3.3.2 Hadoop 2.0 (YARN) 42
3.4 Map-Reduce Job zur Datenvorverarbeitung 44
3.4.1 Architektur 45
3.4.2 Filter 47

4 Fractal-Similarity-Measures Verfahren 52
4.1 Aufbau der Methodik 53
4.2 Fraktale Dimensionen und Box Counting Methode 55
4.3 Fraktaler Informationswert 58
4.4 Fraktales Ähnlichkeitsmaß 61
4.4.1 Optimierungen 63
4.5 Integration von Unsicherheiten 65
4.6 Bootstrapping-Verfahren 68
4.6.1 Spektrenspezifisches Bootstrapping 69
4.6.2 Splitting Maße 71
4.7 Algorithmus 72
4.7.1 Bootstrapping Step 72
4.7.2 Spektrenspezifisches Bootstrapping 74
4.7.3 Performance Step 75
4.7.4 Komplexitätsbetrachtung 77
4.8 Implementierung des Verfahrens 80
4.8.1 Zentrale Datentypen 80
4.8.2 Berechnung der komprimierten Spektren 85
4.8.3 Unsicheres Sortieren 86
4.8.4 Cluster-Manipulationen 88
4.8.5 Binning 90
4.9 Parameterbeschreibung 91
4.9.1 Boxgrößen 91
4.9.2 Binningfaktor 92

5 Evaluierung 94
5.1 Versuchsaufbau und -ablauf 94
5.2 Evaluierung anhand synthetischer Daten 96
5.2.1 Datenbeschreibung 96
5.2.2 Laufzeitanalyse 98
5.2.3 Parameterstudie 103
5.3 Evaluierung anhand der SDSS Daten 106
5.3.1 Laufzeitanalyse 106
5.3.2 Clustering-Ergebnisse 108

6 Zusammenfassung und Ausblick 114

6.1 Zusammenfassung 114
6.2 Ausblick 115

Quellenverzeichnis **117**
Literatur 117
Online-Quellen 126

Kapitel 1

Einleitung

Die Datenanalyse von Datensätzen im Tera- oder Petabyte-Bereich ist einer der aktuellen Trends in den Natur- bzw. Informationswissenschaften und der Industrie. Dieser Trend ist ein Aspekt des bekannten Begriffs Big Data. Der Gartner Hype Cycle for Emerging Technologies 2014 Bericht [49] klassifiziert das Themengebiet Big Data als eine der Technologien, die in den folgenden 2-5 Jahren als Standard-Technologie in Unternehmen etabliert sein wird. Tagtäglich werden sowohl im Privat-Sektor als auch in den Wissenschaften und der Industrie eine enorme Datenmenge produziert. Das geschätzte von Menschen erstellte Datenvolumen bis zum Jahr 2012 betrug 2.72 Zettabytes (2.7×10^{21} Bytes) und laut der Schätzung ist dessen Wachstum rapide: Alle zwei Jahre ist eine Verdopplung des Datenvolumen zu erwarten. Schätzungen sagen vorraus, dass bis Ende des Jahres 2015 ca. 8 Zettabyte an Daten erstellt worden sind [106]. Einzelne wissenschaftliche Experimente wie beispielsweise das Large Hadron Collider (LHC) Projekt des CERN erzeugen mehrere Petabytes pro Sekunde, die verarbeitet werden müssen [21]. Auch neue Technologien z.B. aus dem Bereich Internet of Things (IoT) tragen zu einem ständigen Wachstum des Datenvolumens bei, das zu analysieren ist. Zusätzlich zum ständig wachsenden Datenvolumen tragen Anforderungen zur Kombination und Integration verschiedener Datensätze dazu bei, die Komplexität der Analyse zu erhöhen.

In wissenschaftlichen Anwendungen und insbesondere in der Astronomie generieren Domänenexperten durch die Observationen ein erhebliches Volumen komplexer Daten aus einer Vielfalt an eingesetzten Instrumenten. Das Datenvolumen kann sowohl durch die schiere Größe auf einem Datenspeicher als auch durch die Anzahl der Messungen oder Gruppierungen von Messwerten ausgeprägt sein und repräsentiert jeweils eine eigene Herausforderung. Die performante Bereitstellung von Datenspeichervolumen kann z.B. durch eine verteilte Organisation des Datenspeichers erfolgen. Eine Analyse dieser Datenmengen bedarf jedoch eines skalierbaren Verfahrens, das in der Lage ist, diese Datenmengen in akzeptabler Zeit zu untersuchen. Um dies zu er-

möglichen, sind vollautomatisierte Analysen von Datensätzen (unsupervised analysis) ohne Nutzereingriff erforderlich.

Eine mögliche Fragestellung in der Analyse von Datensätzen ist die Suche nach Gruppierungen von Daten, die ähnliche Eigenschaften besitzen. Eine Methodik um diese Ähnlichkeitsanalyse von Datensätzen umzusetzen, ist das Clustering, das mit Hilfe eines parametrisierten Algorithmus entweder vollautomatisch oder mit einer Nutzerinteraktion die genannten Gruppierungen in den Datensätzen erkennen kann.

1.1 Clustering von komplexen Datensätzen

Die Clustering Analyse von astronomischen Datensätzen ist eine Herausforderung. Messinstrumente sowie die bemessenen Eigenschaften der observierten Objekte besitzen technisch und naturgemäße Unsicherheiten in den Messungen die eine präzise Messung einer Eigenschaft erschweren. Jedoch ist es möglich, die Unsicherheiten näherungsweise zu bestimmen, um so den Fehlerraum eines Messwerts einzugrenzen. Diese ermittelten Unsicherheiten sind in den Analysen zu berücksichtigen, da diese die Resultate einer Analyse deutlich beeinflussen können.

Durch die enorme Anzahl an Möglichkeiten Objekte im Universum, wie z.B. Sterne und Galaxien zu untersuchen, entstehen Datensätze mit einer großen Menge an observierten Objekten. Die von der Erde ausgehenden Observationen ermitteln für jedes beobachtete Objekt tausende Eigenschaften, z.B. in Form von Intensitäten elektromagnetischer Wellen verschiedenster Wellenlängen. Diese Messungen einzelner Intensitäten von Wellenlängen eines Objekts (sog. Spektraldaten) sind als unabhängig anzusehen und ergeben daher jeweils eine eigene Dimension, die bei den Untersuchungen zu berücksichtigen ist. Da diese Experimente kontinuierlich weiter betrieben werden, fallen täglich neue Daten an, die in einen solchen Prozess integriert werden können.

Diese vorhergehende Beschreibung skizziert die Vereinigung der folgenden Aspekte in einem komplexen Datensatz: Hochdimensionaltät, Unsicherheit und enorme Menge an Objekten. Diese Komplexität ist primär eine Herausforderung für den eingesetzten Algorithmus, da dieser alle drei Aspekte zu berücksichtigen hat, um Ergebnisse in einer akzeptablen Zeit liefern zu können. Dabei ist das Datenvolumen bzw. die Anzahl der Objekte sowie die Anzahl der Dimensionen in Kombination mit der Berechnungskomplexität bzw. Skalierbarkeit des eingesetzten Clustering Verfahrens ausschlaggebend für die zeitlich akzeptable Prozessierung eines Datensatzes. In dieser Arbeit wird eine entsprechende Methodik präsentiert, die komplexe Datensätze mit möglichst geringer Berechnungskomplexität bzgl. der Anzahl der Dimensionen und Objekte bearbeiten kann.

1.2 Stand von Wissenschaft und Technik

Die zwei folgenden Unterkapitel stellen kurz die in dieser Arbeit verwendete Terminologie im Clustering vor und vermittelt einen ersten Eindruck über die Möglichkeiten des Clustering in der Astronomie.

1.2.1 Clustering Terminologie

Im Clustering existieren eine Vielzahl an Definitionen mit teilweise gleichen Bedeutungen. Dieser Abschnitt definiert die in dieser Arbeit verwendeten Begrifflichkeiten.

Eine durch ein Clustering Verfahren erkannte Gruppierung an Elementen in einem Datensatz wird als Cluster bezeichnet. Ein Cluster beinhaltet genau die Elemente, die in mehreren Eigenschaften ähnliche Werte aufweisen. Elemente die keine Ähnlichkeiten in mehreren Eigenschaften besitzen, befinden sich i.d.R. nicht ein einem gemeinsamen Cluster. Elemente die keine Ähnlichkeiten mit anderen Elementen besitzen, werden als Ausreißer (outlier) bezeichnet. Die Definition von Ähnlichkeit ist in jedem Clustering Verfahren unterschiedlich, wie im Verlauf dieser Dissertation dargestellt wird.

Datensätze bestehen jeweils aus sog. Dateninstanzen. Beispielsweise repräsentieren die Reihen einer Observationstabelle die Dateninstanzen, die Spalten Objektnummer, Wellenlänge und Intensität die sog. Dimensionen. Dimensionen sind beim Clustering als unabhängige Eigenschaften zum Datenerstellungszeitpunkt von Dateninstanzen anzusehen. Durch eine Clustering Analyse können jedoch Korrelationen zwischen Dimensionen erkannt werden.

Als Datenraum wird in einem Datensatz mit n Dateninstanzen und d Dimensionen die Matrix der Größe nxd bezeichnet, die alle Dateninstanzen beinhalten. Datenräume können im Fall von $d = 3$ in einem üblichen Koordinatensystem visualisiert werden. Zeile in der Matrix entspricht einem Punkt im Koordinatensystem.

1.2.2 Clustering in der Astronomie

Observationsvorhaben in der Astronomie, wie das Sloan Digital Sky Survey Projekt (SDSS) [136], erstellen komplexe Datensätze mit einem Datenvolumen von mehreren hundert Gigabyte, bestehend aus mehreren Millionen Dateninstanzen, die wiederum tausende Eigenschaften besitzen. In den kommenden Jahren und Jahrzehnten werden Projekte wie das Square Kilometer Array (SKA) [135] Projekt etabliert, die gegenüber aktuellen Vorhaben ein exponentielles Wachstum an Daten besitzen. Das SKA Projekt wird Datensätze liefern, die aus mehr als 10^8 Dateninstanzen bestehen und ein Datenvolumen von mehreren Exabyte (10^3 Petabyte) besitzen [36, 101].

Insgesamt erleichtert ein entsprechendes Clustering Verfahren einem Do-

mänenexperten die Suche nach (un)bekannten Gruppierungen und besonderen Objekten. Clustering Verfahren können in der Astronomie eingesetzt werden, um die folgenden zwei Fragestellungen zu beantworten. Zum einen besteht die Möglichkeit durch eine automatisierte Methodik erstellte Cluster auf bisher unbekannte Eigenschaften zu analysieren. Beispielsweise ist es denkbar, neue spezifische Eigenschaften einer Klasse an astronomischen Objekten zu finden und diese anhand der spezifischen einzelnen Dateninstanzen genauer zu untersuchen. Darüber hinaus besitzen Clustering Resultate oft auch Ausreißer, die eine Art von Objekt darstellen, die zu keiner anderen Gruppe von Objekten passt. Sowohl die Suche nach der Gruppierung von Dateninstanzen als auch die Detektion von Ausreißern ist eine Aufgabe, die nicht ausschließlich manuell durchgeführt werden kann.

Die aktuell in der Astronomie angewandten Clustering Methoden, wie z.B. die in Kapitel 2.3 skizzierten Verfahren können oft eine Clustering Analyse mit einer nichtlinearen Skalierbarkeit bzgl. der Anzahl der Dateninstanzen sowie Dimensionen umsetzen [40]. Dies ist für die o.g. Magnituden an Datenmengen als offener Punkt zu sehen. Diese Arbeit liefert einen neuen Ansatz für das Clustering komplexer und großer Datensätze.

1.3 Herangehensweise

Die folgenden zwei Unterkapitel diskutieren die wissenschaftlichen Beiträge dieser Dissertation und geben einen Überblick über die Struktur der Arbeit.

1.3.1 Wissenschaftliche Beiträge

Diese Dissertation stellt eine Methodik vor, die große Datensatze bestehend aus hochdimensionalen und unsicheren Daten prozessieren kann. Zusammengefasst liefert diese Arbeit die folgenden Beiträge:

- **Hadoop basierte Datenvorverarbeitung von Spektraldaten**: Es handelt sich dabei um eine Methodik zur Vorverarbeitung eines Datensatzes zum Zweck des Clusterings. Ein Verfahren prozessiert Spektraldaten aus dem Bereich der Astronomie und bereitet diese für eine visuell orientierte hochdimensionale Ähnlichkeitsanalyse vor.
- **Clustering Verfahren zur Analyse großer und komplexer Datensätze**: Es wird das Fractal Similarity Verfahren vorgestellt, das in der Lage ist, eine skalierbare Ähnlichkeitsuntersuchung von Datensätzen bestehend aus mehreren tausend Dimensionen, Unsicherheiten in Eigenschaften sowie Millionen von Dateninstanzen automatisiert und ohne Nutzerintervention durchzuführen. Das Verfahren integriert die Unsicherheiten in den Analyseprozess und basiert auf dem Modell der Fraktalen Dimension.

- **Resultate der Prozessierung eines wissenschaftlichen Datensatzes**: Diese Arbeit evaluiert das Fractal Similarity Verfahrens anhand eines komplexen wissenschaftlichen Datensatzes aus dem Bereich der Astronomie. Die automatisiert erstellten Resultate sind in ihrer Qualität visuell vergleichbar mit Klassen, die von Domänenexperten manuell erstellt sind.
- **Verfahren zum Sortieren unsicherer Daten**: Das Fractal Similarity Verfahren ermöglicht unter anderem einen Sortiervorgang hochdimensionaler und unsicherer Daten durchführen zu können. Eine mögliche Sortierungsmethodik für unsichere Daten wird in dieser Arbeit ebenfalls dargestellt.

1.3.2 Aufbau dieser Arbeit

Diese Arbeit stellt zuerst die Herausforderungen der Clustering Analysen von komplexen Datensätzen in Kapitel 2 vor. Darüber hinaus gibt das Kapitel einen kategorisierten Überblick verschiedener Clustering-Verfahren und sowie eine kurze Diskussion im Hinblick auf Eigenschaften eines Verfahrens bzgl. der skizzierten Herausforderungen. Diese Arbeit zeigt die Funktionalität des Verfahrens anhand eines reellen Datensatzes, wie ein Clustering Vorgang vorzubereiten ist. Kapitel 3 skizziert die Aufgaben der allgemeinen Vorverarbeitung von Daten im Bezug auf ein hochdimensionales Clustering Verfahren und diskutiert die Eigenschaften des in dieser Arbeit verwendeten Datensatzes aus der Astronomie. Des Weiteren skizziert das Kapitel das entwickelte Verfahren zur Vorprozessierung des Datensatzes sowie das dafür eingesetzte Rahmenwerk. In Kapitel 4 stellt diese Dissertation das im Rahmen dieser Arbeit entstandene Fractal-Similarity-Measures Verfahren vor und erörtert dessen Grundlagen, Eigenschaften sowie die Algorithmik und Implementierung. Das darauf folgende Kapitel 5 evaluiert das Fractal-Similarity-Measures Verfahren anhand eines synthetischen sowie des reellen Datensatzes und bestätigt die Komplexitätsannahmen aus dem vorherigen Kapitel. Eine Zusammenfassung sowie einen Ausblick auf zukünftige Entwicklungsmöglichkeiten in diesem Umfeld gibt das abschließende Kapitel 6.

Kapitel 2

Clustering Verfahren

Dieses Kapitel diskutiert aktuell bekannte Clustering Verfahren mit Bezug auf die Zielsetzung dieser Arbeit, der Analyse von hochdimensionalen großen Datensätzen. Die Verfahren lassen sich in unterschiedliche Kategorien auf Basis der jeweiligen grundsätzlichen Analysestrategie einordnen. Für jede Kategorie existieren mehrere Verfahren, weshalb dieses Kapitel die einzelnen Analysestrategien mit Beispielen einzelner Verfahren statt einer vollständigen Liste aller Verfahren diskutiert. Darüber hinaus stellt dieses Kapitel die Herausforderungen der Clustering Analyse von hochdimensionalen, großen Datensätzen sowie die Kombination beider Eigenschaften in Daten dar.

2.1 Herausforderungen beim hochdimensionalen Clustering

Hochdimensionale Daten existieren in vielen industriellen und wissenschaftlichen Anwendungen bzw. Forschungsfeldern wie z.B. bei Observationen der Astronomie, Finanzdaten, Sensordaten aus dem Internet of Things (IoT) oder Web Dokumenten. Algorithmen, deren Entwicklung von hochdimensionalen Daten getrieben wurde, gehen von der Annahme aus, dass die Anzahl der Dimensionen größer ist als die Anzahl der Dateninstanzen [121]. Durch eine hohe Anzahl an Dimensionen existiert ein großer theoretischer Suchraum in dem Cluster gefunden werden können. Die Sammlung aller Dimensionen besteht im folgenden aus der Menge $D = d_1, .., d_n, |D| = d$. So existieren beispielsweise in einem diskreten Suchraum mit 50 Dimensionen bestehend aus einer jeweils diskreten Skala und Abstufung von 100 möglichen Werten (Abstufung $\frac{1}{100}$) bereits $100^{50} = 1 \times 10^{100} = \text{Googol}_{(10)}$[1] verschiedene Suchpunkte. In der Literatur wird diese expotentiell wachsende Komplexität durch die multivariaten Dimensionen auch als „curse of dimensionality“ bezeichnet [13, 14, 121]. Eine detaillierte Diskussion dieses Effekts

[1] $\text{Googol}_{(x)} = x^{x^2}$

bezogen auf den Bereich der Mustererkennung, ist in [17, S.33ff] beschrieben. So liefert das Clustering von hochdimensionalen Daten eine Vielzahl von verschiedenen zusätzlichen Problemstellungen, die in den nächsten Abschnitten diskutiert werden.

2.1.1 Unterräume

Mit steigender Anzahl der Dimensionen in einem Datensatz treten u.U. komplexe Problemstellungen auf. So könnten beispielsweise irrelevante Dimensionen oder Korrelationen von verschiedenen Unterräumen existieren, die eine Erkennung eines Clusters im volldimensionalen Analyseraum erschweren. Ein Unterraum im Fall des Clusterings oder der Datenanalyse entsteht aus einer arbiträren Kombination verschiedener Dimensionen ($D_{sub} \subset D$), wobei die Anzahl der Dimensionen $d_{sub} = |D_{sub}|, d_{sub} < d$ wird. Je größer die Anzahl der Dimensionen ist, desto mehr mögliche Kombinationen an einzelnen Dimensionen bzw. Unterräume existieren. Eine mögliche Herangehensweise die Komplexität der hohen Dimensionen und deren Kombinationen zu bearbeiten, wäre daher die automatisierte Dimensionsreduktion durch die Wahl geeigneter Unterräume. Die sog. Principal Component Analysis (PCA) ist ein gängiges Verfahren um einen hochdimensionalen Raum in einen Raum mit einer geringeren Anzahl von Dimensionen abzubilden [1, 70, 74].

Die automatisierte Selektion der relevanten Dimensionen bzw. Unterräume stellt einen eigenen Forschungsbereich dar. Diese Verfahren aus dem Bereich der sog. Dimensionsselektion versuchen eine möglichst minimierte Zusammenstellung relevanter Dimensionen aus dem gesamten Datensatz zu extrahieren. Eine Übersicht über diese Verfahren ist in [75] gegeben.

Die relevanten Dimensionen bzw. Features eines bestimmten Clusters können im Vergleich zu anderen Clustern variieren. Beispielsweise existiert ein Datensatz mit $|D| = 20$ Dimensionen mit einen Cluster, das primär Eigenschaften in den Dimensionen $1 - 10$ besitzt. Ein weiteres Cluster im gleichen Datensatz kann jedoch Eigenschaften besitzen, die in den Dimensionen $13 - 17$ ausgeprägt sind. Darüber hinaus erhöht sich die Wahrscheinlichkeit mit steigender Anzahl der Dimensionen Unterräume zu finden, in denen sich keine Cluster befinden, was die Berechnungszeiten einer Analyse ggfs. maximiert. Dies ist für z.B. eine PCA problematisch, da eine solche Methodik für jedes Cluster andere wichtige Komponenten liefert und daher die Cluster a-priori bekannt sein müssten. Eine globale Dimensionsselektion ist jedoch wegen möglicher clusterabhängiger (lokaler) Korrelationen der Dimensionen für Clusteringzwecke nicht geeignet [79].

Die Herausforderung einer Dimensionsselektion ist für das synthetische Beispiel eines drei-dimensionalen Datensatzes in Abbildung 2.1 visualisiert. Alle Cluster im Datenraum sind geometrisch betrachtet quaderförmig mit beliebigen Ausprägungen bzgl. der Dimensionen. Die einzelnen Punkte eines Clusters in den Datenräumen und Unterräumen stellen einzelne Daten-

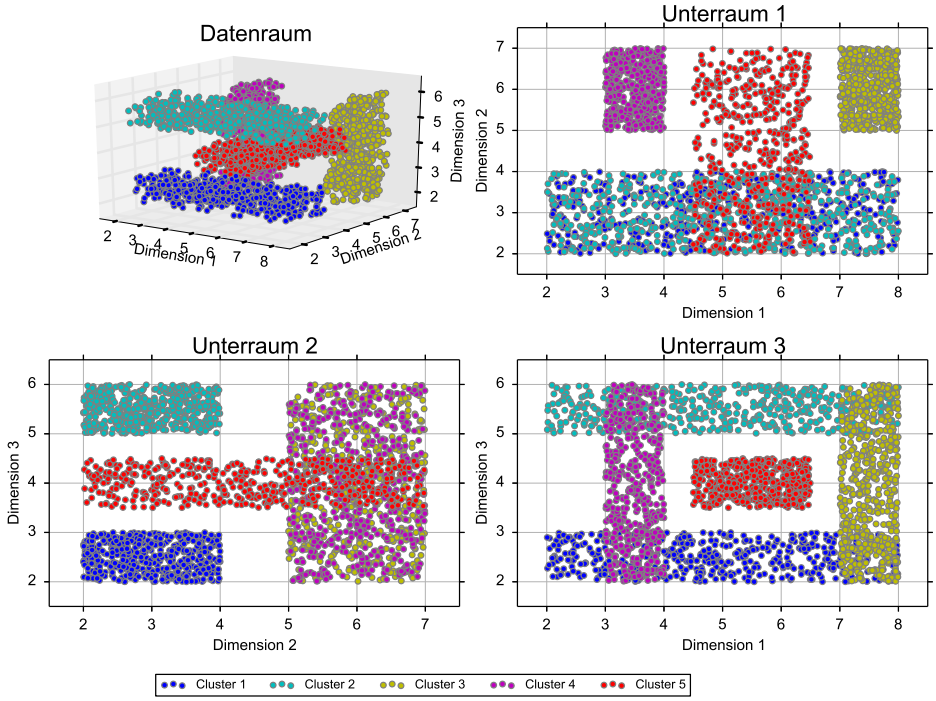


Abbildung 2.1: Darstellung eines synthetischen Datensatzes. Jede Achse in einem Schaubild repräsentiert eine Dimension des Datensatzes.

instanzen dar. Jeder Punkt ist in den entsprechenden Unterräumen bzgl. der jeweiligen Dimensionen an der gleichen Koordinate zu finden. Die dargestellten Unterräume entsprechen allen einzelnen möglichen Dimensionsselektionen. Während in Abbildung 2.1 im Datenraum alle Cluster deren Orientierungen und Ausprägungen visuell differenziert werden können, ist dies bei den Unterräumen nicht mehr möglich. Geometrisch analysiert befinden sich beispielsweise im *Unterraum 1* alle Cluster, jedoch ist es nicht möglich *Cluster 1* von *Cluster 2* und teilweise von *Cluster 3* zu unterscheiden. Diese Unterscheidung ist in *Unterraum 2* gegeben, jedoch nicht mehr die Unterscheidung von *Cluster 2* und *Cluster 3*. Dies zeigt, dass eine Dimensionsselektion keine globale Sicht oder Repräsentation eines hochdimensionalen Raums geben kann und Wahl der Unterräume für einzelne Cluster variieren kann, was beim Clustering Vorgang zu berücksichtigen ist.

2.1.2 Ähnlichkeitsdefinition

Die folgenden Abschnitte beschreiben die Herausforderungen in der Definition bzw. Wahl der Ähnlichkeitsmetrik, die daraus resultierende Herausforderung in der Vergleichbarkeit der Verfahren sowie den Einfluss hochdimensionaler Daten auf die Verwendung einer Distanzmetrik.

Variation der Ähnlichkeitsbegriffe

Clustering wird in der Literatur häufig (z.B. in [3, 73, 79, 121]) abstrakt beschrieben als Verfahren durch das Datensätze in Untermengen (Cluster) geteilt werden. Dabei beinhaltet eine Untermenge genau die Daten, die innerhalb dieser Untermenge ähnlich zueinander sind. Daten verschiedener Cluster sind nicht ähnlich zueinander.

Die Abstrahierung des Begriffs ergibt einen Interpretationsspielraum in der Fragestellung der Ähnlichkeit. Ähnlichkeit ist im Umfeld des Clusterings von hochdimensionalen Daten in verschiedenen Verfahren und Anwendungsbereichen unterschiedlich definiert. Verfahren die z.B. eine Dichte in hochdimensionalen Subräumen unter Verwendung von Distanzmetriken bemessen, definieren Ähnlichkeit als Nähe z.B. bzgl. der Euklidischen Distanz. Sei

$$dist_{ij} = \left(\sum_{k=1}^{d} |x_{ik} - x_{jk}|^q \right)^{\frac{1}{q}} \tag{2.1}$$

die Minkowski Distanz. Dabei respräsentiert $dist_{ij}$ die Distanz zwischen zwei Dateninstanzen x_i und x_j. Die Distanz berechnet sich aus der Summe der absoluten Differenzen der d-fach vorhandenen Dimensionswerte der beiden Dateninstanzen. Die Euklidische Distanz ist eine spezielle Form der Minkowskidistanz mit $p = 2$.

Andere Verfahren erkennen Muster in verschiedenen Unterräumen und vergleichen die Unterschiede anhand dieser Unterräume. Weitere Verfahren erkennen Ähnlichkeiten anhand von Raumteilungen oder der Wahrscheinlichkeit von Elementen zu einer bestimmten Verteilung zu passen. Durch die unterschiedlichen Metriken für Ähnlichkeit und die daraus resultierenden unterschiedlichen Verfahren kann eine Kategorisierung der Verfahren anhand dieser Metriken vorgenommen werden.

Die einzelnen Clustering Kategorien besitzen i.d.R., wie in diesem Abschnitt dargestellt, eigene Notationen bzw. Definitionen von Ähnlichkeit. Aufgrund des intrinsischen Zusammenhangs zwischen Algorithmus und Definition können diese allerdings nicht zwischen den einzelnen Kategorien von Verfahren ausgetauscht werden [79].

Distanzmetriken im hochdimensionalen Datenraum

Es existiert beispielsweise die Kategorie der distanzbasierte Verfahren, die mit Konzepten wie Nähe (z.B. [95]) oder Nachbarschaften (z.B. [42]) arbeiten. Mit steigender Anzahl an Dimensionen können diese Verfahren weniger zuverlässige Ergebnisse liefern [2, 14, 79]. Die Zuverlässigkeit der Ergebnisse wird in den genannten Publikationen indiziert mit folgender Gleichung.

$$d \to \infty, \; f = \frac{dist_{max} - dist_{min}}{dist_{min}} \to 0 \tag{2.2}$$

Gleichung 2.2 beschreibt den Effekt bei steigender Dimensionalität d der Daten bei distanzbasierten Verfahren der maximale Abstand $dist_{max}$ sowie das minimale Pendant auf den gleichen Wert konvergieren, was die Ergebnisse qualitativ mindert. Das heisst, wie in Gleichung 2.2 dargestellt, konvergiert der Unterschiedsfaktor f gegen 0.

Dies ist für die Differenzierung von Unterschieden durch eine distanzbasierte Metrik in hochdimensionalen Räumen als grundsätzliches Problem anzusehen. Wenn die o.g. Extrema geometrischer Distanzen nicht mehr zu unterscheiden sind, können darauf basierende Clustering-Algorithmen das jeweilige Optimierungsproblem nicht korrekt lösen. Darüber hinaus ist bei einer Distanzmetrik die nicht für hochdimensionale Daten adaptiert wurde der Einfluss einzelner Dimensionen i.d.R. identisch gewichtet. Bei divergierender Relevanz einzelner Dimensionen sind deren Gewicht in der Distanzmetrik zu adaptieren, was sich bei einer Anzahl an Dimensionen von 1×10^3 und mehr als arbeitsintensiv darstellt solange die Gewichtung manuell vorgenommen wird. Die Minkowski-Metrik

$$dist_{ij} = \left(\sum_{k=1}^{d} w_k |x_{ik} - x_{jk}|^q \right)^{\frac{1}{q}} \tag{2.3}$$

beschreibt eine mögliche Herangehensweise die geometrische $dist_{ij}$ zwischen zwei Elementen unter Einfluss eines Parameters q berechnet ($q = 2$ entspricht der Euklidischen Distanz). x_{ik} repräsentiert den Wert für das i-te Element, der k-ten Dimension im Datensatz sowie w_k das dimensionsabhängige Gewicht. Aktuell existieren keine Algorithmen, die diese Gewichte automatisiert anpassen. In [2, 66] werden Verbesserungen für distanzbasierte Verfahren vorgestellt, um diesen Effekt zu minimieren, jedoch wird der Effekt nicht eliminiert.

Ähnlichkeitsmetriken in Clustering-Algorithmen können sowohl explizit als auch implizit definiert werden. Explizite Metriken sind beispielsweise distanzbasierte Metriken, die durch eine wohl definierte Notation unabhängig vom Algorithmus etabliert sind und auch zwischen Algorithmen ausgetauscht werden können. Dagegen sind implizite Ähnlichkeitsmetriken in den Algorithmus integriert, wie z.B. bei den Gridbasierten Verfahren (siehe Kapitel 2.3.5). Die implizite Ähnlichkeitsmetrik ist nicht explizit im Verfahren definiert.

2.1.3 Ergebnisvalidierung

Einhergehend mit der Definition der Ähnlichkeitsmetrik ist die Validierung der Ergebnisse einer Clustering-Analyse bzw. die Analyse der Cluster-Validität (cluster validity). Cluster die mit Hilfe eines nicht-überwachten Algorithmus erstellt wurden, besitzen eine Ergebnisqualität, die von ihrer impliziten oder expliziten Ähnlichkeitsmetrik abhängt. In den Ergebnissen sollen im

Allgemeinen die Datenpunkte gruppiert sein, die signifikant ähnlich bzgl. einer Ähnlichkeitsmetrik sind. Die Validierung dieses Ergebnisses soll z.B. anhand der Zuweisung von Dateninstanzen zu einem Cluster oder anhand der Anzahl der Cluster selbst erfolgen. Beide Varianten stellen eine Herausforderung dar, da bei nicht-synthetischen Datensätzen insbesondere bei neu durchgeführten wissenschaftlichen Experimenten i.d.R. keine vorab definierten Cluster oder Klassen existieren.

Ein solcher, vorab definierter Datensatz, wird annotierter Datensatz (labeled dataset) genannt. Speziell bei hochdimensionalen Datensätzen mit mehr als drei Dimensionen ist die Erstellung aufgrund der herausfordernden Wahl einer adäquaten Visualiserungsmöglichkeit, die clusterabhängig alle notwendigen Dimensionen einbezieht, komplex. Annotierte Datensätze existieren ggfs. für bestimmte Anwendungsfelder und sind oft synthetisch erstellt. Ebenfalls anwendungsspezifisch ist es möglich Validierungsverfahren und Metriken zu entwickeln, wie z.B. in [124] für den spezifischen Fall der Genexpressionsdaten. Zum Zeitpunkt der Erstellung dieser Arbeit existiert kein automatisiertes Validierungsverfahren für die Analyse eines Clusterergebnisses für Spektraldaten in der Astronomie.

Ohne einen annotierten Datensatz ist es nicht möglich den absoluten Fehler in Form von falsch klassifizierten Dateninstanzen zu detektieren. Neben einer manuellen Inspektion der Ergebnisse durch einen Experten ist es prinzipiell möglich, stattdessen die geometrischen oder statistischen Eigenschaften der Cluster zu untersuchen. Die Validität eines Cluster kann ein Validierungsverfahren quantitativ durch die folgenden Kategorien an Kriterien ermitteln:

- Externe Kriterien ermöglichen eine Evaluation der Resultate auf Basis einer bereits existierenden Partitionierung oder eine bereits annotierten Clusteringergebnises. Während eine Partitionierung ausschließlich eine Aufteilung des Datenraums darstellt, ist ein annotiertes Clustering Resultat einer Dateninstanz-spezifischen absoluten Lösung. Die korrespondierenden Cluster oder Partionen sind geometrisch oder statistisch zu analysieren. Des Weiteren können beispielsweise Null-Hypothesentests mittel Monte Carlo Simulation durchgeführt werden [60].
- Interne Kriterien stellen die Möglichkeit dar, ohne externe annotierte Ergebnisse das Resultat zu untersuchen, d.h. die Analyse bezieht sich nur auf die inhärenten Eigenschaften des Ergebnisses bzw. des Datensatzes. [96] beschreibt eine Monte Carlo basierte Studie bekannter Metriken bzw. interne Kriterien zur Analyse eines Clustering Ergebnisses.

Eine weiterführende Beschreibung und Diskussion der o.g. Kriterien kann in [60–62] nachgelesen werden.

Generell stellen diese Validierungsmethoden und Kriterien die Möglichkeit dar, die Ergebnisse bzgl. eines Modells, Metriken oder annotierten Daten

zu prüfen, um damit die Präzision eines Algorithmus zu evaluieren. Damit kann eine Detektion von fehlerhaft oder nicht zugewiesenen Dateninstanzen erfolgen. Je nach Anwendungsfall existiert jedoch keine Option zur objektiven Validierung mittels der o.g. Kriterien, weshalb in diesen Fällen eine manuelle Inspektion und Validierung der Ergebnisse durch einen Domänenexperten notwendig ist. Diese Art der Ergebnisinterpretation ist subjektiv und daher abhängig vom jeweiligen Inspekteur. Eine Unterstützung dessen Arbeit ist mit Visualisierungsmethoden für hochdimensionale Daten ggfs. möglich und herausfordernd zugleich, da eine anwendungsabhängige Visualisierungsmöglichkeit existieren muss, die bei einem Datenraum von $d > 3$ Dimensionen eine adäquate Dimensionsreduktion durchführen muss.

2.2 Herausforderungen im Clustering unsicherer Daten

Normalerweise beschäftigen sich Clustering Methoden mit der Fragestellung der nicht überwachten Analyse von konkreten präzisen Werten. Die Analyse von unsichereren Datensätzen ist darüber hinaus ein zusätzlicher Aspekt, der einem präzisen Wert ggfs. für jede Dimension eine Varianz oder Wahrscheinlichkeitsdichtefunktion hinzufügt. Das Clustering unsicherer Daten wird in der Literatur unter dem Begriff Uncertainty Clustering (UC) geführt. UC befasst sich im Gegensatz zum Fuzzy Clustering [16, 58, 123] (dem probabilistischen Zuweisen von Dateninstanzen zu einzelnen Clustern) mit dem eindeutigen, sog. harten Clustering von Dateninstanzen und deren Unsicherheiten in der Datenquelle. Unsicherheiten in Daten sind speziell bei wissenschaftlichen Experimenten wie z.B. Observationen in der Astronomie vorhanden und können auf verschiedene Ursachen bzw. Kombinationen der Ursachen zurückzuführen sein:

- Systemfehler, wie z.B. auflösungsbasierte Diskretisierungen, Messfehler oder Toleranzen in Instrumenten, Verfahrensfehler, Verluste durch Übertragungen, können ggfs. Varianzen in den Daten erzeugen, die nicht durch Kalibrierung eliminiert werden können. Diese Art der Fehler können i.d.R. quantifiziert werden. Diese Quantifikationen sind beim Clustering zu beachten.
- Numerische Fehler, die bei modellgestützten Untersuchungen in z.B. Simulationen auftreten. Durch die Diskretisierung von eigentlich kontinuierlichen Daten entstehen Fehler, die ebenfalls statistisch quantifiziert werden können.
- Des Weiteren sind Datenfehler die bei naturwissenschaftlichen Experimenten inhärent sind, ebenfalls als Unsicherheiten zu klassifizieren. Diese Art von Fehlern, z.B. gegeben durch variierende Materialeigenschaften können nicht reduziert werden.

Eine detailliertere Abhandlung von Fehlerquellen und deren Behandlung bei Experimenten in der Physik ist in [46] gegeben. Die Beschreibung von möglichen Unsicherheiten die bei numerischen Simulationen auftreten können sowie ein Rahmenwerk das diese entsprechend in die Simulationen integriert, ist in [102] skizziert. Alle Unsicherheiten, die nicht reduziert werden können, werden als sog. aleatorische Unsicherheiten bezeichnet. Die Unsicherheiten die systemseitig vermieden werden können, sind als erkenntnistheoretische Unsicherheiten bekannt. Für beide Arten von Unsicherheiten ist eine Quantifizierung für ein Clustering-Vorhaben wichtig, da die Unsicherheiten einen Einfluss auf das Ergebnis bzw. dessen Qualität haben können. Abhängig von der eingesetzten Ähnlichkeitsmetrik beeinflussen Unsicherheiten die Qualität der Berechnung. Eine weitere Sicht auf Unsicherheiten ist die Unterscheidung zwischen künstlich erstellten und inhärenten Unsicherheiten. Systematische sowie numerische Fehler zählen zu den künstlich erstellten Fehlern. Varianzen in natürlichen Eigenschaften der instrumentierten Objekten sind als inhärente Unsicherheiten bezeichnet.

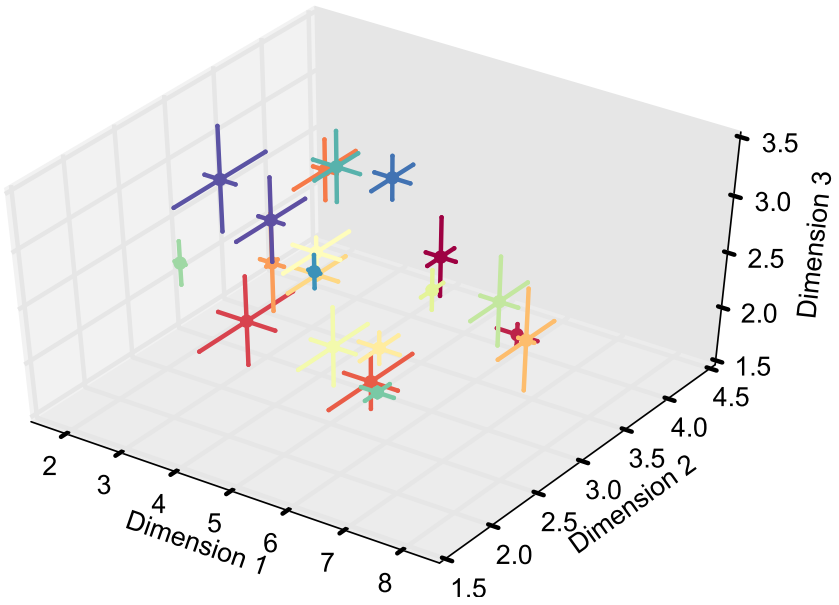


Abbildung 2.2: Darstellung der Unsicherheiten eines synthetischen Datensatzes. Jede Achse repräsentiert eine Dimension des Datensatzes. Die einzelnen Datenpunkte sind umgeben von Fehlerschranken, die die Unsicherheit der Dateninstanz in der jeweiligen Dimension repräsentieren. Die Fehlerschranken in den jeweiligen Dimensionen sind in dieser Visualisierung dargestellt durch die ausgeprägte Linien. Zur vereinfachten Differenzierung der Elemente ist eine farbliche Markierung der Elemnte und ihrer Fehlerschranken gegeben.

Abbildung 2.2 zeigt exemplarisch anhand eines synthetischen Datensatzes ein Cluster, dargestellt durch dessen Dateninstanzen in Form von Punkten in einem drei-dimensionalen Datenraum. Zusätzlich weisen die Punkte Unsicherheiten (in diesem Beispiel dimensionsabhängige Varianzen) auf, die durch die Fehlerschranken in Form von Linien markiert sind. Jeder Datenpunkt besitzt eine eigene Unsicherheit, die bei einem Clustering Vorgang zu beachten ist. Alle Punkte können sich in dem Raum, der durch die Fehlerschranken aufgespannt ist, befinden.

Clustering Verfahren, die unsichere Daten prozessieren, müssen Methoden implementieren, die eine explizite Integration der Unsicherheiten in den Analyseprozess vornehmen um korrekte Ergebnisse zu produzieren. Integrationen können entweder im jeweilige Algorithmus mittels a-priori Behandlung in Form eines Rahmenwerks z.B. dargestellt in [126] oder alternativ in einer Ähnlichkeitsmetrik vorgenommen werden. Eine mögliche Ähnlichkeitsmetrik, die durch die Erweiterung einer dichtebasierten Metrik zu einer probabilistischen Dichtemetrik modifiziert wurde, kann in [80] nachgelesen werden. Das Rahmenwerk ermöglicht eine vorgeschaltete Behandlung der Unsicherheiten, während die modifizierte Dichtemetrik innerhalb des dichtebasierten Algorithmus agiert. In beiden Fällen ist entweder ein parametrisiertes Modell oder ein bereits dateninstanz- und dimensionsabhängiger quantifizierter Wert oder Wahrscheinlichkeitsdichtefunktionen der Unsicherheit des Datensatzes a-priori zu bestimmen. Diese a-priori Definition der Unsicherheiten ist bereits eine eigene Herausforderung, da dies eine anwendungsfallabhängige Fragestellung ist, die i.d.R. von einem Domänenexperten gelöst werden kann. Aktuell erfordert die Integration von Unsicherheiten in einem Clustering-Vorhaben die Definition einer Dichtefunktion, alternativ die empirische Ermittlung der Unsicherheiten während des Experiments.

Hochdimensionale, unsichere und große Datensätze sind eine weitere Herausforderung in der Clustering-Analyse. Ähnlichkeitsmetriken und Algorithmen sind so zu gestalten, dass die Berechnungskomplexität sich durch die Integration der Unsicherheiten nicht gravierend verschlechtert. Eine Balance zwischen der Berechnungskomplexität bzw. Laufzeit und der Qualität des Ergebnisses ist zu finden. Die Metriken müssen sowohl in der Lage sein die Unsicherheiten einzelner Dimensionen als auch die hohe Dimensionalität in der Berechnung zu integrieren. Eine Übersicht über einige aktuelle adaptierte Verfahren für das Clustering unsicherer Datensätze kann in [4] nachgelesen werden.

2.3 Kategorien

Die in dieser Arbeit vorgestellten Methoden können in einzelne Kategorien eingeteilt werden, die in folgenden Abschnitten durch einzelne repräsentierende Algorithmen beschrieben sind.

2.3.1 Dichtebasierte Verfahren

DBSCAN [42] ist das bekannteste Clusteringverfahren, das auf ein dichtebasiertes Konzept setzt. Für DBSCAN ist Dichte definiert als Markierung der Verbindung zweier Objekte p und q. Diese ist definiert als direkt dichteverbunden (directly density connected) soweit q im Radius (Eps) von p liegt ($dist(p,q) \leq Eps$). Falls eine Mindestanzahl an Punkten ($MinPts$) um p direkt dichteverbunden sind, wird p als Kernpunkt (core point) bezeichnet. Diese dadurch entstehende Nachbarschaft ist vom Radius abhängig und ist folgenderweise definiert.

$$NEps(p) = \forall q \in D, dist(p,q) \leq Eps. \tag{2.4}$$

Zwei Punkte p und q sind dichte-erreichbar (density reachable), falls ein weiterer Punkt o von beiden Punkten direkt dichteverbunden ist. Ein Cluster entsteht in diesem Verfahren durch einen Kernpunkt und seine Umgebung, die dichte-erreichbar ist. Rauschen (Noise) ist in diesem Verfahren definiert als ein Punkt der nicht dichteverbunden ist, ein sog. Rauschpunkt (noise point). Falls ein Punkt zu weniger als $MinPts$-Punkten dichteverbunden ist, wird dieser Punkt als Randpunkt (border point) bezeichnet. Der dazugehörige Algorithmus initialisert sich durch eine randomisierte Auswahl eines Punkts p und berechnet dessen $NEps(p)$ bzgl. der Datengrundlage. Die Berechnungskomplexität des Algorithmus ist stark abhängig von der gewählten Implementierung und kann je nach Datenstruktur zwischen $O(n)$ und $O(n^2)$ variieren [3].

OPTICS [8] verbessert DBSCAN durch eine halb-automatisierte statt automatisierte Analyse der Daten, jedoch mit einer multiplen Wahl von Eps und $MinPts$. Die Erweiterung ermöglicht die Ermittlung clusterabhängiger Parameter bzw. Eigenschaften in einem realen Datensatz zur Beschreibung individueller Cluster. Beispielsweise hebt die Erweiterung die Limitierung auf, dass unterschiedliche Cluster identische Dichten besitzen müssen oder deren Anzahl der Repräsentanten kleiner als der Schwellwert $MinPts$ ist.

Aufgrund der Abhängigkeit zu einer Distanzmetrik ist der Einsatz von DBSCAN jedoch bei hochdimensionalen Daten wie in Kapitel 2.1.2 skizziert, nicht geeignet [121]. Ein weiterer offener Punkt in DBSCAN ist die direkte Berücksichtigung von Unsicherheiten im Verfahren. Bei einem entsprechenden Anwendungsfall müssten diese in einer speziellen, noch zu entwickelnden Distanzmetrik integriert werden. Zum Zeitpunkt der Erstellung dieser Arbeit existiert keine entsprechende Metrik im Zusammenhang mit DBSCAN.

2.3.2 Wahrscheinlichkeitsbasierte Verfahren

Im Allgemeinen arbeiten probabilistische Verfahren mit dem Konzept der Approximation eines Modells an den Datenraum, Unterräume oder an Cluster. Ein Cluster entspricht demnach einer oder einer Kombination von para-

metrisierten Wahrscheinlichkeitsdichtefunktionen (propability density function, PDF). Die Sammlung aller Cluster repräsentiert eine Liste an parametrisierten Wahrscheinlichkeitsdichtefunktionen. In der Regel ist ein flexibles Modell zu adaptieren, das mit Hilfe eines Algorithmus die Parameter für die Cluster erlernt. Die Modelle in Form der o.g. Kombinationen an Wahrscheinlichkeitsdichtefunktionen bestehen aus sog. Mischverteilungsmodellen (Mixture Models, MM), die aus einer Summe von Komponenten bestehen. Eine Komponente repräsentiert eine individuelle Verteilung, die ein Cluster modelliert. Die jeweiligen Komponenten können sowohl durch eine bestimmte Verteilung mit komponentenabhängigen Parametern sowie jeweils durch eine eigene Verteilung repräsentiert werden. Bekannte Modelle sind die gauss'schen Mischverteilungsmodelle (Gaussian Mixture Models), sowie die bernoulli Mischverteilungsmodelle (Bernoulli Mixture Model). Die Modelle werden mit Hilfe des Expectation Maximization (EM) Algorithmus [97] randomisiert initialisiert und durch den Algorithmus so optimiert, dass jede Komponente des Mischverteilungsmodells ein Cluster repräsentieren kann. Die Berechnungskomplexität des EM Algorithmus ist nicht deterministisch und variiert je nach implementierter Wahrscheinlichkeitsdichtefunktion. Durch die Flexibilität der Mischverteilungsmodelle ist es mit entsprechendem Berechnungsaufwand möglich, auch hochdimensionale Cluster sowie deren Unsicherheiten zu approximieren.

2.3.3 Partitionierende Verfahren

Die Kategorie der partitionierenden Verfahren unterteilt den Datenraum entweder aktiv mittels Integration von mehrdimensionalen Ebenen bzw. Trennungskörpern oder passiv durch Abgrenzung mittels Clusterrepräsentanten. Das bekannteste partitionierende Verfahren ist der K-Means Algorithmus [92]. Dieses Verfahren benötigt einen manuell zu definierenden Wert K, der die Anzahl der zu findenden Cluster repräsentiert. Ein Cluster C_i wird dabei durch einen virtuellen zentralen Punkt (Centroid) m_i und den Punkten repräsentiert, die genau diesem Centroid am nächsten sind.

Um die Cluster zu erhalten selektiert der distanzbasierte Algorithmus initial genau K Centroids, die arbiträr im Datenraum platziert werden. Daraus resultieren zufällige virtuelle Punkte, die nicht zwangsweise einen echten Datenpunkt repräsentieren. Der Algorithmus [89, 95] wiederholt die folgenden Schritte, bis ein Konvergenzkriterium zutrifft:

1. Erstellung der Cluster durch Zuweisung aller Instanzen zu einem Centroid. Die Zuweisung erfolgt an den Centroid mit der geringsten Distanz. Ein Datenpunkt wird genau einem Centroid zugewiesen.
2. Durch die ggfs. neu hinzugekommen Instanzen aus dem vorherigen

Schritt wird der Centroid m_i durch folgende Gleichung neu berechnet.

$$\forall d \in D : m_i^{(d)} = \frac{1}{|C_i^{(d)}|} \sum_{x \in C_i} x^{(d)} \tag{2.5}$$

Bei K-Means bestimmt der Algorithmus den Centroid dimensionsweise ($m_i^{(d)}$) durch die Bildung des Mittelwerts pro Cluster, wie Gleichung 2.5 darstellt.

Das Konvergenzkriterium kann frei gewählt werden. Ein oft verwendetes Konvergenzkriterium ist die Verschiebung der Centroids. Wurde nach einer Iteration kein Centeroid verändert (d.h. $m_{iIteration\ q} = m_{iIteration\ q+1}$), ist der Algorithmus konvergiert. In S_{SQE} ist eine Fehlerfunktion (sum of squared errors) über alle Cluster C dargestellt.

$$S_{SQE} = \sum_{k=1}^{K} \sum_{x_i \in C_k} ||x_i - m_k|| \tag{2.6}$$

Der Fehler S_{SQE} berechnet sich dabei aus der Summe der quadrierten Differenzen zwischen dem Punkt x_i und dem Mittelpunkt des Clusters m_k. Angemerkt sei, dass die distanzbasierte Differenz folgendermaßen definiert ist.

$$||x_i - m_k|| = \sum_{l=1}^{d} (x_{il} - m_{kl})^2 \tag{2.7}$$

K-Means bildet die Differnenz durch die Summe der quadrierten Differenzen von Werten korrespondierender Dimensionen. Da durch den initialen Schritt der randomisierten Selektion der ersten Centroid-Position für mehrfache Durchführungen unterschiedliche Ergebnisse entstehen können, ist es möglich K-Means mehrfach auszuführen um so ggfs. die Fehlerfunktion S zu mininmieren. K-Means ist einer der Clustering-Algorithmen dessen Konvergenz bewiesen ist [110, 121].

Es existiert eine Abhängigkeit zwischen der Konvergenzgeschwindigkeit des Algorithmus und der Methode, wie die initialen Centroids selektiert werden. Arbeiten bzgl. der Modifikation der randomisierten initialen Definition verbessern diesen Aspekt [3, 20, 96] unter anderem durch den Einsatz von Konzepten aus anderen Kategorien von Clustering Algorithmen z.B. mittels eines dichtebasierten Ansatzes [63]. K-means++ [9] optimiert die Selektion durch eine Maximierung der Abstände zwischen den Centroids durch ein probabilistisches Verfahren. Des Weiteren existieren für die Wahl eines geeigneten Werts für K einige unterschiedliche Lösungsansätze z.B. in [22, 100, 116].

Neben den o.g. Verbesserungen existieren weitere Varianten des K-Means Algorithmus:

- Der K-Medians Algorithmus tauscht die Berechnung des Centroids aus. Anstatt der Berechnung der durchschnittlichen Dimensionswerte als Cluster Mittelpunkt wird der jeweilige Median kalkuliert. Dadurch ergibt sich eine andere Fehlerfunktion, wie folgt dargestellt ist.

$$S_{median} = \sum_{k=1}^{K} \sum_{x_i \in C_k} ||x_i - median_k|| \tag{2.8}$$

  Das Grundgerüst von Gleichung 2.8 ist identisch zu dem aus Gleichung 2.6, jedoch ist die der Cluster Mittelpunkt ersetzt durch den Median basierten Clustermittelpunkt. Durch die Verwendung des Medians ist K-Median robuster gegenüber Ausreißern.
- Mit Hilfe der Fuzzy K-Means [15] Variante ist es möglich, die fest definierte Zuordnung einer Dateninstanz zu einem Centroid durch einen Grad der Zugehörigkeit bzw. Gewicht zu ersetzen. Dabei existiert ein Wert $u_{ik}(u_{ik} \in [0,1], \sum_{k=1}^{K} u_{ik} = 1)$, der die Gewichtung eines Elements x_i zu einem Cluster C_k beschreibt. Das normale K-Means Clustering ist ein Spezialfall von Fuzzy K-Means, wobei u_{ik} den Wert 0 oder 1 annehmen kann. Fuzzy K-Means minimiert die folgende Fehlerfunktion:

$$S_{FK} = \sum_{k=1}^{K} \sum_{x_i \in C_k} u_{ik}^{\alpha} |x_i - m_k|^2. \tag{2.9}$$

  Die weitere Alternative zur Berechnung des S_{SQE} Fehlerkriteriums ist die beschriebene Gleichung 2.9. Hierbei stellt α einen Parameter dar, der die Form der Cluster verändert (i.d.R. wird $1 \leq \alpha \leq 2.5$ verwendet [105, 115]):

$$u_{ik} = \frac{1}{\sum_{j=1}^{K} \left(\frac{x_i - m_k}{x_i - m_j}\right)^{\frac{2}{\alpha - 1}}}. \tag{2.10}$$

  Die clusterspezifische Gewichtung einer Dateninstanz u_{ik} ist definiert über die Formel in Gleichung 2.10. Hierbei sind m_j sowie m_i Mittelpunkte eines Clusters
- Intelligent K-Means (IK-Means) [27, 28, 105] ersetzt das Auswahlverfahren der Centroids durch einen geschachtelten 1-Means Algorithmus, der mit einem sog. Anormalen Muster arbeitet. Das anormale Muster entsteht durch die Selektion der Dateninstanz, die bzgl. einer Distanzmetrik vom aktuellen Centroid am weitesten entfernten entfernt liegt. Des Weiteren besitzen Cluster eine schwellwertbasierte Auswertung der Mindestgröße. Soweit die Anzahl der Dateninstanzen in einem Cluster unter den Schwellwert sinkt, wird das Cluster aufgelöst und die darin enthaltenen Elemente repräsentieren Ausreißer. Insgesamt wird dieses Verfahren gegenüber dem Standard K-Means

Verfahren deterministisch, da durch die identischen Ausgangssituation der konsistenten Wahl der initialen Centroids immer die gleichen Ergebnisse resultieren [105].

- Die Verfahren Kernel K-Means [109] sowie gewichteter KMM (Weighted Kernel K-Means) [37] integrieren, anstatt der Euklidischen Distanz als Basis der Optimierungsfunktion, nicht-lineare Kernelfunktionen. Alle jeweiligen Kernel (ϕ) erstellen implizit eine Projektion des Datenraums in einen höherdimensionalen Suchraum. Dies verändert die objektive Fehlerfunktion im Falle des gewichteten Kernels:

$$S_{WK} = \sum_{k=1}^{K} \sum_{x_i \in C_k} w_i |\phi_i - m_k|^2. \tag{2.11}$$

  Dabei repräsentiert ϕ_i die entsprechende Kernelfunktion, die in Abhängigkeit einer Dateninstanz parametrisiert ist. w_i entspricht dem Dateninstanz abhängigen Gewicht. Des Weiteren berechnet der gewichtete KMM Algorithmus die Centroids mit dem folgenden Berechnungsschritt.

$$m_k = \frac{\sum_{x_i \in C_k} w_i \phi_i}{\sum_{x_i \in C_k} w_i} \tag{2.12}$$

  Die meistgenutzten Kernel in diesem Verfahren sind der polynomielle Kernel, Gauss'sche Kernel sowie sigmoide Kernel die in [37] ebenfalls skizziert sind. In der nicht gewichteten Variante dieses Verfahrens wird das dateninstanzabhängige Gewicht w_i entfernt. Durch den Kernel und die pro Rechnenschritt zu bestimmenden Gewichte sowie der höherdimensionalen Projektion steigt der Berechnungsaufwand, womit sich dieses Verfahren nicht für große Datensätze bzw. hochdimensionale Datensätze eignet.

Es existieren des Weiteren Varianten von K-Means die unterschiedliche Metriken bzgl. der Optimierung, rekursive Implementierungen oder probabilistische Ansätze verfolgen [35, 69, 81, 90, 120].

Insgesamt sind partitionierende Verfahren in der Lage Cluster zu bilden, die sphärenförmig organisiert sind. Die Berechnungskomplexität dieser Kategorie ist jeweils individuell und reicht von $O(nkd)$ bis hin zu einer kubischen Berechnungs-Komplexität. In der Regel ist innerhalb des Verfahrens eine Distanzmetrik in Verwendung. Dieser Aspekt kann bei steigenden Dimensionen zu einer Minimierung der Zuverlässigkeit der Ergebnisse führen (siehe Kapitel 2.1.2). A-priori definierte Unsicherheiten (z.B. durch Modelle oder berechnete Werte) müssten in diesen Verfahren z.B. in der Entwicklung einer spezifischen Distanzmetrik einfließen, die Unsicherheiten bei deren Kalkulationen beachtet. Die Fuzzy K-Means Variante bietet die Möglichkeit die Unsicherheiten auch im Ergebnis in Form von den Gewichtungen abzubilden, soweit das im jeweiligen Anwendungsfall sinnvoll erscheint [118].

2.3.4 Hierarchische Verfahren

Die Entwicklung der hierarchischen Verfahren ist motiviert durch die Verbesserung der partitionierenden Verfahren, wie z.B. die Abschätzung des Parameters K. Grundsätzlich sind zwei Arten von hierarchische Verfahren zu unterscheiden.

- Die Verfahren im Bereich des agglomerativen Clustering verfolgen den Bottom-Up Ansatz. Jede einzelne Dateninstanz befindet sich zum Zeitpunkt der Initialisierung in einem individuellen Cluster. Während der Analyse fassen die Verfahren entsprechend ähnliche Dateninstanzen oder Gruppen von Dateninstanzen zusammen.
- Analog dazu arbeitet das teilende Clustering (divisive clustering), das aus einem voll umfänglichen Cluster kleinere Cluster bildet.

Beide Arten arbeiten rekursiv und die Ergebnisse können mittels Dendrogram manuell validiert werden. Abbildung 2.3 zeigt beispielhaft einen Da-

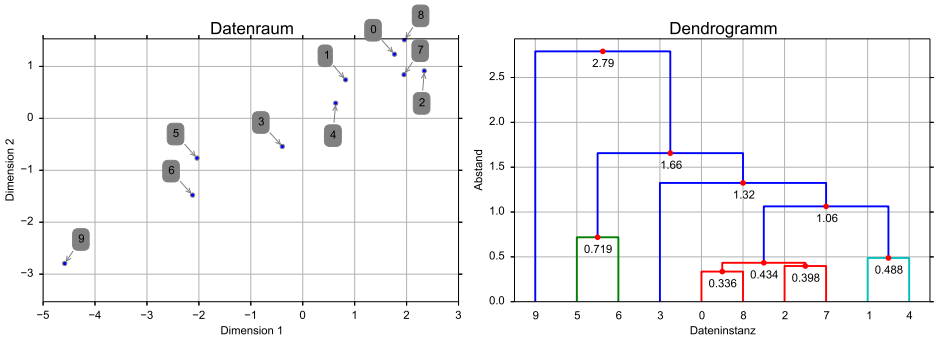


Abbildung 2.3: Visualisierung eines hierarchischen Clustering Vorhabens eines Datenraums und einer Ergebnisdarstellung mittels Dendrogram

tenraum sowie ein agglomeratives Clustering in visualisierter Form durch ein Dendrogram. Jedes Cluster ist im Dendrogram farblich markiert ,womit beispielsweise ein Cluster aus den roten Linien besteht, die die Datenpunkte 0,2,7,8 verbindet. Implizit verwenden alle hierarchischen Verfahren ein graphbasierten Ansatz.

Agglomerative Clustering

Die agglomerativen Clustering Verfahren unterscheiden sich in der Wahl der Fusionierungskriterien bzw. der Ähnlichkeitsmetrik. Prinzipiell verflechten agglomerative Verfahren genau die Cluster miteinander, die sich bzgl. einer Distanzmetrik am nächsten sind. Das Verfahren erfolgt iterativ und genau so lange, bis ein Konvergenzkriterium zutrifft (beispielsweise sobald sich alle Elemente in nur noch einem Cluster befinden). Ein weiteres Beispiel für ein Abbruchkriterium ist das Überschreiten eines Abstandsschwellwerts, der

während des Clusterings überschritten wird. Da ein Cluster durch seine einzelnen Dateninstanzen repräsentiert ist, selektiert das Fusionskriterium insgesamt je einen Repräsentanten aus den Clustern zum Vergleich mit einem anderen Cluster (Die Repräsentanten eines Clusters können pro verschiedenem zu vergleichendem Cluster variieren). Eine Distanz- bzw. Ähnlichkeitsmatrix der Dimension $|C|x|C|$ enthält die berechneten Werte aller Cluster zueinander bzgl. der Distanzmetrik. Die folgende Liste zeigt die bekanntesten Fusionierungskriterien, die in einem hierarchischen agglomerativen Clustering Einsatz finden.

- Das einfache Kopplungs Kriterium (single linkage criteria) selektiert Repräsentanten eines Clusters anhand des minimalen Abstands gegeben durch $D(C_a, C_b)$ aller möglichen Kombinationen der Repräsentanten beider Cluster:

$$D(C_a, C_b) = \forall c_i \in C_a, \forall c_j \in C_b : mind(c_i, c_j). \tag{2.13}$$

  Gleichung 2.13 sucht unter allen Clustern entsprechend das Cluster Paar mit dem minimalen Abstand zueinander. Da dieses Verfahren lokal orientiert ist, wird dieses auch als sog. Nearest Neighbor Verfahren bezeichnet.
- Gegensätzlich dazu verhält sich das vollständige Kopplungs Kriterium (complete linkage criteria). Dieses selektiert die Repräsentanten anhand des maximalen Abstands aller möglichen Kombinationen:

$$D(C_a, C_b) = \forall c_i \in C_a, \forall c_j \in C_b : maxd(c_i, c_j). \tag{2.14}$$

  Dieses Kriterium wird auch Furthest Neighbor Verfahren genannt aufgrund der o.g. abstandsmaximierenden Metrik.
- Mit Hilfe des durchschnittlichen Kopplungs Kriterium (average linkage criteria) berechnet ein Algorithmus den Abstand anhand des durchschnittlichen Abstands aller Elemente in einem Cluster:

$$D(C_a, C_b) = \frac{1}{|C_a||C_b|} \sum_{c_i \in C_a} \sum_{c_j \in C_b} d(c_i, c_j). \tag{2.15}$$

- Das durchschnittliche Gruppenkopplungs Kritierium (average group linkage criteria) berücksichtigt neben dem durchschnittlichen Kopplungskriterium die Abstände innerhalb eines Clusters. Dazu vereinigt das Kriterium temporär alle Elemente beider Cluster und berechnet den durchschnittlichen Abstand des neuen temporären Clusters, wie folgt darstellt.

$$D(C_a, C_b) = \frac{1}{(|C_a| + |C_b|) * (|C_a| + |C_b| - 1)} \sum_{c_i, c_j \in C_a \cup C_b, i \neq j} d(c_i, c_j) \tag{2.16}$$

- Alternativ zu den o.g. Verfahren die bestehende Dateninstanzen als Repräsentanten bzw. Berechnungsgrundlage wählen, nutzt das Centroid Kopplungs Kriterium Centroids aus dem im vorherigen Kapitel genannten partionierenden Verfahren. Die folgende Gleichung berechnet das Kriterium entsprechend anhand der zusätzlich zu bestimmenden Centroid m_a und m_b.

$$D(C_a, C_b) = d(m_a, m_b) \tag{2.17}$$

- Ein weiteres bekanntes Fusionskriterium ist Median Kopplungs Kriterium, das den medianen Mittelpunkt eines Clusters bestimmt und diesen als Repräsentanten selektiert.

Das Kapitel [78, S. 224] diskutiert die einzelnen Fusionskriterien im Detail und geht auf die jeweiligen Vor- und Nachteile ein. Insgesamt betrachtet sind agglomerative Verfahren von der jeweiligen Distanzmetrik abhängig und die Integration von Unsicherheiten bzw. die Möglichkeit der Analyse hochdimensionaler Datenräume ist von der Metrik zu gewährleisten. Die in Kapitel 2.1.2 diskutierten Punkte bzgl. der Nutzung von Distanzmetriken in hochdimensionalen Datenräumen sind bei einer Applikation des agglomerativen Clustering zu beachten. Des Weiteren besitzen agglomerative Verfahren i.d.R. trotz der Minimierung des theoretischen Suchraums von $O(2^n)$ eine hohe Berechnungskomplexität von mindestens $O(n^2 \log n)$ [43, S.80ff]. Die Komplexität der Speicherauslastung entspricht durch die benötigte Distanzmatrix $O(n^2)$ [121]. Die o.g. Verfahren und Kriterien sind bereits in einigen Analysebibliotheken bzw. Software vorhanden, beispielsweise in der kommerziellen Software STATA [134], MATLAB [132] oder in der Open Source Software SciPy [127].

Teilendes Clustering

Die bzgl. des agglomerativem Vorgehen gegensätzlichen Verfahren sind die divisiven Verfahren, die mittels Top-Down Strategie arbeiten. Beginnend mit einem einzigen Cluster das alle Dateninstanzen beinhaltet, teilen diese Art von Verfahren Cluster sukzessive rekursiv auf, bis ein entsprechendes Abbruchkriterium zutrifft. Gegenüber dem agglomerativen Verfahren besteht die Möglichkeit eine höhere Effizienz zu erreichen, da eine vollständige Berechnungsmatrix der Dimension NxN nicht berechnet wird, soweit das Abbruchkriterium frühzeitig zugreift. Zur Teilung des ersten einzelnen großen Clusters stehen theoretisch $2^{n-1} - 1$ Kombinationen zur Verfügung. Das Divisive Analysis (DIANA) Verfahren [78] nutzt einen heuristischen Ansatz zur Minimierung des Suchraums und funktioniert mit dem folgenden skizzierten Algorithmus.

1. DIANA sucht die Dateninstanz (c_i) deren durchschnittliche Unähnlichkeit d_{avg} (z.B. hoher Wert in der euklidischen Distanzmetrik) bzgl.

aller anderen Objekte maximal ist.

$$d_{avg,c_i}^{\{C_a\}} = \frac{1}{|C_a| - 1} \sum_{c_j \in C_a, i \neq j} d(c_i, c_j) \tag{2.18}$$

Gleichung 2.18 stellt die Berechnungsfunktion der durchschnittlichen Unähnlichkeit d_{avg} einer Dateninstanz c_i zu einem Clusters C_a dar. Das zum Cluster unpassende Objekt c_i bildet eine sog. Splinter Group, die den Anfang eines neuen Clusters repräsentiert, $C_a = C_a \setminus c_i, C_{splint} = \{c_i\}$.

2. DIANA prüft jedes Objekt, das sich nicht in der Splintergroup befindet auf eine entsprechende Integration.

$$\begin{aligned} \forall c_i \in C_a : d_i = & \frac{1}{|C_a| - 1} \sum_{c_j \in C_a, i \neq j} d(c_i, c_j) \\ & - \frac{1}{|C_{splint}| - 1} \sum_{c_j \in C_{splint}} d(c_i, c_j) \end{aligned} \tag{2.19}$$

Gleichung 2.19 zeigt das Vorgehen von DIANA zur Detektion eines weiteren Kandidaten für die Splinter Group. Die Distanz d_i der i-ten Dateninstanz eines Clusters wird untersucht.

3. Das maximale und positive d_i indiziert eine weitere Dateninstanz c_i, die DIANA zur Splinter Group hinzufügt.
4. Existieren nur noch negative Differenzen von durchschnittlichen Unähnlichkeiten, ist das Aufteilen des Clusters c_i abgeschlossen.
5. Schritte 1-4 werden für das Cluster mit dem maximalen Durchmesser (dem größten maximalen Abstand zweier Dateninstanzen) wiederholt.

Dieser Prozess läuft genau so lange, bis ein o.g. Abbruchkriterium zutrifft oder alle Cluster die Kardinalität eins besitzen. Die Berechnungskomplexität der aufteilenden hierarchischen Clustering-Algorithmen ist vergleichbar mit den agglomerativen Verfahren. Analog zu den o.g. Verfahren basiert das genannte Clustering-Verfahren auf einer euklidischen Distanzmetrik, womit die im vorherigen Abschnitt genannten offenen Punkte für hochdimensionale unsichere Daten ebenfalls zutreffen. Ebenfalls ist eine Integration von Unsicherheiten in den Analyseprozess in von einer entsprechenden Distanzmetrik ein weiterer offener Punkt.

CURE

Die in den vorherigen Abschnitten diskutiertern Verfahren erstellen aufgrund der Repräsentanten-Selektion und der radialen Orientierung kreis- bzw. kugelförmige Cluster. Existieren jedoch stark geometrisch verzerrte oder elliptische Cluster, können diese von den o.g. Verfahren nicht korrekt

erkannt werden bzw. diese werden in mehrere Cluster aufgeteilt [3, 121]. Das Clustering using Representatives (CURE) [55, 56] ist ein Ansatz um diese offenen Punkte zu verbessern. CURE arbeitet in den folgenden Schritten.

1. Durch die Organisation der Daten in einem k-d Baum existiert eine effiziente Zugriffsstruktur für die Anfragen. Der Baum wird mit allen Dateninstanzen befüllt. Eine randomisierte Selektion einer Untermenge aller Dateninstanzen ist der Basisdatensatz für die Erstellung aller Cluster. Dieser Schritt wird als Random Sampling bezeichnet.
2. In einem sog. Partitioning Schritt erstellt ein Algorithmus iterativ eine definierte Anzahl k an Clustern mit einem eigenen agglomerativen distanzbasierten Verfahren auf Basis der Untermenge. Der Algorithmus gruppiert iterativ die nächsten Nachbarn einer aktuell zu berechnenden Datensatz mit Hilfe der Baumstruktur.
3. Der restliche Anteil der Gesamtmenge an Dateninstanzen wird, mit Hilfe der nächsten Nachbarbeziehung, in die bereits ermittelten Cluster integriert (der sog. Disk Labeling Schritt).

Insgesamt betrachtet ist bei dem CURE Algorithmus eine distanzbasierte Metrik die Grundlage für die Berechnungen. Zusätzlich ist es erforderlich manuell und a-priori die Anzahl der zu erstellenden Cluster zu definieren. Die Komplexität des CURE Algorithmus liegt für hochdimensionale Daten bei $O(n^2_{Untermenge} \log n_{Untermenge} + n_{Rest})$ bzw. für $1 \leq d \leq 3$ $O(n^2_{Untermenge})$ [56]. Aufgrund der primären Datenstruktur in Form des k-d Baums besitzt das Verfahren eine Speicherkomplexität von $O(n)$.

CHAMELEON

Der agglomerativ und divisiv gestaltete CHAMELEON [76] Algorithmus konstruiert die Cluster mit Hilfe von dünnen partitionierten Graphen. Dieser Graph besteht aus Knoten (Dateninstanzen) und Kanten (Werte einer Ähnlichkeitsmetrik zwischen zwei Knoten). Deren Berechnung des Graphen erfolgt mit Hilfe des K-nearest-neighbor Verfahren (kNN Verfahren) [32]. CHAMELEON definiert zwei Metriken, die zur Erstellung der Cluster eingesetzt werden sowie einer vom jeweiligen Anwendungsfall abhängigen Ähnlichkeitsmetrik (z.B. euklidische Distanz). Das Verfahren geht davon aus, dass diese Metrik hohe Werte für unähnliche Cluster oder Dateninstanzen liefert. Chameleon führt das Konzept der Interkonnektivität sowie der relativen Nähe zweier Clustern ein.

$$RI_{\{C_i,C_j\}} = \frac{|EC_{\{C_i,C_j\}}|}{\frac{|EC_{C_i}|+|EC_{C_j}|}{2}} \tag{2.20}$$

Gleichung 2.20 beschreibt die sog. relative Interkonnektivität $RI_{\{C_i,C_j\}}$ auf Basis der folgend beschriebenen Schnittkante (Edge Cut).

Die absolute Interkonnektivität bzw. Schnittkante $|EC_{\{C_i,C_j\}}|$ ist definiert als die Summe der Kanten, die die zwei Cluster C_i und C_j verbinden. Diese Verbindungen zwischen den Clustern werden Schnittkanten genannt, da diese die Cluster unterscheiden bzw. verbinden. Neben dieser Inter-Cluster-Konnektivität existiert eine virtueller Cluster-interne Schnittkante EC_{C_i}, die lt. [76] ein Cluster in zwei Hälften teilt, wobei jeweils $\approx 50\%$ der Anzahl der Dateninstanzen in jedem Subgraph liegen.

Mit Hilfe der sog. relativen Nähe (relative Closeness) evaluiert CHAMELEON die Nähe zweier Cluster, definiert durch die Schnittkante relativ zu der Nähe der Punkte innerhalb eines Clusters. Dazu nutzt der Algorithmus eine Metrik $\bar{S}_x$, die den durchschnittlichen Wert der Schnittkanten in x angibt. Ist x ein einzelnes Cluster, z.B. C_i, so ist $\bar{S}_{C_i}$ der durchschnittliche Wert für Nähe bzw. der Kanten innerhalb eines Clusters. Analog zu einem einzelnen Cluster entspricht $\bar{S}_{\{C_i,C_j\}}$ der durchschnittlichen Nähe zweier Cluster, repräsentiert durch die Schnittkanten. Die relative Nähe ist definiert durch:

$$RC(C_i, C_j) = \frac{\bar{S}_{\{C_i,C_j\}}}{\frac{|C_i|}{|C_i|+|C_j|}\bar{S}_{C_i} + \frac{|C_j|}{|C_i|+|C_j|}\bar{S}_{C_j}}. \tag{2.21}$$

Die relative Nähe $RC(C_i, C_j)$ ist definiert als die Nähe zweier Cluster relativ zur relativen Nähe innerhalb eines Clusters.

CHAMELEON analysiert Datensätze innerhalb der folgenden zwei Phasen, da der alleinige Einsatz von Phase zwei nur genau dann funktioniert, wenn alle (Unter-)Cluster eine entsprechende Anzahl an Knoten und Kanten besitzen, um die o.g. Metriken berechnen zu können [32].

1. In Phase 1 sucht der Algorithmus mit Hilfe eines Graph Partitionsalgorithmus vom gegebenen kNN Graph initiale Unter-Cluster. Die Anzahl der Unter-Cluster ist mit m notiert. In einem divisiven iterativen Vorgehen ermittelt der Algorithmus das größte Cluster (anfangs existiert genau ein Cluster) und separiert die Unter-Cluster mit Hilfe des Partitionierungsalgorithmus (z.B. hMetis [77]). Während der Unterteilung minimiert der Partitionierungsalgorithmus die Anzahl der Schnittkanten, sodass sich mindestens 25% aller Dateninstanzen eines Clusters in einem Subcluster befinden. Der Prozess stoppt, sobald ein manuell definierter Schwellwert für die Anzahl der Dateninstanzen in einem Cluster ($minSize$) bei der Aufteilung eines Clusters erreicht ist.
2. In Phase 2 wird im Nachgang zum divisiven Verfahren aus der ersten Phase ein agglomeratives Verfahren angewandt, das den eigentlichen Hauptteil CHAMELEONs darstellt und die finalen Cluster anhand der o.g. Metriken erstellt. Ziel in dieser Phase ist es, die sehr kleinen Unter-Cluster zu größeren Cluster zu vereinigen, sobald diese bestimmte Kriterien erfüllen. Es existieren zwei verschiedene Kriterien, die über eine Verflechtung zweier Unter-Cluster entscheiden.

(a) Als erste Möglichkeit steht eine schwellwertbasierte Analyse zur Verfügung. Soweit die Kriterien der relativen Nähe und der relativen Interkonnektivität die Schwellwerte T_{RC}, T_{RI} überschreiten, verflechtet CHAMELEON die beiden Cluster.

$$RI(C_i, C_j) \geq T_{RI} \text{ und } RC(C_i, C_j) \geq T_{RC}$$

Im Falle einer multiplen Verflechtungsmöglichkeit verschiedener Cluster mit dem Cluster C_i selektiert der Algorithmus die Verflechtung, deren absolute Interkonnektivität maximal ist.

(b) Alternativ zu einem Schwellwert definiert das Verfahren, das folgend genannte Kriterium dessen maximaler Wert für eine Verflechtung entscheidend ist:

$$RI(C_i, C_j) * RC(C_i, C_j)^{\alpha}. \tag{2.22}$$

Der manuell selektierte Parameter α repräsentiert in diesem Kriterium die Möglichkeit den Einfluss der relativen Nähe Metrik zu variieren.

Insgesamt kann das Verfahren beliebig geformte Cluster erkennen und besitzt eine Berechnungskomplexität von $O(nm + n \log n + m^2 \log m)$ [76]. In der originalen Veröffentlichung vorgestellten Variante des Verfahrens basiert dieses auf einer Distanzmetrik, weshalb diese Variante sich im Fall der euklidischen Distanz nicht für hochdimensionale und unsichere Daten eignet (siehe Kapitel 2.1.2). Dennoch besteht die Möglichkeit, dass dieses Verfahren auf unsicheren und hochdimensionalen angewendet werden kann, soweit eine Ähnlichkeitsmetrik definiert wird, die die o.g. Eigenschaft erfüllt sowie eine repräsentative Ähnlichkeit zweier Cluster wiederspiegeln kann.

Des Weiteren ist der mit Hilfe des kNN Verfahrens erstellte Graph durch die euklidische Distanzmetrik aufgebaut. Diese Metrik ist ebenfalls zu ersetzen beim Einsatz von hochdimensionalen und unsicheren Daten. Dieses Verfahren benötigt eine a-priori Definition der Schwellwerte bzw. der Parameters α und $minSize$.

2.3.5 Gridbasierte Verfahren

Die Gridbasierten Clustering Verfahren analysieren die Daten mit Hilfe eines Gitters, das über den Datenraum gelegt wird. Dadurch ist es möglich, den Datenraum zu unterteilen und so die Anzahl der Zugriffe auf einzelne Dateninstanzen und damit auch die Berechnungskomplexität zu minimieren. Die Diskretisierung des Datenraums resultiert in Zellen, die entsprechende Dateninstanzen beinhalten. Im Allgemeinen können sowohl geometrische als auch statistische sowie dichtenbasierte bzw. eine Kombination der Eigenschaften für das Clustering genutzt werden.

In der Literatur sind verschiedene Analysetypen auf das Clustering mittels Gittern gegeben. Es existieren diverse Verfahren, die sich etabliert haben und sich primär in der Art und Weise unterscheiden, wie die Gitter (i.d.R. wird mehr als ein Gitter pro Verfahren erstellt) analysiert werden. Die Analysetypen sind in den folgenden Kategorien zu unterscheiden.

Adaptive Gitteranalyse

Die Adaptive Gitteranalyse ist z.B. im Adaptive Mesh Refinement Verfahren (AMR) [86] gegeben. AMR arbeitet mit nicht uniformen Gittergrößen: Das Verfahren verfeinert einzelne Zellen im Gitter, solange die Dichte in der Zelle einen Schwellwert überschreitet. Die Datenorganisation in AMR erfolgt mittels Baumstruktur, sodass eine Verfeinerung einer neuen Ebene in der Hierarchie entspricht. AMR erstellt die Cluster beginnend in der Blattebene unter Zuweisung von Grids zu einem Cluster anhand der Dichte der Grids. AMR besitzt die Komplexität $O(dn\frac{1-p^h}{1-p} + r\frac{1-q^h}{1-q}(dk + 6^d))$. h entspricht der Höhe der Baumstruktur, p der durchschnittlichen Anzahl an Punkten die in jeder Baumebene verfeinert werden, r ist die initiale Gittergröße sowie q das durchschnittliche Verhältnis der Gittergrößen zweier aufeinander folgenden Ebenen in der Baumstruktur.

Axenverschiebende Verfahren

Axen-Verschiebende Verfahren analysieren und verschieben Gitter um maximal eine Zellenbreite. Damit ist es möglich, lokale Variationen einzelner Cluster durch z.B. einem relativen großen Abstand zweier Datenpunkte bzgl. des Gitters auszugleichen. Der Axis-Shifted Grid-Clustering (ASGC) Algorithmus [24, 29] prozessiert Daten durch multiple Gittererstellung und der zusätzlichen einfachen Verschiebung des Gitters. ASGC erstellt zunächst ein Gitter, dessen Dichte pro Zelle analysiert wird. Das Gitter ist für alle d Dimensionen jeweils in k Teile äquidistant aufgeteilt. Schwellwertbasiert entscheidet ASGC, ob die Dichte in einer Zelle signifikant ist (insgesamt k^d Prüfungen), um mit benachbarten signifikant dichten Zellen ein Cluster zu bilden. Anschließend verschiebt ASGC das Gitter in jeder Dimension um einen Wert v und wiederholt die Analyse zur Bildung der Cluster. Überlappende Cluster aus beiden Ergebnissen (mit und ohne Axenverschiebung) werden vereinigt. Die Berechnungskomplexität von ASGC beträgt $O(k^d+n)$.

Der New Shifting Grid Clustering Algorithmus (NSGC) [91] arbeitet vergleichbar zum ASGC Algorithmus, jedoch verschiebt NSGC die Datenpunkte anstatt die Zellen. Dadurch vereinigt NSGC keine Cluster aus zwei verschiedenen Clustering-Ergebnissen, sondern gibt Datenpunkten einen definierte vergrößerten Raum bei der Dichtenanalyse. Dies resultiert in einer Komplexität von $O((2w)^d)$, wobei w die Anzahl der Iterationen darstellt.

Gitteranalyse von hochdimensionalen Daten

Verfahren zur Analyse hochdimensionaler Daten funktionieren i.d.R. durch Analyse von Gittern kombiniert mit einer individuellen Analyse des Gitters. Der Clustering in Quest (CLIQUE) Algorithmus [5] ist sowohl ein dichtebasiertes als auch ein gitterbasiertes Verfahren, das durch eine Suche in Unterräumen des Datenraums Cluster analysiert. Die Kernidee in CLIQUE ist zurerst ein Cluster in einem Unterraum zu finden. Anschließend versucht CLIQUE den Unterraum entsprechend zu modifizieren oder erweitern, um weitere Cluster dem gemeinsamen Unterraum zu erschließen. Prinzipiell arbeitet CLIQUE zur Darstellung von Clusters mit disjunktiven Normalformen (DNF). Diese beschreiben virtuelle Ebenen in einem Unterraum, der ein Cluster eingrenzt. Ein Cluster in CLIQUE wird durch benachbarte dicht besiedelte Zellen repräsentiert. Die Berechnungskomplexität ist linear abhängig von der Anzahl der Dateninstanzen, jedoch quadratisch bzgl. der Dimensionen [122].

Es existieren Varianten des CLIQUE Verfahrens. Ein Beispiel einer Variante ist der Merging of Adaptive Finite Intervals (MAFIA) Algorithmus [52, 99]. Ebenso wie in AMR spannt der MAFIA Algorithmus ein Gitter auf. Die Verfeinerung des Gitters ist in MAFIA gesteuert durch die Verteilung der Daten in den jeweiligen Dimensionen auf Basis eines Histograms, das das Verfahren als ersten Schritt erstellt. Anhand des Gitters bildet MAFIA sog. Dichteeinheit (Dense Unit), die ein Objekt in verschiedenen Dimension geometrisch betrachtet rechteckig umfassen. Eine Dichteeinheit ist genau dann gegeben, wenn die Dateninstanzen in der Zelle eine definierten zellgrößenabhängen dynamischen Schwellwert überschreiten. Ein Cluster entspricht daher einem Verbund zusammenhängender Dichteeinheiten. MAFIA besitzt eine Berechnungskomplexität von $O(c^d + \frac{n}{b} d\gamma)$, wobei c eine nicht weiter definierte Konstante darstellt und γ die durchschnittliche I/O Zugriffszeit repräsentiert. Ein weiteres Beispiel einer Variante des CLIQUE Verfahren ist das Entropy Clustering (ENCLUS) Verfahren [26], das die Entropie einer Zelle anstatt deren Dichte analysiert.

Das optimal Grid-Partitioning Verfahren (OptiGrid) [67, 71] erstellt im Gegensatz zu den o.g. Verfahren auch irreguläre Gitter. Während die o.g. Methoden pro Dimension sich nur in einer Äquidistanz im Gitter oder einer lokalen bzw. globalen Adaption bzw. Verschiebung unterscheiden, sucht OptiGrid zur Unterteilung des Datenraums sog. irreguläre optimale Schnitthyperebenen (Cutting Hyperplanes) ($\mathbb{R}^d$). Das Verfahren analysiert den Datenraum mit Hilfe einer Kernel-Dichte Funktion ϕ, die ein Cluster erkennt, sobald die Funktionen einen Schwellwert ξ überschreitet. Weniger dichte Bereiche ($\phi < \xi$) werden als Rauschen interpretiert.

Diese irregulären (nicht axen-parallele) arbiträren Schnittebenen unterteilen den Datenraum, um die Cluster zu teilen. Schnitthyperebenen $\mathbb{R}^{d-1}$ existieren nur in den Bereichen, die ein lokales Minimum der jeweiligen

Dichtefunktion im Unterraum besitzen. In den weniger dichten Bereichen etabliert OptiGrid die Schnitthyperebenen zur Trennung der Cluster. Insgesamt besitzt das Verfahren eine Komplexität $O(nd \log n)$.

Alle genannten Verfahren die Gitteranalyse einsetzen besitzen die Möglichkeit Rauschen durch schwellwertbasierte Analysen zu verwalten. Eine explizite Integration zur Analyse allgemeiner Unsicherheiten ist nicht gegeben, könnte jedoch im Falle von OptiGrid u.U. in die Dichtefunktion in Form eines geeigneten Kernels integriert werden.

2.3.6 Fraktale Verfahren

Zum Zeitpunkt der Erstellung dieser Arbeit existiert ein weiterer Ansatz, das sog. Fractal Clustering (FC) [11, 12]. Der Ansatz berechnet Ähnlichkeiten anhand der Fraktalen Dimension FM_q, die von einem Parameter q abhängig ist. [11, 12] beschreibt den Parameter $q = 0$ als Haussdorff Dimension, $q = 1$ als Informationsdimension und $q = 2$ als Korrelationsdimension. Die folgende Gleichung zeigt die Berechnung der lt. [12] Fraktalen Dimension FM_q.

$$FM_q = \begin{cases} \frac{\partial \log \sum_i p_i \log p_i}{\partial \log g} & \text{for } q = 1 \\ \frac{1}{q-1} \frac{\partial \log \sum_i p_i^q}{\partial \log g} & \text{otherwise} \end{cases} \tag{2.23}$$

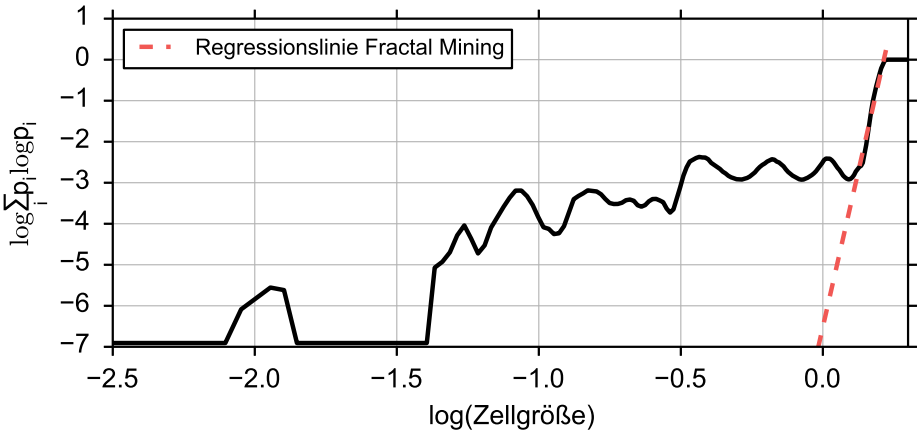


Abbildung 2.4: Visuelle Darstellung der FM-Regressionslinie. Die Abszisse stellt die Zellgrößen, die Ordinate die dazu gehörigen Werte für den Zähler des FM_q Wertes, dar.

Voraussetzung für die Berechnung ist ein bereits berechnetes d dimensionales Gitter mit der Seitenlänge g und einer Variable i, die eine Zellennummer darstellt. Eine Funktion p berechnet eine nicht weiter definierte Frequenz, die prinzipiell angibt, wie viele Datenpunkte relativ sich in der jeweilige Zelle befinden relativ zur Gesamtanzahl der Datenpunkte.

Werte für FM_q stellen die q-abhängige analytische oder approximierte Lösung für die Steigung eines quasi-linearen Teils der approximierten Funktion dar. Abbildung 2.4 zeigt beispielhaft die Herausforderung in der Berechnung des Wertes FM_q. Die Daten der zur Erzeugung von Abbildung 2.4 verwendet wurden, entsprechen den in Kapitel 5.2.1 beschriebenen Daten für ein bestimmtes Cluster. Die rote und gestrichelt dargestellte Regressionslinie ist an genau einer Zellgröße für das Gitter definiert. Das heisst, für jedes Cluster und Boxgröße existiert durch die Ableitung, die nicht a-priori analytisch bestimmt werden kann, genau ein Wert FM_q. Sowohl die Berechnung als auch die Unterscheidung der Cluster anhand der Ableitung aus Gleichung 2.23 ist aufwändig. Da sich die Steigungen in der überwiegenden Anzahl der Punkte unterscheiden, ist kein eindeutiges Ergebnis für FM_q festzustellen. Beispielsweise würde sich eine Regressionslinie im Punkt $\log(Zellgroesse) = -1.9$ gravierend von der dargestellten Regressionslinie unterscheiden. Dies impliziert, dass multiple Lösungen für entsprechende Cluster vorliegen können.

Mit diesem Ansatz besteht die Möglichkeit die Änderung der fraktalen Dimensionen durch Hinzufügen einzelne Dateninstanzen zu evaluieren. Dabei gilt eine minimale Änderung als beste Möglichkeit einer Zuweisung einer Dateninstanz zu einem Cluster. Jedoch benötigt dieser Ansatz bereits vorgefertigte Cluster durch einen sog. Initialisierungsschritt. Zum Zeitpunkt der Erstellung dieser Arbeit existiert keine Veröffentlichung, die einen realen Datensatz mit FM prozessiert.

2.3.7 Zusammenfassung

Es existiert eine Vielzahl an Clustering Algorithmen die für ein verschiedene Szenarien eingesetzt werden können. Mit Hinblick auf die o.g. Kategorien bzw. deren Repräsentanten beinhalten die Beschreibungen jeweils mindestens eine der folgenden offenen Fragen:

1. Unabhängigkeit zu einer distanzbasierten Metrik bzw. eine Ähnlichkeitsmetrik für hochdimensionale Datenräume ist vorhanden
2. Integration von Unsicherheiten in den Analyseprozess oder in der Ähnlichkeitsmetrik
3. Lineare oder akzeptabel superlineare Berechnungskomplexität bzgl. der Dimensionen und der Anzahl der Dateninstanzen

Die dichtebasierenden, partitionierenden sowie hierarchischen Verfahren besitzen zwar teilweise eine bzgl. Punkt 3 akzeptable Komplexität, jedoch zum aktuellen Zeitpunkt weder eine entsprechende hochdimensionale Ähnlichkeitsmetrik, noch eine integrierte Methodik zur Behandlung von Unsicherheiten. Gridbasierte Verfahren bieten die Möglichkeiten Unsicherheiten in Form von Rauschen durch eine Diskretisierung zu subtrahieren, jedoch existiert aktuell keine explizite Methodik z.B. zur Verwendung von a-priori

bestimmten Varianzen bzgl. einzelner Messwerte. Das dargestellte Fraktale Verfahren bietet einen neuen Ansatz, jedoch integriert dieser ebenfalls keine Unsicherheiten. Darüber hinaus ist die Berechnungskomplexität beeinflusst durch die notwendige Optimierung der Suche nach der korrekten Regressionsline für das jeweilige Cluster. Probabilistische Verfahren sind theoretisch in der Lage beliebige Datensätze sowie Cluster zu erlernen jedoch i.d.R. mit einer hohen Berechnungskomplexität sowie einer manuellen Definition eines entsprechenden Modells.

Kapitel 3

Datenvorverarbeitung

Die Vorverarbeitung von Daten zum Zweck eines späteren Clustering Vorhabens ist ein notwendiger Schritt, um ein Clustering überhaupt zu ermöglichen. Roh-Daten aus beispielsweise wissenschaftlichen Experimenten oder Observationen sind in der Regel nicht geeignet, um eine Datenanalyse zu betreiben. Diese enthalten beispielsweise folgende Eigenschaften, die bei eine Datenanalyse komplexer gestalten:

- Kalibrationsfehler gegeben durch Messtechnologien und deren Kalibrationsverfahren
- Duplikate in den Dateninstanzen oder Dimensionen durch z.B. Korrelationen
- Nicht normalisierte Wertebereiche
- Rauschen

Das Ziel bei der Generierung der Daten ist diese im originalen unverfälschten Zustand abzuspeichern und erst in einem späteren Schritt bzgl. der jeweiligen und unterschiedlichen gewünschten Analyse zu optimieren. Die Art und Weise der Vorprozessierung und Vorbereitung hängt von der individuellen Anwendungsdomäne, dem Datensatz und eventuellen Optimierungsschritten ab. Die Vorverarbeitung von hochdimensionalen Daten für den Fall des Clusterings beinhaltet beispielsweise:

- Eine Datenkomprimierung in effiziente Datenformate oder Konsolidierung der Daten von verschiedenen Quellen
- Die Selektion von Eigenschaften (Features) (Feature Selection z.B. in [88]) zur Ausgrenzung irrelevanter Eigenschaften, die die Qualität des Clusterings beeinflussen können bzw. zur Sicherung der Vergleichbarkeit mehrerer Instanzen
- Das Erstellen neuer Features auf Basis mehrerer bestehender Eigenschaften (Feature Extraction z.B. [59]) zur Dimensionsreduktion und damit zur Verkleinerung des Suchraums
- Die Detektion von fehlerhaften Datensätze bzw. Ausreißern

- Normalisierung der Datenwerte pro Dimension über alle Instanzen hinweg oder auch zusätzlich über alle Dimensionen hinweg, falls die einzelnen Dimensionen unabhängige Messungen aber gleiche Messwerttypen (z.B. Messung von Durchflussmenge von Licht einer bestimmten Wellenlänge) darstellen
- Kalibrierung der Messdaten auf Basis der Messtechnologie (z.B. Dark-, Bias-, Flat-Field Frame Kalibrierung bei CCD Kameras)

Die Vorverarbeitung von Datensätzen ist ein eigener Forschungsbereich und in der Literatur wird z.B. in [7, 48, 72, 98] unter anderem diese Thematik behandelt.

Diese Thesis konzentriert sich auf die für diese Arbeit relevanten Bereiche der Vorverarbeitung. Zuerst diskutiert dieses Kapitel den in dieser Thesis genutzten Datensatz und dessen Eigenschaften bzgl. des Clusteringvorhabens. Anschließend werden die theoretischen Überlegungen sowie die Implementierung der Vorverabeitungsschritte auf Basis des Hadoop Rahmenwerks skizziert .

3.1 Beschreibung des Datensatzes

Die in dieser Thesis verwendeten Daten zur Demonstration der Funktionalität des Algorithmus stammen vom Sloan Digital Sky Survey (SDSS) [136] Projekt. Dieses Gemeinschaftsprojekt von Instituten aus fünf verschiedenen Nationen wurde von der Alfred P. Sloan Foundation initiiert. Die Erstellung einer dreidimensionalen weitreichenden Karte zur Darstellung der großskalierenden Strukturen des Universums [112] stellt das Projektziel dar. Die Daten die Wissenschaftler primär aus Observationen gewinnen, sind auf der Projektwebseite öffentlich verfügbar.

Diese Arbeit verwendet Daten aus dem Data Release 10 (DR10) [138]. Das durch die teilnehmenden Institute finanzierte 2.5 m Weitwinkel-Teleskop observiert im Projektverlauf definierte Bereiche des Himmels zur Untersuchung von Galaxien, Sternen und Quasaren. Das Teleskop beinhaltet zwei unterschiedliche Arten von Instrumenten, die Daten generieren [57].

- Das $5x6$ Array von CCD-Kameras zur Aufnahme von Bilddaten dienen, ist umgeben von weiteren CCD-Kameras. Insgesamt sind die 54 CCD-Kameras teilweise unterschiedlich dimensioniert ($2048x2048$ Pixel bzw. $2048x400$ Pixel). Die Bilddaten werden in dieser Arbeit nicht verwendet und werden daher auch nicht genauer diskutiert. Insgesamt würde der Algorithmus auch die Verarbeitung von Bilddaten mit einem adaptierten hier nicht diskutierten Vorverarbeitungsverfahren unterstützen.
- Zwei Spektrographen, die mit 640 bzw. 1000 Glasfasern bestückt sind. Insgesamt wird mit den Spektrographen pro Glasfaser eine Abdeckung bzgl. der aufgenommenen Wellenlängen zwischen $3800 - 9200\,\text{Å}$ er-

reicht. Jeder Spektrographen ist mit 50% der Fasern bestückt, die innerhalb des Spektrographen nach einer komplexen Optik in CCD-Kameras enden. Insgesamt existieren 4 CCD-Kameras (Jeweils eine „Blaue“ Kamera für Wellenlängen $< 6000\,\text{Å}$ und eine „Rote“ Kamera für Wellenlängen $> 6000\text{Å}$). Ein vor die Kameras platziertes Gitterprisma teilt die Teilchen der jeweiligen Wellenlängen für die Kameras entsprechend auf. Alle Fasern sind mit einem Spektrographen und einer Platte verbunden, die für jede Observation speziell angefertigt wird. Diese besitzt einzelnen Bohrungen, deren Position auf der Platte so organisiert ist, dass eine die jeweilige Faser auf ein bestimmtes zu observierendes Objekt ausgerichtet ist. Mit Hilfe einer wechselbaren Kassette, die aus der benannten Platte und Glasfasern hin zu einer Faserschnittstelle besteht, können zeit effizient neue Platten in das System integriert werden. Die speziell angefertigten Kassetten für verschiedene Platten ermöglichen eine schnelle neue Zuordnung von Fasern zum Plattenloch.

3.1.1 Datenerfassung der Spektrographen

Zunächst werden vor der Datenerfassung mit Hilfe der Bilddaten des bildgebenden (ersten) Instruments die Objekte bzw. deren Positionen definiert, die mit dem Spektrograph näher untersucht werden. Mittels der Positionen wird eine entsprechende Platte für die Glasfasern angefertigt, die im Anschluss in eine neue Kassette inklusive der Glasfasern implementiert wird. Die Positionen der Objekte bzw. deren dedizierte Fasern mit deren Zuordnung zu einer Glasfaser werden auf einem entfernten Kontrollrechner des Instruments vor der Observation bzw. bei der Anfertigung der Kassette gespeichert.

Die Verbindung des Kontrollrechners mit dem Instrument erfolgt über eine Versa Module Eurocard (VME) Verbindung, die bis zu $400\,MByte/s$ im seriellen Modus übertragen kann. Mit Hilfe von Standardnetzwerkprotokollen kann das Instrument über Remote Procedure Calls (RPC) angesteuert werden. Werden die CCD-Kameras zur Datenerfassung aktiviert, speichert ein Single-Board Computer vom Typ Motorola MVME167 die Daten auf einer SCSI-Festplatte lokal innerhalb des Instruments zwischen. Der Kontrollcomputer kann diese Rohdaten im Anschluss über die VME-Verbindung abrufen.

Nach einer Aufnahmesitzung werden alle Rohdaten zur zentralen Speicherung und Kalibrierung an das Fermi-Lab [133] übertragen. Eine typische Observation mit 640 Fasern liefert ca. $1.5\,GByte$ an Daten pro Nacht. Dabei wird jede Platte bzw. jedes Objekt drei bis fünf mal mit je 15 Minuten Aufnahmedauer observiert. Die Anzahl der Aufnahmen (sog. Science Exposures) ist abhängig von der jeweiligen Wettersituation. Im Anschluss an jede Science Exposure folgen weitere Aufnahmen zur Kalibration im Kontext der CCD-Kameras [112]. Diese beinhalten die Illumination der Kameras mit drei

verschiedenen Projektoren, bestückt mit:

- Einer 3100K Quarz-Jod Lampe für Flat-Fields Kalibrierung aller Kameras
- Eine Quecksilber-Cadmium Lampe für die Kalibrierung Kameras im zur Aufnahme im Bereich < 6000Å
- Eine Neon-Lampe für die Kalibrierung Kameras im Bereich > 6000Å

Darüber hinaus wird je CCD-Kamera eine Bias-Aufnahme (Aufnahme bei abgedunkelter Kamera) angefertigt und eine sog. Smear-Aufnahme (Aufnahme des Himmels während sich das Teleskop bewegt. Dabei überstreicht jede Glasfaser eine Fläche von $5.5x9\,arcsec$).

3.1.2 Kalibrierung der Messungen

Wie in [112] beschrieben, werden die im Spektrographen erzeugten Daten direkt nach der Observation zwei Kalibrierungen unterzogen:

- Kalibrierung der Aufnahmen bzgl. der Flussdichte (Flux), die sog. spektrophotometrische Kalibierung
- Kalibrierung der Aufnahmen bzgl. der Wellenlängen (mit Hilfe der Flat-Field Aufnahmen)

Die spektrophotometrische Kalibrierung beruht auf der Einstellung der Verbindung zwischen den Ausgabesignalen der CCD-Kamera und der tatsächlichen physikalischen Flussdichte des Lichts. Nach der Observation werden die Rohdaten mit zwei Prozessen nachbearbeitet bzw. kalibriert [114].

Der sog. **spectro2d** Prozess [114] reduziert die Daten der beiden Kameras pro Spektrograph durch die Verknüpfung der Daten aus den Kameras, der Kalibrierung der Rohdaten mit Hilfe der Zusatzaufnahmen. Alle Flussdichten sind in der Einheit $10^{-17} ergs\, cm^{-2} s^{-1} \text{Å}^{-1}$ gespeichert. Die Kalibrierungssoftware bildet neben dem eingestellten Flux-Messwert pro Wellenlänge auch weitere Daten zur Verfügung [114]. Unter anderem wird pro Flux-Messwert eine inverse Varianz ($1/\sigma^2$) angegeben, die bei einer zu geringen Signal-to-Noise Ratio (SNR) auf 0 gesetzt wird. Deren Berechnungsprozess sowie eine Beschreibung der **spectro2d**-Pipeline ist in [114] verfügbar. Eine detaillierte Beschreibung der Berechnungen sowie der Pipeline ist jedoch nur im öffentlich verfügbaren Code [137] einsehbar.

3.2 Der Clustering Datensatz

Ob ein Datensatz überhaupt durch ein bestimmtes Clustering-Verfahren analysiert werden kann, ist abhängig vom jeweiligen Verfahren bzw. auch von der jeweiligen Anwendungsdomäne. Beispielsweise funktioniert ein Verfahren mit einer distanzbasierten Metrik (wie z.B. einer Minkowski-Distanz,

siehe Gleichung 3.1) für Datensätze mit wenigen Eigenschaften. Mit steigender Anzahl von Eigenschaften (und damit auch die Anzahl der Ausreißer bzw. der irrelevanten Eigenschaften) sinkt die Repräsentationsfähigkeit der Metrik. Dies wirkt sich im Extremfall dahingehend aus, dass die Distanz zweier nächster bzw. die Distanz zweier entfernter Elemente gleich groß ist [14].

$$D_{ij} = (\sum_{k=1}^{d}(|x_{ik} - x_{jk}|)^q)^{(1/q)} \tag{3.1}$$

Die Minkowski Distanz mit der Distanz D zwischen den Objekten i und j auf Basis der absoluten Differenz zwischen der k-ten Eigenschaft von i und j zeigt Gleichung 3.1.

Die Daten aus den SDSS Observationen sind für Clustering-Anwendungen als ein geeigneter und herausfordernder Datensatz anzusehen, da mehrere in Kombination für eine Analyse komplexe Eigenschaften vereint sind:

1. Ein großes Volumen des Datensatzes bzw. die steigende Anzahl zur Verfügung stehenden Objekte (durch die weitergeführten Observationen). Ein Clustering-Verfahren muss bzgl. der Berechnungskomplexität skalieren. Zum Zeitpunkt der Erstellung dieser Thesis stehen im DR10 ca. $3 \cdot 10^6$ Spektren zum Clustering zur Verfügung.
2. Eine große Anzahl unabhängiger Eigenschaften pro Objekt (bis zu 5100 sind über alle Spektren überlappend). Die Unabhängigkeit der Eigenschaften ergibt sich aus der Unabhängigkeit der Messwerte zweier Wellenlängen eines Spektrums. Diese Anzahl der Dimensionen wirkt sich in Clustering-Verfahren auf die Laufzeit (durch z.B. die Berechnungskomplexität einer Distanzmetrik) aus.
3. Eine durch Kalibration ermittelte Unsicherheit in Form einer inversen Varianz pro Dimension (Wellenlänge) pro Objekt (Spektrum) ist zusätzlich gegeben und kann die Qualität des Clusterings beeinflussen.

Die auf der Projektwebseite [138] zur Verfügung gestellten Daten sind im FITS Dateiformat [130] gespeichert. Alle bei der Observation erzeugten Daten eines Objekts zu einem Observerationszeitpunkt sind in einer Datei abgelegt. Dabei repräsentieren die Dateien jeweils eine Aufnahmesitzung einer Glaserfaser an einem Observationsdatum (Modified Julian Date, MJD). Für eine Analyse stehen somit maximal $3 \cdot 10^6$ Dateien mit einer durchschnittlichen Größe von ca. $500\,KByte$ zur Verfügung.

Wie bereits beschrieben, werden sowohl Flux Daten als auch die inversen Varianzen zur Analyse zur Verfügung gestellt. Abbildung 3.1 zeigt beispielhaft ein nicht vorprozessiertes Spektrum nach der astronomischen Kalibrierung durch die o.g. spectro2d-Pipeline mit den Bias, Flat-Field und Smear-Aufnahmen inklusive der kalkulierten inversen Varianzen ($1/\sigma^2$) aus dem SDSS Datensatz.

Die Messwerte der Spektren liegen im Bereich $0 <= flux <= 70 *$

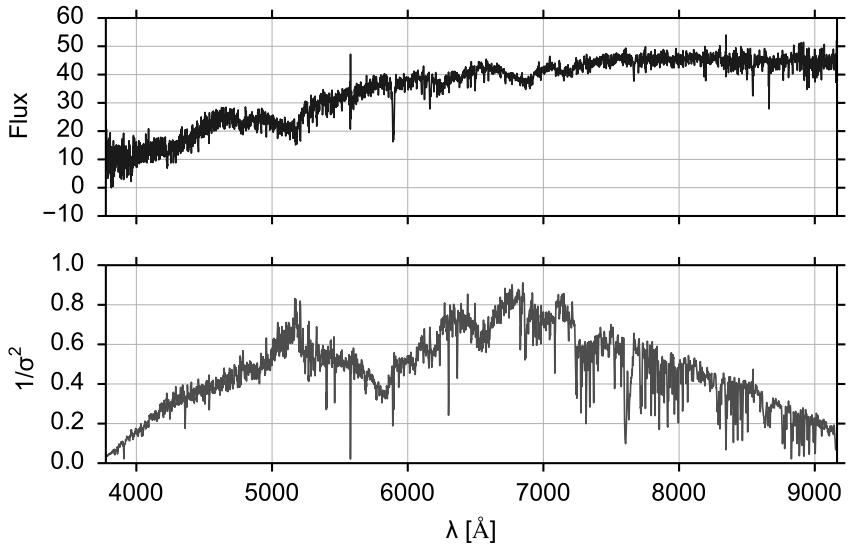


Abbildung 3.1: Spektrum aus Platte 0269, MJD 51581, Faser 0432: Flux-Messwerte und inverse Varianzen. Auf der Ordinate sind im oberen Graph die Flux-Messwerte, im unteren Graph die inversen Varianzen aufgetragen. Die Abszisse beschreibt jeweils die dazugehörige Wellenlänge in Ångström.

$10^{-17} ergs\ cm^{-2} s^{-1} \text{Å}^{-1}$. Die inversen Varianzen befinden sich in einem Wertebereich zwischen 0 und theoretisch ∞. Dabei ist ein Flux-Messwert der mit einer berechneten inversen Varianz $(1/\sigma^2)$ von 0 gemessen wurde, als unsicher bzw. unzuverlässig einzustufen. Diese Einstufung ist qualitativ einem fehlenden Messwert gleichzusetzen. Im Beispielspektrum sind die ersten 100 Werte als weniger zuverlässig anzusehen. Die Vorverarbeitungsroutine muss sicherstellen, dass sich alle Spektren im gleichen Flux-Wertebereich befinden.

Im Vergleich mit anderen Spektren hat jedes Spektrum eine individuelle Form bzw. einen eigenen Verlauf über alle Dimensionen hinweg. Abstrakt betrachtet ist in einem Clustering Verfahren das Ziel die Spektren in einem Cluster zu gruppieren, die einen ähnlichen Verlauf über alle Messwerte hinweg haben unter der Berücksichtigung der inversen Varianzen. Zum Beispiel kann eine Ähnlichkeit die optische Überlagerung der zu untersuchenden Spektren überprüft werden. Abbildung 3.2 stellt diese Variante dieser manuellen Ähnlichkeitsuntersuchung dar.

Beide Spektren in Abbildung 3.2 besitzen einen ähnlichen Verlauf, jedoch sind diese im Universium unterschiedlich weit von der Erde entfernt, so dass eine Rotverschiebung für das weiter entfernte Objekt (Spektrum 2, grau dargestellt) entsteht. Die Vorverarbeitungsroutine muss daher sicherstellen, dass alle Rotverschiebungen ausgeglichen sind, um die Vergleichbarkeit pro

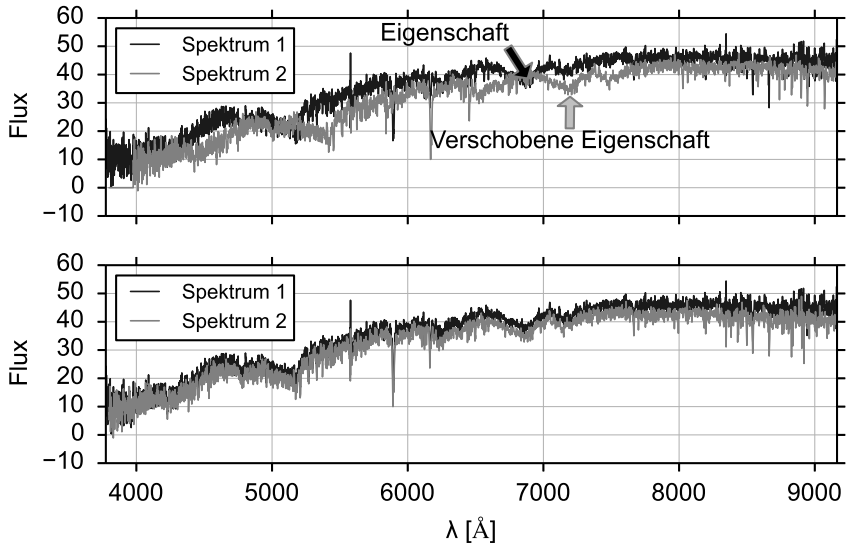


Abbildung 3.2: Ähnliche aber rotverschobene Spektren

Dimension zwischen den Spektren herzustellen. Die pro Spektrum vorhanden und bestimmten Rotverschiebungskonstanten werden hierfür eingesetzt. Abbildung 3.2 zeigt beispielhaft im oberen Graphen das (rot)verschobene Spektrum im Vergleich zu einem ähnlichen nicht rotverschobenen Spektrum, im unteren Graphen die ausgeglichene Rotverschiebung von Spektrum 2.

Die Vorverarbeitungsroutine muss zusätzlich durch teilweise extreme Rotverschiebungen einen Wellenlängenbereich über alle Spektren finden, in dem in allen Spektren Messdaten vorhanden sind. Zusammengefasst muss die Vorverarbeitungsroutine sicherstellen, dass die diversen Spektren zueinander vergleichbar sind. Diese Voraussetzung erreicht ein entsprechendes Verfahren im Fall der SDSS Daten durch die Einhaltung eines definierten Wertebereichs, der Korrektur der in den Spektrendaten vorhandenen Rotverschiebungen sowie der Wahl gleicher Wellenlängen zum Vergleich der Spektren.

3.3 Hadoop

Diese Arbeit verwendet zur Vereinfachung der Entwicklung der Vorprozessierungsroutinen das Hadoop-Rahmenwerk (Framework). Hadoop ist ein skalierbares Rahmenwerk, das eine verteilte Umgebung bereitstellt um große Datensätze mit Hilfe eines verteilten Dateisystems und einer verteilten, sich selbst organisierenden Laufzeitumgebung zu prozessieren. Das Rahmenwerk besteht aus den Implementierungen des von Google entwickelten Programmiermodells MapReduce sowie dem dazu gehörigen verteilen Dateisystem

Hadoop-Distributed-File-System (HDFS).

Der zentrale nichtfunktionale Kernbestandteil von Hadoop ist die inhärente Fehlertoleranz des Systems, die einen Betrieb auf sog. Commodity-Hardware erlaubt. Die Fehlertoleranz zeigt sich, z.B. in HDFS, durch die mehrfache Verteilung von Datenblöcken, die über die Server des HDFS verteilt werden. Mit Hilfe von verschiedenen Parametern lassen sich Replikaanzahl, Blockgröße sowie die physische Lokation der Server bzgl. der Racks definieren.

Diese Parameterisierung erlaubt Hadoop die Datenorganisation so zu gestalten, dass die möglichen Ausfallszenarien (z.B. Serverausfall, Rackausfall, etc.) keine Schäden an den Daten hervorrufen können. HDFS ist prinzipiell mittels Master-Slave Prinzip organisiert, bei dem ein sog. Namensknoten (Name Node) die Metadaten des Clusters (Zuordnungen von Datenblöcken zu Datenknoten) beinhaltet. Die sog. Datenknoten (Data Nodes) speichern jeweils die einzelnen Datenblöcke. Weitere Informationen zum HDFS System wie z.B. die Organisation der klientenseitig initiierten Schreib- bzw. Lesevorgänge sind in [19, 119] detailliert diskutiert.

Die folgenden beiden Unterkapitel skizzieren kurz das Programmiermodell MapReduce und die Weiterentwicklung Hadoop 2.0.

3.3.1 MapReduce

Bei der Datenverarbeitung großer Datenmengen eigenen sich zur effizienten Bearbeitung Verfahren die, soweit möglich, die Daten parallel bzw. nebenläufig verarbeiten. Da eine nebenläufige Prozessierung von Daten eine generelle Problemstellung ist, umfasst das MapReduce Paradigma einen möglichen, allgemeinen Lösungsansatz. Eine vollautomatisierte Organisation der Aufgabenstellungen bzw. zu prozessierenden Daten auf Basis weniger Parameter sowie der aktuellen Auslastung erfolgt in sog. Arbeitsaufträge (Jobs). Hadoop unterteilt diese Arbeitsaufträge wiederum in Einzelaufgaben (sog. Tasks), die eine atomare Einheit zur Lastverteilung über die Knoten hinweg bilden.

Durch eine Master-Slave Architektur in Hadoop orchestriert ein Master in Form des Auftragsüberwachers (sog. Job-Tracker) die Arbeitsaufträge und Einzelaufgaben. Ein Einzelaufgabenüberwacher (sog. Task-Tracker) bearbeitet die Einzelaufgaben im First-In-First-Out (FIFO) Prinzip. Neben der Abarbeitung der Aufgaben stellen die überwachenden Komponenten sicher, dass auch in einer Hardware- oder Rahmenwerksbasierten Ausfallsituation alle Aufgaben prozessiert werden. Dabei setzt der Einzelaufgabenüberwacher abgebrochene Einzelaufgaben, aufgrund der Atomarität, vor der erneuerten Bearbeitung in den initialen Zustand zurück. Grundsätzlich besteht das Programmiermodell aus den zwei Funktionen *map* und *reduce*, deren Ausführung zur Laufzeit jeweils eine Einzelaufgabe darstellen.

map-Funktion

Während die Implementierung der *map*-Funktion die eigentliche Datenverarbeitung übernimmt, organisiert die *reduce*-Funktion ggfs. die Zusammenfassung der verteilt verarbeiteten Daten.

$$map : (\text{Schlüssel}_{Eingabe}, Wert_{Eingabe}) \rightarrow ((\text{Schlüssel}_{temp}, Wert_{temp}))^* \quad (3.2)$$

Die *map*-Funktion ist definiert in Gleichung 3.2 und beschreibt eine allgemeine Abbildung eines initialen Schlüssel/Wert Tupels aus dem Eingabevorgang auf eine Liste temporärer Schlüssel/Wert Tupel. Dabei entspricht die Notation $(x)^*$ einer Liste von verschiedenen Elementen x. Durch die Definition dieser aufteilenden Operation können Entwickler beschreiben, wie Daten für die spätere Zusammenfassung zu gruppieren sind. Die *map*-Funktion dient darüber hinaus zur eigentlichen (Vor-)Verarbeitung der Daten und stellt damit den zentralen Schritt des Ansatzes dar.

Die *map*-Funktion erhält initiale Schlüssel/Wert Paare, die aus den Eingabefunktionalitäten des Rahmenwerk stammen . Je nach benutzerdefinitierter Konfiguration der Aufträge unterscheidet sich die Art und Weise der Eingabedatenypen. Liest Hadoop beispielsweise textbasierte Dateien ein, entspricht der Schlüssel ($key_{Eingabe}$) dem Dateinamen und der Wert ($value_{Eingabe}$) der jeweiligen eingelesenen Zeile der Textdatei, jeweils vom Datentyp *String*. Angemerkt sei, dass Hadoop aufgrund der Nebenläufigkeit keine Abarbeitungsreihenfolge, der pro Datei sequentiell eingelesenen Schlüssel/Wert Paare, garantiert.

Der Aufbau der *map*-Funktion in Kombination mit der Möglichkeit der Implementierung der Eingabemethodik zeigt, dass diese beiden Komponenten ausschlaggebend sind für die Definition der Nebenläufigkeit der jeweiligen Aufgabenstellung. Je feingranularer die Tupel aus der Eingabemethodik orchestriert sind, desto mehr Einzelaufgaben kann Hadoop für die Abarbeitung der *map*-Funktionsinstanzen generieren, was u.U. die Bearbeitungsdauer reduzieren kann, da durch die Adaption der Anzahl der Einzelaufgaben auch die maximal mögliche Nebenläufigkeit beeinflusst wird. Die Bearbeitungsdauer ist stark abhängig vom Verhältnis der durchschnittlichen Bearbeitungdauer einer Einzelaufgabe und der Bearbeitungsdauer des Rahmenwerks zur Organisation bzw. Verteilung der Aufgaben. Bei einem ungünstigen Verhältnis ist der zeitliche Verwaltungsaufwand höher, als der zeitliche Aufwand zur Prozessierung der einzelnen Map-Funktion. Dies kommt i.d.R. bei vielen kleinen Dateien vor, weshalb in diesen Fällen die Eingabekomponenten durch ein aggregierendes Äquivalent auszutauschen sind, die viele einzelnen Dateien zu einer Gruppe zusammenfassen. Eine Einzelaufgabe prozessiert demnach mehrere kleinere Dateien, was wiederum das o.g. Laufzeitverhältnis optimiert.

reduce-Funktion

Die aus Gleichung 3.2 entstandenen temporären Tupel aggregiert die *reduce*-Funktion.

$$reduce : (\text{Schlüssel}_{temp}, (Wert_{temp})^*) \rightarrow (\text{Schlüssel}_{final}, Wert_{final})^* \quad (3.3)$$

Gleichung 3.3 zeigt die Beschreibung der *reduce*-Funktion, die aus einem temporären Schlüssel und dessen Liste an Werten $(Wert_{temp})^*$ eine neue, aber finale Liste an Schlüssel/Wert Paaren definiert. Dabei entspricht die Struktur der finalen Schlüssel/Wert Tupel der Struktur der Ausgabe. In der Regel entspricht Schlüssel_{final} der atomaren Einheit, die von den benutzerdefinierten oder von Hadoop bereitgestellten Ausgabefunktionalitäten (der sog. OutputFormat und OutputWriter) persistiert werden und damit das Ergebnis der Analyse bereitstellen.

Prinzipiell ist es nicht für jeden Anwendungsfall erforderlich, nach einer *map*-Phase die einzelnen temporären Werte zu aggregieren, sodass es auch möglich ist die *reduce*-Funktion ausschließlich nur als Wrapper für die Ausgabe zu verwenden. Darüber hinaus ist die Semantik dieser Funktion so definiert, dass entweder ein oder kein Wertepaar als Resultat zugelassen ist [34].

Die Resultatstupel der *map*-Funktion steuern die Anzahl der maximal möglichen nebenläufig zu bearbeitenden Instanzen der *reduce*-Funktion. Neben dieser Anzahl organisiert der Auftragsüberwacher die Anzahl nebenläufigen Bearbeitungsprozesse anhand der aktuellen Auslastung.

Die erste Instanz der *reduce*-Funktion kann genau dann mit der Bearbeitung starten, sobald alle *map*-Einzelaufgaben prozessiert sind. Angemerkt sei, dass in den aktuellen Implementierungen von Hadoop in der Statusanzeige die Reducer-Einzelaufgaben starten, während *map*-Einzelaufgaben aktiv prozessiert werden. Da die Reduzierungsoperation alle Werte eines Schlüssels benötigt, besteht der erste Schritt der Operation in einer randomisierten Verteilung der Zwischenergebnisse auf verschiedene Knoten. Nach dieser Verteilung sortiert das Rahmenwerk die Schlüssel/Wert-Tupel, sodass alle Werte eines Schlüssels sich entsprechend in einer Liste befinden, um anschließend von der *reduce*-Funktion prozessiert zu werden. Während die randomisierte Verteilung parallel zur Verarbeitung der *map*-Einzelarbeitsschritte erfolgen kann, müssen diese für den anschließenden Sortiervorgang vollständig abgeschlossen sein.

Beispiel

Es sei das fiktive Beispiel der Quantifizierung der Vorkommnisse einzelner Wörter in verschiedenen Texten gegeben. Jeder Text sei in einer einzelnen einfachen Textdatei ohne spezifisches Format gespeichert. Um diese Art von Daten bearbeiten zu können, sind zwei Punkte zu prüfen. Es sind sowohl der

Eingabe- als auch der Ausgabemechnismus zu definieren. Zur vereinfachten Darstellung kann in diesem Fall jeweils die vom Rahmenwerk bereitgestellten Funktionalitäten zur Ein- und Ausgabe von Texten, in Form des InputReaders und RecordWriters, genutzt werden. Nach dem Start des Auftrags ermittelt das Rahmenwerk die Anzahl der zu lesenden Dateien auf Basis des definierten Eingabepfads und initiiert m Lesevorgänge auf verschiedenen Knoten, wobei m der Anzahl der Dateien entspricht. In der Standardimplementierung versucht Hadoop die Dateien mit einem Leseverfahren für Textdateien einzulesen, wobei dieses $key_{Eingabe}$ als den Dateinamen und $Wert_{Eingabe}$ als die einzelnen Zeilen einer Textdatei definiert und für die *map*-Einzelaufgaben auf den lokalen Knoten bereitstellt.

map ist in diesem Beispiel so zu definieren, dass zuerst Satzzeichen entfernt und anschließend die Zeile anhand der Leerzeichen aufgeteilt wird. Die daraus resultierenden Wörter (key_{temp}) liefert *map* jeweils mit dem Wert 1 zurück. Falls ein Wort in einer Zeile mehrfach vorkommen sollte, liefert diese *map*-Funktion mehrfach das Tupel $(Wort, 1)$ zurück.

Die *reduce*-Funktion erhält die durch das Rahmenwerk mehrere schlüsselweise sortierte Schlüssel-Liste Paare $(key_{temp}, (Wert_{temp})^*)$ und aggregiert diese durch Iteration der Liste mit gleichzeitigem Aufsummieren der Werte innerhalb der Liste. Im konkreten Beispiel summiert die Funktion das temporäre Tupel durch:

$$reduce : (Wort_x, (Wert_1, Wert2, \ldots, Wert_n) \rightarrow (Wort_x, \sum_{i=1}^{n} Wert_i)) \tag{3.4}$$

Dabei entspricht n der Anzahl der von der *map* Funktion gelieferten Werte und $Wert_i = 1$. Eine detaillierte Diskussion über die MapReduce Implementierung kann in [33, 34, 39] nachgelesen werden.

3.3.2 Hadoop 2.0 (YARN)

Hadoop ist das de-facto Standard Werkzeug zur Prozessierung von Big Data Szenarien sowohl in der Industrie als auch in den Daten getriebenen Wissenschaften [39, 83, 107]. Eine Weiterentwicklung und Erweiterung des in den vorherigen Abschnitten diskutierten Funktionalitäten des Hadoop 1.0 Rahmenwerks ist der Yet Another Ressource Negotiator (YARN). Dieser ermöglicht es unter anderem, Aufgaben unabhängig vom MapReduce Paradigmas umzusetzen. Die erste Version von des Hadoop Rahmenwerks wurde primär zur Prozessierung von Daten mittels MapReduce-Programmiermodell entwickelt. Mit der verbreiteten Nutzung zeigten sich folgenden Eigenschafen in der Architektur von Hadoop 1.0 als kritische Punkte [117]:

1. Durch die enge Kopplung des Datenprozessierungsparadigmas mit dem Infrastruktur-Rahmenwerk sind Anwender gezwungen, alle Arbeitsschritte kompatibel zum Paradigma zu gestalten.

2. Die Skalierbarkeit des Gesamtsystems ist durch die zentralisierte Jobsteuerung eingeschränkt.
3. Ausfälle bei Wartungsaufgaben umfassen das vollständige Cluster. Eine Aktualisierung der Hadoop Software ist daher nur mit Ausfallzeiten des Clusters umzusetzen.
4. Eine Partitionierung des Clusters zur parallelen Prozessierung mehrerer Jobs ist nur mit Hilfe der Definition der Anzahl von Map- und Reduce-Slots pro Knoten bzw. pro Job möglich.

Mit YARN wurde die im Jobtracker integrierten Funktionalitäten zur Infrastruktur bzw. Ressourcenverwaltung separiert und in die in Abbildung 3.3 dargestellte Architektur überführt. Zusätzlich wurde eine fundamentale Änderung der Datenstruktur des Job eingeführt. Mit dieser Änderung besteht ein Job nicht nur aus der reinen Implementierung des Map-Reduce Algorithmus, sondern beinhaltet auch der abstrakten Job-Konfiguration und den ggf. vorhandenen E/A-Implementierungen von speziellen Datentypen/Datenformaten oder Datenbanken. Trotz der Änderungen besteht die Abwärtskompatibilität der Prozessierung von Standard Map-Reduce Jobs ohne YARN Implementierung in einem YARN Cluster.

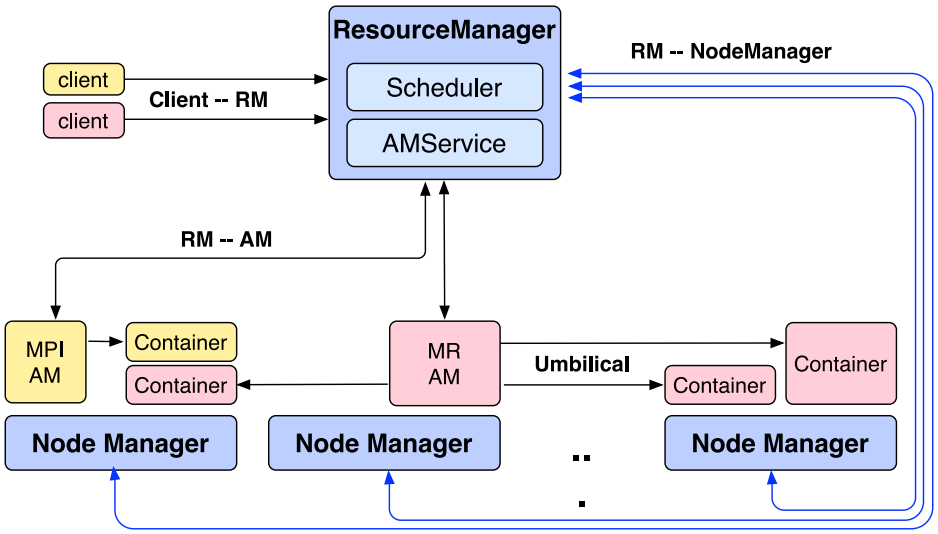


Abbildung 3.3: Hadoop YARN Architektur, entnommen aus [117]

YARN-Architektur besteht aus den folgenden Komponenten:

- Der **Ressource Manager (RM)** stellt die sowohl zentrale primäre Web-Schnittstelle zur internen Cluster-Kommunikation als auch eine Schnittstelle zur Kommunikation der Klienten mit dem Cluster zur Verfügung. Der RM überwacht alle Ressourcen und Aktivitäten des Clusters und vergibt die Cluster-Ressourcen auf entsprechende An-

fragen, was der Aufgabenstellung eines reinen Cluster-Schedulers entspricht. Das heisst, weiterführende Eigenschaften wie z.B. Fehlertoleranz gegenüber Ressourcenausfällen müssen von anderen Komponenten übernommen werden. Mittels Zookeeper können mehrere Instanzen des RM Dienstes in einer active/passive Konfiguration hochverfügbar betrieben werden. Im Falle eines neu zu startenden Jobs (Job Submission) greift ein Klient auf die Webschnittstelle des RM zu und fragt ein entsprechendes Ressourcengerüst an (z.B. Anzahl der virtuellen Cores, Arbeitsspeicher, Anzahl der parallel zu betreibenden Containern). Der RM verteilt den Auftrag innerhalb des Clusters unter Einbezug der Node Manager.

- Die pro Clusterknoten installierten **Node Manager (NM)** informieren den RM im Heartbeat Verfahren über den jeweiligen Node Status und die Ressourcen-Nutzung der jeweils aktuell aktiven Containern auf dem Knoten. Zusätzlich implementiert der NM die Rechteverwaltung aus den in der Container-Konfiguration definierten Access-Control-Lists (ACLs).
- Mit Implementierungen der **Application Master (AM)** Komponente können Entwickler rahmenwerkspezifische Bibliotheken erstellen, um den Kommunikationspfad zwischen Containern und zentralen Komponenten, wie dem Ressource Managern zu definieren. Beispielsweise können optimierte AM-Bibliotheken zur Prozessierung von Graphen oder Message-Passing-Interface (MPI) basierten Simulationen bzw. Analysen erstellt werden. Der AM verhandelt die Ressourcenanforderungen der Container mit dem RM (Ressource Request) und startet bzw. überwacht die Container mit Hilfe der NM. Die Separierung der Ressourcenverwaltungsaufgaben in AM,RM und NM löst den offenen Punkt 2 aus der o.g. Liste.
- Jobs in YARN werden in sog. **Containern** verwaltet. Durch die Kapselung der Datenverarbeitungsimplementierungen in isolierte Container wird innerhalb des geteilten und konsolidierten Clusters Mandantenfähigkeit realisiert. Das Konzept dieser Anwendungsvirtualisierung ist bekannt aus weiteren Lösungen, wie z.B. Docker [128]. Seit Version 2.6 unterstützt YARN auch den Betrieb von nativen Docker-Containern [94].

Neben den Änderungen in der Datenprozessierung wurde das HDFS verbessert. Dieses bietet nun die Möglichkeit der Erstellung von Snapshots zur Datensicherung.

3.4 Map-Reduce Job zur Datenvorverarbeitung

Die Verarbeitung vieler einzelner kleiner unabhängiger Daten ist eine stark nebenläufige Aufgabenstellung. Daher wurde die Datenvorverarbeitung mit

dem Hadoop 2.0 Rahmenwerk realisiert. Folgende Komponenten wurden im Rahmen dieser Arbeit entwickelt:

- Ein Map-Reduce Job mit einer integrierten Pipe-and-Filter Architektur,
- ein Hadoop InputFileFormat für FITS Dateien und
- ein komprimiertes binäres Dateiformat, um Schreib/Lesevorgänge effizient und platzsparend zu ermöglichen.

Zusammengefasst sind die in Kapitel 3.1 dargestellten Voraussetzungen an die Vorverarbeitungsroutine zu erfüllen:

- Die Vergleichbarkeit einzelner Dimensionen durch Kompensation der Rotverschiebung
- Sicherstellung eines definierten Wertebereichs für die Flux-Werte
- Ausschneiden eines spektren-übergreifenden Wellenlängenbereichs

Die folgenden Abschnitte beschreiben die Architektur des Map-Reduce Jobs sowie die Implementierung der einzelnen Filter.

3.4.1 Architektur

Die Vorverarbeitung von Spektren ist eine Gesamtaufgabe, die sich durch einzelne kleinere Teilaufgaben lösen lässt. Jede einzelne der im vorherigen Abschnitt genannten Teilaufgaben kann prozesstechnisch und organisatorisch isoliert und unabhängig von der anderen ausgeführt werden. Jedoch sollte, um zu einem korrekten finalen Spektrum zu kommen, eine gewisse Reihenfolge in der Abarbeitung eingehalten werden. Eine Pipe-and-Filter Architektur kann exakt den Umfang der Aufgabenstellung abdecken.

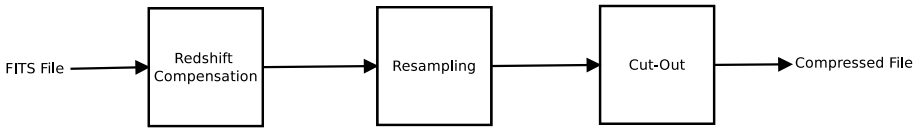


Abbildung 3.4: Logische Ansicht der Pipe and Filter Architektur

Abbildung 3.4 zeigt den schematischen Aufbau der in dieser Arbeit entwickelten Pipe-and-Filter Architektur. Diese ermöglicht es, unabhängige Elemente/Filter (dargestellt durch die Rechtecke) zu entwickeln, deren Reihenfolge im Prozess leicht verändert werden kann. Zusätzlich existiert die Möglichkeit durch die Isolation der Filter eine testgetriebene Entwicklung zu betreiben. Einzelne Filter können separat getestet werden und ermöglichen die Wiederverwendbarkeit in anderen Prozessketten. Die Eingabe bzw. Ausgabedatentypen von Filtern sind i.d.R. identisch, womit die sog. Pipes die Ausgabedaten eines Filters als Eingabedaten des darauffolgenden Filters umsetzen. Mit einer statischen oder dynamischen Konfiguration der Filter

zur Laufzeit können die Filter entsprechend aneinander gereiht werden und ergeben eine Prozesskette.

Die SDSS FITS Dateien enthalten wie in Kapitel 3.1 beschrieben ist, eine große Anzahl von Informationen, die zum Observationszeitpunkt gewonnen wurde. Zusätzlich zu den eigentlichen Daten der Spektren, wie Flux und inverse Varianz, existieren weitere Metainformationen wie z.B. die Rotverschiebung eines einzelnen Spektrums, die spezifische Eigenschaften und Parameter der Observation oder weitere Eigenschaften des Objekts beschreiben. Für die Clustering Analyse wird nur ein Teil der Informationen benötigt.

Zur effizienten Speicherung der zur Analyse benötigen Daten und damit zur Verkürzung der Einlesezeit für den Algorithmus bzw. zur Komprimierung der notwendigen Daten, wurde ein eigenes einfaches Dateiformat entwickelt. Nach der Vorverarbeitung speichert Hadoop die prozessierten Spektren in der in Abbildung 3.5 dargestellen Dateistruktur ab. Ziel bei der

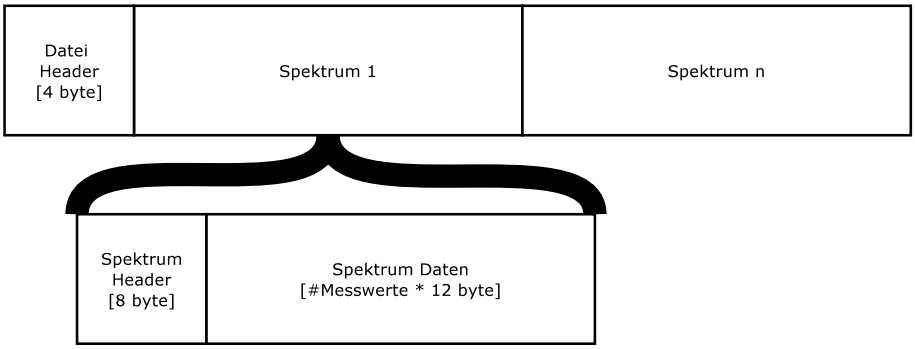


Abbildung 3.5: Ausgabedateiformat

Konzeption der Dateistruktur ist die Speicherung einer beliebigen Anzahl von Spektren mit den folgenden Eigenschaften.

- Flux-Messwert pro Wellenlänge
- Inverse Varianz pro Wellenlange
- den Wert der interpolierten Wellenlänge
- die Identifikationsnummer der Glasfaser (Fiber-ID)

Die Dateistruktur ist folgendermaßen aufgebaut. Das obere Segment von Abbildung 3.5 zeigt die grobe Dateistruktur. Diese bsteht aus einem 4 Byte großen Datei-Header, der die Anzahl der in dieser Datei gespeicherten Spektren notiert. Im Anschluss folgen die Spektren, die aus den folgenden Komponenten bestehen:

- Der 8 Byte Spektrum Header umfasst zwei Zahlen. Zuerst die Fiber-ID gefolgt von der Anzahl der gespeicherten Spektraldaten.
- Die sequentiell organisierten Spektraldaten beinhalten in folgender

Reihenfolge: Flux-Messwert, Inverse Varianz, die dazu berechnete Wellenlänge

Die Rohdaten sind so organisiert, dass pro Kombination von Platte, Glasfaser, MJD eine eigene Datei existiert. Durch die Organisation des MapReduce Jobgs befinden sich nach der Vorprozessierung alle Spektren einer Platte in einer Ausgabedatei. Dadurch wird die Anzahl der Ausgabedateien gegenüber den Rohdaten um den Faktor der Anzahl der Glasfasern verringert. Von Vorteil sind die entstehenden größeren Dateien, die bei der Analyse einen erhöhten Datendurchsatz bei dem nun sequentiellen Schreibvorgang während des Vorprozessierens sowie beim Einlesevorgang in der Analysesoftware erreichen. Zusätzlich wird die Anzahl der im Map-Reduce Job zu startenden Tasks und damit der Verwaltungsoverhead minimiert. Es wird dadurch eine zeitlich performantere Prozessierung ermöglicht.

3.4.2 Filter

Die nächsten Unterkapitel skizzieren die implementierten Filter, die zur Prozessierung eingesetzt werden.

Rotverschiebungs-Kompensation

Um die in den vorherigen Abschnitten diskutierten gemessenen Rotverschiebungen zu kompensieren, werden zwei Aktionen an jedem Spektrum vorgenommen, das rot verschoben ist. Zum einen wird die Wellenlänge anhand der gemessenen Rotverschiebung korrigiert und zum anderen muss die damit erneuerte Abtastung auf die Messwerte angewandt werden, um ein identisches vergleichbares Gitter an Wellenlängen über alle Spektren hinweg garantieren zu können. Dieser Resampling genannte Vorgang ist im nächsten Unterkapitel beschrieben. Die SDSS Daten beinhalten einen kalkulierten Rotverschiebungsfaktor z, den dieser Filter verwendet.

Generell beschreibt eine Rotverschiebung elektromagnetischer Wellen die Verlängerung der gemessenen Wellenlängen gegenüber der tatsächlichen emittierten Wellenlänge λ_{emit} der Strahlungsquelle unter Einhaltung der eigentlichen Intensitäten. Das Rotverschiebungsfaktor z beschreibt die Abweichung in

$$z = \frac{\Delta\lambda}{\lambda_{emit}} = \frac{\lambda_{gemessen} - \lambda_{emit}}{\lambda_{emit}} = \frac{\lambda_{gemessen}}{\lambda_{emit}} - 1. \tag{3.5}$$

Die dadurch berechnete Rotverschiebung stellt ausschließlich eine Änderung der Wellenlängen dar. Die gemessenen Photonen bzw. Intensitäten einer Wellenlänge bleiben von der Rotverschiebung unberührt. Eine Kompensation der Rotverschiebung pro Spektrum ist unter Kenntnis von z folgendermaßen durchzuführen:

$$\lambda_{emit} = \frac{\lambda_{spek}}{1 + z_{spek}}. \tag{3.6}$$

Diese Kompensation wird pro Spektrum pro Messwert ausgeführt. Durch die diversen Rotverschiebungsfaktoren entstehen für unterschiedliche Spektren unterschiedliche Bereiche an Wellenlängen, die observiert wurden. Wegen dieser Verschiebung der Wellenlängen müssen auch die entsprechenden Flux-Messwerte sowie die inversen Varianzen bzgl. der neuen Wellenlänge adaptiert werden, was vom Resampling Filter übernommen wird.

Resampling

Das Resampling der Spektren verfolgt zwei grundlegende Aufgaben. Die bei den Observationen gewonnen Messungen, beinhalten nicht die exakt gleichen Wellenlängen. Unterschiede zwischen den einzelnen Spektren sind möglich und sind unabhängig von deren Rotverschiebung. Durch Anwendung des Filters zur Kompensation der Rotverschiebung bzw. durch die individuellen Rotverschiebungen der observierten Objekte sind die Wellenlängen über alle Spektren hinweg betrachtet in einer heterogenen Konstellation. Zur Etablierung einer Vergleichbarkeit der Spektren ist eine homogener Menge an Wellenlängen über alle Spektren hinweg notwendig.

Wie wir in [82] beschreiben wird das folgende globale Wellenlängengitter definiert:

$$log(\lambda(p)) = 0.0001 \times p + 3.5222 \tag{3.7}$$

λ ist die Wellenlänge in Å an einer Position bzw. einer Dimension p im Datensatz eines Spektrums, wobei $0 \leq p < 5\,100$. Die Wellenlängen und die Messwerte der Spektren müssen angepasst werden. Λ ist die Sammlung aller Wellenlängen aus dem o.g. generierten finalen Gitter, λ_{git} die entsprechenden Wellenlängen des Gitters. λ_{git-gt} entspricht einer Wellenlänge auf dem definierten Gitter, die größer ist als eine gemessene Wellenlänge λ_{emit}. Analog dazu entspricht λ_{git-lt} einer kleineren Wellenlänge als λ_{emit}.

Für jedes adaptierte Spektrum muss daher gelten: Für alle λ_{emit} sind die beiden umschließenden λ_{git} zu finden, sodass $\lambda_{git-lt} < \lambda_{emit} < \lambda_{git-gt}$ gilt. Die Flux-Messwerte werden anteilsmäßig aufgeteilt, womit folgende Gleichungen gelten:

$$\delta_0 = |\lambda_{git-lt} - \lambda_{emit}|, \tag{3.8}$$

$$\delta_1 = |\lambda_{git-gt} - \lambda_{emit}|, \tag{3.9}$$

$$\mu_0 = 1 - \frac{\delta_0}{\delta_0 + \delta_1}, \tag{3.10}$$

$$\mu_1 = 1 - \frac{\delta_1}{\delta_0 + \delta_1} \tag{3.11}$$

Unter der Annahme, dass $flux_{emit}$ den Messwert an der Wellenlänge λ_{emit} und $flux_{git}$ den Messwert bei λ_{git} darstellt, werden die Messwerte folgendermaßen angepasst. Die Menge $\Lambda_{\lambda_{emit}}$ beinhaltet die Messwerte der um die Rotverschiebung korrigierten Wellenlängen $\lambda_{emit} : \lambda_{git-lt} < \lambda_{emit} < \lambda_{git-gt}$:

$$flux_{git-lt} = \sum_{flux_i \in \Lambda_{\lambda_{emit}}} \mu_0 * flux_i, \tag{3.12}$$

$$flux_{git-gt} = \sum_{flux_i \in \Lambda_{\lambda_{emit}}} \mu_1 * flux_i. \tag{3.13}$$

Diese Gleichungen summieren anteilig einen Messwert an der gemessenen Wellenlänge λ_{emit} auf den Messwert $flux_{git-lt}$ sowie $flux_{git-gt}$. Den jeweiligen Anteil bestimmen die o.g. Faktoren μ_0, μ_1. Dadurch beinflusst ein Messwert an der Stelle λ_{emit} sowohl den vorherigen als auch den nachfolgenden Messwert im definierten Gitter.

Die inversen Varianzen $ivar_{emit}$ der Wellenlänge λ_{emit} müssen ebenfalls an das Gitter angepasst werden. Um alle schlechten Messwerte auch im vorprozessierten Datensatz detektieren zu können, wird die inverse Varianz rechnerisch nicht modifiziert. Stattdessen verfolgt die Methodik den Ansatz der pessimistischen Wahl der nächsten Inversen Varianz. Ein Messwert des Gitters an einer Stelle λ_{git} wird von den umgebenden Messwerten an den Stellen $\lambda_{emit-lt} < \lambda_{git} < \lambda_{emit-gt}$ beinflusst. Die Gleichung

$$ivar_{git} = min(ivar_{emit-lt}, ivar_{emit-gt}) \tag{3.14}$$

definiert die Selektion der minimalen inversen Varianz bzgl. des Messwertes. Die Gleichung nutzt die minimale inverse Varianz, da Gleichung 3.13 die Flux-Messwerte aufteilt und ein aufgeteilter schlechter Messwert (kleinere inverse Varianz) in der Clustering Analyse berücksichtigt werden muss. Die Alternative der Selektion der größeren inversen Varianz ist möglich, würde aber eine eventuell schlechte Messung maskieren.

Der Resampling-Filter kann unabhängig von den anderen Filtern ausgeführt werden. In der Prozesskette ist dessen Integration jedoch erst nach der Korrektur der Rotverschiebungen sinnvoll, da andernfalls wieder heterogene Mengen an Wellenlängen bestehen. Allgemein kann nur durch das Resampling oder eine vergleiche adaptive Methodik eine Vergleichbarkeit der Dimensionen verschiedener Spektren für den SDSS Datensatz erreicht werden.

Min/Max

Der optionale Min/Max Filter dient zur Herstellung einer absoluten Homogenität des Flux-Wertebereichs unter Erhaltung der relativen Verhältnisse der Messwerte innerhalb eines Spektrums. Eine Skalierung der einzelnen

Werte in ein definierten Wertebereich $[A, B], B > A$ ermöglicht diese lineare Normalisierung. $flux_{\lambda-norm}$ stellt den normierten Flux-Wert, $flux_{\lambda}$ den nicht normierten Flux-Wert, $max(flux)$ das Maximum aller Flux Werte eines Spektrums sowie $min(flux)$ analog dazu das Minimum dar.

$$flux_{\lambda-norm} = \frac{flux_{\lambda} - min(flux)}{max(flux) - min(flux)} \times (B - A) + A \tag{3.15}$$

Die Applikation einer Min/Max Normierung auf die SDSS Spektren ist möglich, jedoch besitzen die Spektren vereinzelt extreme Spitzen, die um teilweise zweistellige Faktoren vom Durchschnittswert des Spektrums abweichen. Diese Eigenschaft erweist sich als Nachteil während der Analyse, da die Spektren, die z.B. eine (evtl. sogar qualitativ schlechte) Spitze im Spektrum haben, nicht mehr als ähnlich zu vergleichbaren Spektren ohne Spitze identifiziert werden können. Diese Intraspektrum Normierung ist daher nicht weiter in den Vorverarbeitungsprozess integriert.

Eine theoretische Alternative dazu stellt die Interspektrum Normierung dar, die statt über alle Wellenlängen hinweg innerhalb eines Spektrums zu normieren, eine Normierung über eine Wellenlänge über alle Spektren vorsieht. Auch in dieser Methodik werden die Maxima-Ausreißer berücksichtigt, wobei die Auswirkungen für alle Spektren identisch sind. Damit sind die Spektren ebenfalls vergleichbar, aber der Berechnungsaufwand ist höher, da jedes Spektrum mindestens zweimal prozessiert wird. Zusätzlich muss die vollständige Berechnung aller Spektren wiederholt werden, sobald ein neues Spektrum nach der Berechnung bzw. Analyse neu hinzukommt, falls sich das Maximum oder Minimum verändern würde.

Eine weitere Problematik ist, dass alle Werte mit einfach genauen Fließkommazahlen repräsentiert werden. Für eine ungünstige Kombination von sehr kleinen bzw. sehr großen Messwerte können numerische Effekte auftreten. Diese Situation kann durch den Einsatz von Fließkommazahlen mit doppelter Genauigkeit verbessert werden, was den Speicherbedarf zur Speicherung, zur Analyse im Hauptspeicher aber verdoppelt und nur das Risiko des Effekts minimiert, aber nicht eliminiert.

Dies zeigt, dass der Einsatz eines einfachen Min/Max-Filters nicht sinnvoll ist, weshalb der Filter wie in Abbildung 3.4 dargestellt, nicht mehr in den Prozess integriert ist. Da der Filter jedoch für andere Anwendungsfelder sinnvoll sein könnte, ist dieser in dieser Thesis dennoch aufgeführt.

Bereichsselektion

Im Anschluss an die Schritte der Rotverschiebungskompensation und des Resamplings ist jede Dimension einer Spektrums mit dem Index p mit der p-ten Dimension eines anderen Spektrums vergleichbar. Eine manuelle Inspektion der um die Rotverschiebungen korrigierten und mit dem Resampling prozessierten Spektren zeigt, dass die korrigierten Messdaten teilweise

weit außerhalb des Bereiches $3600,\text{Å} \leq \lambda \leq 10000\,\text{Å}$ liegen. Dieser Effekt kann sich bei wenigen Spektren auf mehr als 50% der Messwerte auswirken. Der Bereichsselektionfilter muss daher den gleichen Wellenlängenbereich aus allen Spektren extrahieren, der am meisten Daten bietet. Um die Prozessierung nicht weiter zu verzögern, ist eine manuelle Selektion implementiert. Diese nimmt die Parameter $offset$ und $numberofvaluestoimport$ entgegen den entsprechenden, um Bereich der Extraktion zu definieren. Der Wert des Parameter $offset$ entspricht der Wellenlänge berechnet mit $p = offset$ aus Gleichung 3.7. Abbildung 3.6 zeigt beispielhaft ein Spektrum, dessen Werte stark rotverschoben waren und daher die Majorität der Messwerte nicht vorhanden sind. Im Gegensatz dazu stellt Abbildung 3.7 beispielhaft ein Spektrum mit geringer Rotverschiebung dar. Dies entspricht einem typischen vorprozessierten Spektrum.

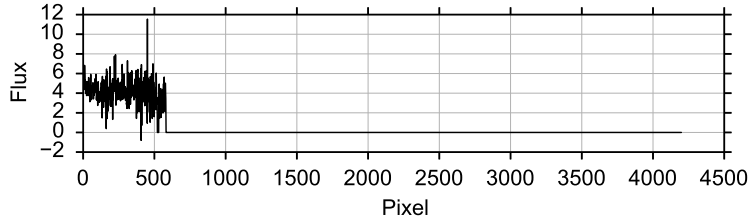


Abbildung 3.6: Beispielspektrum nach der Prozessierung mit hoher Rotverschiebung (Platte: 0269, MJD: 51581, Faser: 59)

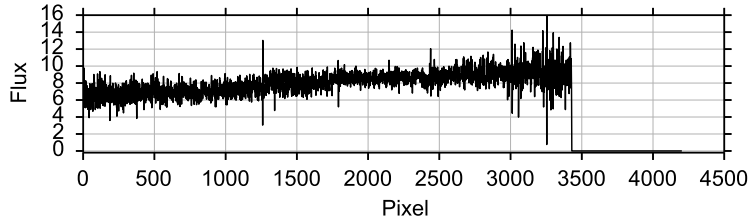


Abbildung 3.7: Beispielspektrum nach der Prozessierung mit geringer Rotverschiebung (Platte: 0269, MJD: 51581, Faser: 170)

Kapitel 4

Fractal-Similarity-Measures Verfahren

Der Hauptteil dieser Dissertation beschäftigt sich mit dem eigens entwickelten Verfahren (Fractal-Similarity-Measures (FSM)) zur automatisierten (unsupervised) Clustering Analyse von hochdimensionalen und unsicheren Daten mit großem Datenvolumen. Das Clustering von Daten ist ein Teilgebiet der Informatik und Statistik und beschäftigt sich schon vor den 70er Jahren mit der Detektion von Gruppierungen in Datensätzen [41, 73, 78]. Jedes Clustering Verfahren hat eigene Vorteile bzw. Nachteile bzgl. unterschiedlicher Anwendungsgebiete bzw. Einsatzszenarien.

Die Aufgabe des FSM Algorithmus ist die Fähigkeit, die drei genannten Eigenschaften (hohe Dimensionalität der Daten, Unsicherheiten in Daten und große Datenvolumina) in einem Verfahren zu vereinen unter der Berücksichtigung der Skalierbarkeit zur Analyse der SDSS Daten. Das heisst, die Berechnungskomplexität sowie die Speicherungskomplexität müssen in einer akzeptablen Komplexitätsklasse liegen. Das folgende Beispiel verdeutlicht die enormen Unterschiede zwischen einer akzeptablen und nicht akzeptablen Komplexitätsklasse.

Eine Komplexitätsklasse bzgl. der Berechnung von $>= O(n^2)$ eines Datensatzes mit großem Datenvolumen (z.B. $n = 3 \cdot 10^6$ Instanzen) ist nicht akzeptabel, da die Möglichkeiten der Parallelisierung durch die sequentiellen Anteile in einem Algorithmus je nach Algorithmus begrenzt sind. Für das konkrete Beispiel sei die Aufgabe der Berechnung einer Distanzmatrix N^2 zur distanzbasierten Analyse aus einer Menge an n Instanzen ($|N| = n$) und den Elementen $a, b \in N$ mit der Distanzmetrik $d(a, b)$ gegeben. Als weitere Annahme wird für eine Operation mit beliebiger Kombination von Elementen aus a und b $\forall a, b \in N(a \neq b) : d(a, b)$ eine durchschnittliche Berechnungszeit von $0.1\,ms$ geschätzt. Die Berechnung der Matrix mit einer asymmetrischen Distanzmetrikfunktion $d(a, b)$ dauert rein sequentiell $28.5388\,Jahre$ bei $9 \cdot 10^{12}$ zu berechnenden Distanzen. Im Vergleich dazu

benötigt ein Algorithmus der Berechnungskomplexitätsklasse $O(n \log n)$ sequentiell nur ca. $1.24\,h$.

Analog dazu wirkt sich die Komplexität der Speicherung der Daten aus. Angenommen es seien alle Distanzen der Distanzmatrix zu speichern und eine Distanz könne durch eine einfach genaue Fließkommazahl repräsentiert werden. In diesem Fall sind für die Speicherung aller Distanzen im $32\,TByte$ bei $O(n^2)$ erforderlich sowie $170\,MByte$ im $O(n \log n)$ Fall. Sind die Dimensionen ein Faktor, der in der Berechnung der Instanzen eine Rolle spielt, erweitert sich der Speicherplatzbedarf entsprechend um den Faktor der Anzahl der Dimensionen. Daraus resultieren im quadratischen Fall mit 5000 Dimensionen $160\,PByte$ an Speicher. Die Optimierung eines Algorithmus bzgl. der Skalierbarkeit ist daher eine wichtige Aufgabe.

Das FSM-Verfahren ist in der Lage die SDSS Daten bestehend aus den 3×10^6 Spektren, bis zu 5100 Messwerten und Unsicherheiten in akzeptabler Zeit zu analysieren. Teile der folgenden Unterkapitel wurden bereits in einem eigenen Konferenzbeitrag in [68] vorgestellt.

Das folgende Kapitel 4.1 beschreibt das grundsätzliche Konzept und den Aufbau der Methodik. Die theoretischen Grundlagen und die Möglichkeiten der Nutzung der Fraktalen Dimension zur Datenanalyse skizziert Kapitel 4.2. Die Kapitel 4.3 und 4.4 beschreiben den entwickelten Fraktalen Informationswert sowie dessen Einsatz in einer eigenen Metrik zur Detektion der relativen Ähnlichkeit einer Dateninstanz zu einem Cluster. Kapitel 4.5 zeigt, wie Unsicherheiten in die Berechnungen verarbeitet und integriert werden. Die Bootstrapping Methodik sowie die SDSS bzw. spektrenspezifischen Modifikationen werden in Kapitel 4.6 diskutiert. Der Aufbau des zweigeteilten Algorithmus und dessen Komplexitäten werden in Kapitel 4.7, die Implementierung in Kapitel 4.8 dokumentiert. Das Kapitel endet mit einer Parameterdiskussion im Unterkapitel 4.9. Die Kapitel enthalten allgemeine Erläuterungen, wobei die für SDSS spezifischen Optimierungen hervorgehoben werden.

4.1 Aufbau der Methodik

Die Beschreibung des Aufbaus der Methodik nimmt Bezug auf den Anwendungsfall des Clusterings des SDSS Datensatzes. Eines der genannten Ziele bei der Entwicklung des Fractal Similiarity Measures Algorithmus ist dessen Skalierbarkeit bzgl. großer Datenmengen und hoher Anzahl an Dimensionen. Dieses Ziel wird primär erreicht durch den Einsatz des Fraktalen Ähnlichkeitsmaßes (Fractal Similarity Measure) und dem dazugehörigen Algorithmus. Abbildung 4.1 zeigt eine grafische Aufbereitung des Aufbaus der Methodik durch die Darstellung einzelner Schritte.

- Anfangs stehen die originalen SDSS Daten zur Verfügung. Um die Spektren einem Clustering Verfahren unterziehen zu können, prozes-

siert diese ein Datenvorarbeitungsschritt wie in Kapitel 3.4 beschrieben. Dies entspricht Schritt 0 im Schaubild.

- Die Implementierung des ersten Clustering Schritts im Verfahren (der Initialisierung) diskretisiert die Spektren. Unter der Annahme es existiert ein 2D-Graph mit einer Dimensions- und einer Dimensionswerteachse, werden die Spektren auf beiden Achsen diskretisiert. Die Diskretisierung wird auf der Dimensionswerteachse anhand sog. Boxgrößen (beschrieben in Kapitel 4.3) gruppiert, während im Falle der SDSS Daten zusätzlich eine Gruppierung auf der Dimensionsachse (dem sog. Binning, siehe Kapitel 4.6.1) mit Hilfe des Binning-Faktors stattfindet. Aus dem diesem Diskretisierungsverfahren entsteht das Gitternetz mit verschiedenen Diskretisierungen in Dimensions- und Dimensionswerteachse.
- Das hierarchische Clustering in Schritt 2 dem sog. Bootstrapping erzeugt aus einer kleinen Untermenge der Spektren initiale Cluster. Die Untermenge entspricht einer randomisierten Untermenge aller vorverarbeiteten Spektren. Entsprechend erstellte Cluster beinhalten die grobe Struktur des finalen Clusters und stellen die Basis für den hauptsächlichen Clustering Schritt dar. Das initiale Clustering, beschrieben in Kapitel 4.7.1, arbeitet auf dem Gitternetz mit der höchsten Informationsdichte um Cluster zu generieren. Insgesamt kann dieser Schritt mehrfach durch Rekursion wiederholt ausgeführt werden, bis eine weitere Aufteilung nicht weiter möglich ist oder weitere Abbruchkriterien zutreffen.
- In einem finalen Schritt 3, dem sog. Performance Clustering, weist das Verfahren unter Einsatz des Fraktalen Informationswerts sowie des Frakalen Ähnlichkeitsmaßes die von Schritt 2 ausgeschlossenen Spektren den Clustern zu. Dabei wird die Anzahl der Cluster primär nicht modifiziert mit Ausnahme der Identifizierung von Ausreißern, die in ein separates Cluster ausgegliedert werden. Am Ende dieses Schritts wurden alle Spektren genau einem Cluster zugeordnet und das Clustering Verfahren ist abgeschlossen.

Die Trennung zwischen Schritt 2 und 3 erlaubt eine unabhängige Entwicklung, Validierung sowie die Möglichkeit zum Austausch der Komponenten. Beispielsweise kann der Schritt des Bootstrappings durch andere effiziente oder domänenspezifische Komponente ersetzt werden, um optimierte initiale Cluster zu erhalten. Des Weiteren können auch bereits anderweitig (z.B. manuell) erstellte Cluster verwendet werden, um das Performance Clustering als Klassifikationsmethodik mit integrierter Ausreißerdetektion einsetzen zu können.

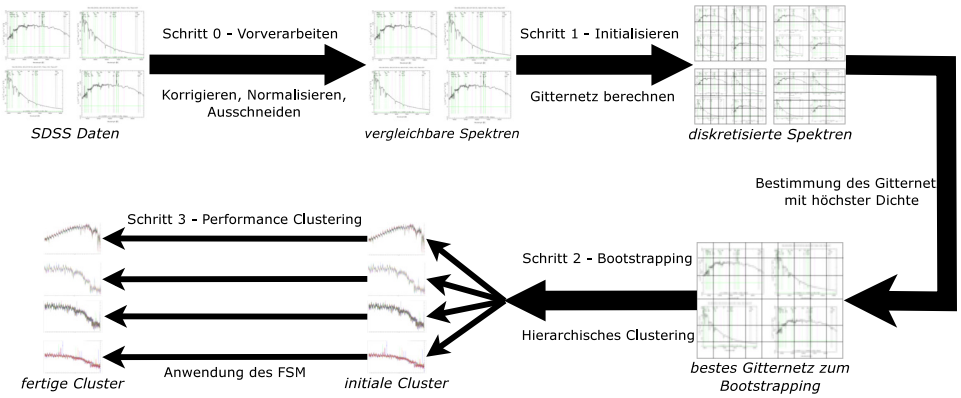


Abbildung 4.1: Abstrakte Darstellung des Aufbaus der Methodik

4.2 Fraktale Dimensionen und Box Counting Methode

Aus der Theorie der Fraktalen Dimensionen stammt die Grundidee für den FSM Algorithmus. Der Begriff der Fraktalen Dimension ist in der Literatur in unterschiedlichen Kontexten erwähnt. Es existieren verschiedene formal definierte fraktale Dimensionen. Eine davon ist die sog. Hausdorff Dimension, die ein Maß für die Rauigkeit einer Oberfläche, Körper oder Graph darstellt [51]. Die Hausdorff Dimension ist die Basis für den Fraktalen Informationswert.

Geprägt und primär benannt wurde der Begriff eines Fraktals bzw. der fraktalen Geometrie von Mandelbrot in [93]. Ein Fraktal (lat. fractus für gebrochen) sind Gebilde, die eine geometrische skaleninvariante Selbstähnlichkeit von sich wiederholenden Mustern aufweisen. Dabei kann ein Fraktal geometrisch so geteilt werden, dass die einzelnen Teile identische bzw. approximativ identische Kopien des gesamten Teils ergeben.

Abbildung 4.2 zeigt vier Iterationsstufen einer Koch Flocke. Initial besteht diese aus einer geraden Linie. Durch eine Drittelung wird das dadurch entstehende mittlere Stück entfernt und durch ein offenes Dreieck ersetzt, was in der ersten Iteration der Abbildung (links oben) zu sehen ist. Für jede weitere Iteration wird dieses Verfahren pro Teilabschnitt einfach-rekursiv wiederholt. Diese zeigen, dass sich das Hauptmuster des offenen Dreiecks in höheren Iterationsstufen grundsätzlich wiederholt.

Vergleichbar ist das Sierpinski Dreieck in Abbildung 4.3, das durch die rekursive Aufteilung von Dreiecken in vier kongruente Dreiecke entsteht. Durch eine entsprechende Skalierung bzw. Magnifizierung von höheren Iterationsstufen kann das initiale Muster wieder herauskristallisiert werden. Diese Fraktale dieser Art werden als exakt selbstähnlich bezeichnet.

Neben dieser geometrischen Selbstähnlichkeit existiert eine statistische

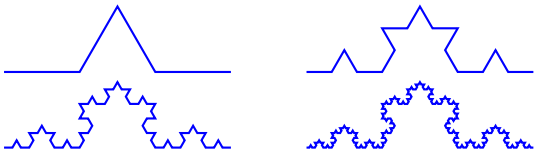


Abbildung 4.2: Koch-Flocke in verschiedenen Iterationsstufen

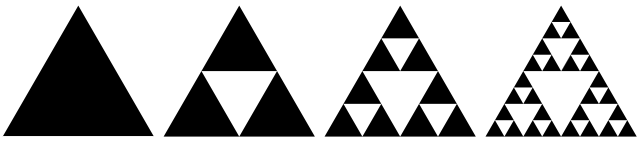


Abbildung 4.3: Sierpinski Dreiecke

Selbstähnlichkeit [44] bzw. statistische Fraktale. Diese bestehen aus Ähnlichkeitsbeziehungen deren stochastische Maße (anstatt der geometischen) über verschiedene Skalen hinweg invariant sind. Zusätzlich existieren neben der geometrischen und statistischen Selbstähnlichkeit weitere Arten der Selbstähnlichkeit, die durch formale Definitionen beschrieben sind. Diese sind in dieser Thesis nicht weiter diskutiert, da sie nicht relevant für das Verständnis der Methodik sind [44].

Eine interessante Eigenschaft der Fraktale ist deren Dimension. Die Dimension von geometrischen nicht fraktalen Objekten ist ganzzahlig: Eine Line hat eine, eine Fläche hat zwei, ein Würfel hat drei Dimensionen bzw. ($D = 3$). Für Fraktale existieren nach [93] jedoch nicht ganzzahlige Dimensionen. Eine Methode um diese zu bestimmen, ist die sog. Box Counting Methode, die durch Änderungen der Skalierung des Maßes eines Objekts eine Näherung an die Hausdorff Dimension eines Objekts berechnet.

Zum Verständnis der Box Couting Methode ist das formal definierte Hausdorff Maß zu nennen. Eine d-dimensionale Punktmenge $X \subset \mathbb{R}^d$ wird mit einer zählbaren Menge an d-dimensionalen Bällen $B_i : i \in \mathbb{N}^+$ überdeckt. Diese sog. ϵ-Überdeckungen ($\epsilon > 0$) erstecken sich über den gesamten Raum. Dabei entspricht $|B_i|$ dem Durchmesser des Balls, der $|B_i| \leq \epsilon$ ist. D.h. ϵ ist der maximale Durchmesser eines Balls. $H_s^\delta(X)$ beschreibt das δ-dimensionale Hausdorff-Maß von X.

$$H_s^\delta(X) = \lim_{\epsilon \to 0} inf \left\{ \sum_{i=1}^{\infty} |B_i|^\delta \text{ ist eine } \epsilon \text{ Abdeckung von } X \right\}$$

Dazu existiert ein einziger, nicht-negativer Wert $D \in \mathbb{R}$, sodass $H_s^\delta(X) = \infty$ für $\delta < D$ und $H_s^\delta(X) = 0$ für $\delta > D$. Dieser Wert entspricht der sog. Hausdorff-Dimension [44, 51, 64]. Die Notation wurde aus [44] übernommen.

Des Weiteren existiert eine Box Counting Dimension, siehe Gleichung 4.1.

$$D_{BC} = \lim_{\epsilon \to 0} \frac{\log N(\epsilon)}{\log(1/\epsilon)} \tag{4.1}$$

Unter bestimmten Bedingungen ist die Box Couting Dimension identisch zur Hausdorff Dimension [44]. Der primäre Unterschied zwischen der Hausdorff-Dimension bzw. der Box Counting Dimension ist die Methodik der Raumüberdeckung. Während die Hausdorff Dimension die Überdeckung mit Kugeln umsetzt, implementiert die Box Counting Dimension Würfel. Der Parameter ϵ repräsentiert dabei die exakte Kantenlänge eines Würfels $\in \mathbb{R}^d$ und $N(\epsilon)$ entspricht der minimalen Anzahl an besetzen Würfel im Raum.

Im Unterschied zur Hausdorff-Dimension lässt sich die Box-Counting Dimension effizienter berechnen, da kein Infimum zu bestimmen ist. Darüber hinaus bestimmt die Box-Counting Dimension eine Näherung an die analytische fraktale Dimension [44]. Das folgende Beispiel wendet die Box-Counting Methodik auf die Koch Flocke an: Durch die Aufteilung pro Iteration existieren jeweils 4 Liniensegmente auf einer Geraden ($N(\epsilon) = 4$), die durch eine topologische Aufteilung mit Hilfe eines Skalierungsfaktors $\epsilon = 1/3$ geteilt werden. Dieser Aufteilungsfaktor repräsentiert die drei einzelnen Teile der Kochkurve. Durch Einsetzten in 4.1 hat die Koch Flocke $D_{BC} = \log(4)/\log(3) \approx 1.26$. Dieser Wert verändert sich entsprechend nicht bei Anpassung der Skalen (ϵ). Beispielsweise ergibt ein Skalierungsfaktor von $\epsilon = 1/9 \rightarrow D_{BC} = \log(16)/\log(9) \approx 1.26$.

Dennoch sind aufgrund der Grenzwertbetrachtung bei der Bestimmung der Box Counting Dimension bei diskreten Mengen mit nicht exakter Selbstähnlichkeit mit in akzeptabler Rechenzeit nur Annäherungen an die Dimension möglich. Zur Abschätzung dieser Dimension existieren in der Literatur einige Arbeiten z.B [47, 87, 108]. Weitere Arbeiten beschäftigen sich mit der Box Counting Methodik in einem breiten Anwendungsfeld, beispielsweise in der Bildverarbeitung zur Mustererkennung z.B. [31, 85], in der Medizin zur Analyse von strukturellen Aspekten in der Neuroanatomie [38] oder zur Analyse von Sozialen Graphen [113].

Die Verwendung der Box Counting Methodik in der aktuellen Literatur beschäftigt sich i.d.R. mit der Analyse einzelner Strukturen auf geometrischer oder statistischer Ebene z.B. [50]. Die analysierten einzelnen Strukturen der Publikationen sind entsprechend Bilder, Zeitreihen oder spektrenähnlichen Graphen, die durch deren fraktale Eigenschaften im statistischen oder geometrischen Sinn differenziert werden.

Der Ansatz der reinen Bezifferung von besetzen Boxen wird in [54] zur Analyse von dynamischen Systemen bestehend aus Attraktoren um weitere Aspekte ergänzt. Im Falle einer (diskreten) Datendarstellung einer Zeitreihe können durch die entsprechend definierten Boxen ein bzw. mehrere Punkte in eine Box fallen. Durch die multiplen Punkte in einer Box, kann dessen Population bzw. Entropie als Alternative zur reinen Bezifferung besetzter

Boxen $N(\epsilon)$ dienen, siehe Gleichungen 4.2,4.3. Gleichung 4.2 übernimmt die Rény Entropie [111] als Informationsbasis zur Bestimmung einer fraktalen Informationsdimension.

$$D_{Inf.} = \lim_{\epsilon \to 0} \frac{\sum_{i=1}^{N(\epsilon)} p_i \log p_i}{\log \epsilon} \tag{4.2}$$

$$D_q = \lim_{\epsilon \to 0} \frac{1}{1-q} \frac{\log \left(\sum_{i=1}^{N(\epsilon)} p_i^q \right)}{\log(1/\epsilon)} \tag{4.3}$$

In [54] beschreiben die Autoren ein Verwendungsszenario sowie deren Bestimmung des p_i der besetzten Boxen der Attraktoren. Die Variable p_i entspricht der Wahrscheinlichkeit, dass ein Punkt sich in der entsprechende Box mit dem Index i befindet [54]. Eine Verallgemeinerung der Shannon Entropie ist die Rény Entropie [104], die in Gleichung 4.3 als Informationsbasis dient. Eine Änderungen in der eigentlichen Messgrundlage (Zählung, Rény Entropie bzw. Populationsfrequenz) ermöglicht eine qualitativ unterschiedliche Analyse für verschiedene Aspekte. Während die Populationsfrequenz zusätzlich zur reinen Zählungsmethodik die Verteilung der Elemente beachtet, liefert die Rény Entropie eine Analyse bzgl. des Informationsgehaltes der einzelnen Box.

Alle skizzierten Methodiken und Theorien basieren auf dem Ansatz der Analyse eines einzelnen Objekts (z.B. einem Attraktor eines dynamischen Systems). Die in dieser Thesis vorgestellte Methodik erweitert diesen Ansatz mit der Idee, nicht ein einzelnes Objekt jedoch eine Gruppe von Objekte bzgl. der fraktalen Eigenschaften zu analysieren und deren Unsicherheiten in die Analyse zu integrieren.

4.3 Fraktaler Informationswert

Der in diesem Kapitel definierte fraktale Informationswert bzw. Fractal Information Value (FIV) ist eine Repräsentation des Ergebnisses einer modifizierten Box Counting Methode. Ein FIV repräsentiert dabei einen spezifischen Informationswert für eine analysierte Gruppe von Objekten. Der FIV wird vom fraktalen Ähnlichkeitsmaß verwendet zur Detektion einer Ähnlichkeit einer Dateninstanz zu einem Cluster. Mit Hilfe der L^2 Norm als Indikator können Änderungen des FIVs durch Änderungen am Cluster festgestellt werden. Diese Änderungen dienen als Maß zur Detektion einer relativen Ähnlichkeit, was die folgenden Abschnitte beschreiben.

Abstrakt betrachtet, kann ein Cluster als ein hochdimensionaler Körper angesehen werden. Ein Datensatz mit d Eigenschaften bzw. Dimensionen stellt einen d-dimensionalen Körper im Datenraum dar. Jedes Cluster besitzt durch seine Unterschiede zu anderen Clustern eine charakteristische d-dimensionale Form, die die Berechnungsmethodik des FIV analysiert. Durch

Operationen am Cluster mittels Hinzufügen bzw. Entfernen von Cluster Mitgliedern verändert sich die Charakteristik der Körperform. Eine Dateninstanz die perfekt in das Cluster passt, verändert nach dem Hinzufügen die Charakteristik bzw. den FIV nicht signifikant. Eine Veränderung des Clusters durch Hinzufügen einer unpassende Dateninstanz ist analog dazu gegeben.

Diese quantifizierbare Veränderung wird durch den folgenden Prozess ermittelt. Das Verfahren diskretisiert den d-dimensionalen Datenraum in d Dimensionale Boxen mit einer konstanten bzw. dimensionsabhängigen Boxgröße g. Alle Boxen zusammen betrachtet ergeben ein hochdimensionales Gitternetz, das den Datenraum überdeckt und damit diskretisiert. Die Analyse, der mit dem Gitternetz schablonierten Verteilung der Daten, ist die Hauptaufgabe des FIV. Die folgende Beschreibung repräsentiert den Fall einer konstanten Boxgröße g für alle Dimensionen zum einfacheren Verständnis. Prinzipiell ist der Einsatz dimensionsabhängiger Boxgrößen möglich und ausschließlich in der Implementierung relevant.

Unter der Annahme es existiert nach dem Clustering Prozess eine Menge an Clustern

$$C = \{C_1, C_2, .., C_k\}, ||C|| = k$$

mit C_i als das i-te Cluster und der Kardinalität k der Menge $|C|$. Jedes Cluster C_i besteht aus einer Menge an Cluster-Mitgliedern DI_{C_i}, die eine Teilmenge aller Dateninstanzen $DI_{C_i} \subset DI$ sind. Die Dateninstanzen bestehen aus d Dimensionen mit:

$$DI = \{\mathbf{di}_1, \mathbf{di}_2, ..., \mathbf{di}_n\}, \mathbf{di}_j \in \mathbb{R}^d$$

Analog zu C_i ist $\mathbf{di}_j$ die j-te Dateninstanz der Menge aller zur Verfügung stehenden Dateninstanzen DI mit Kardinalität n. Die Methodik spannt das o.g. hochdimensionale Gitternetz mit der Kantenlänge g auf. Das Gitter wird durch die Abbildung $\mathbf{dig}_j(g)$ repräsentiert:

$$\forall \mathbf{di}_j \in DI_{C_i} : \mathbf{dig}_j(g) = grid(\mathbf{di}_j, g) = \lceil \mathbf{di}_j * (1/g) \rceil \tag{4.4}$$

Die Grid-Funktion diskretisiert durch die Abbildung $\mathbf{dig}_j(g) : \mathbb{R} \to \mathbb{N}$. Anzumerken ist, dass alle $\mathbf{di}_j$ Vektoren $\in \mathbb{R}^d$ sind und die Multiplikation $\mathbf{di}_j * g$ komponentenweise auszuführen ist. Jedes $\mathbf{dig}_j(g)$ entspricht damit der diskretisierten Form von $\mathbf{di}_j$ für ein gegebenes Boxgröße g, aufgerundet pro Dimension/Komponente. Die Werte in $\mathbf{dig}_j(g)$ entsprechen der Koordinate der Box im d-dimensionalen Raum.

Alle diskretisierten Elemente eines Clusters C_i sind in einer von g abhängigen Menge

$$DIG(g) = \{\mathbf{dig}_1(g), \ldots, \mathbf{dig}_r(g)\}, r = ||DI_{C_i}|| = ||DIG(g)||$$

definiert. Pro Cluster existiert eine eigene Menge,

$$DIG_{C_i} = \{DIG(g_0), \ldots, DIG(g_o)\}$$

die die Box Counting Datenstruktur zur Analyse darstellt und aus o Elementen besteht ($g \in G, o = ||G||$). Durch Variieren der Anzahl der eingesetzten Boxgrößen o, bzw. der Menge DIG_{C_i}, kann der Detailgrad der Analyse verändert werden. Weiterhin bestimmt auch die Wahl der einzelnen Boxgrößen in G, wie detailliert Unterschiede im Datenraum analysiert werden. Je kleiner die minimale Boxgröße, desto präziser sind die Unterschiede, die festgestellt werden können. Kapitel 4.9 diskutiert diesen Punkt genauer.

Nach einer Diskretisierung befinden sich die Elemente in einer Box, die über alle Dimensionen gleiche Werte besitzen. Benachbarte Boxen zeigen ähnliche Elemente bzgl. der jeweiligen Boxgröße g. Insgesamt entspricht dies der Ähnlichkeit der Dateninstanzen in Abhängigkeit von einer Boxgröße.

Die Diskretisierung des Datenraums ergibt eine Partionierung von u besetzen Boxen. Die Menge $BX(g_l) = \{BX_1(g_l), \ldots, BX_u(g_l)\}$ beinhaltet die besetzten Boxen eines Clusters für eine definierte Boxgröße g_l. Das heisst, für jede Boxgröße existiert für jedes Cluster eine eigene Menge $BX(g_l)$.

Jedes Cluster hat i.d.R. pro Boxgröße mehrere besetzte Boxen, die äquivalent groß, jedoch nicht äquivalent besetzt sind. Im Extremfall einer sehr groß gewählten Boxgröße können sich jedoch alle Elemente in einer Box befinden. Nicht besetzte Boxen sind für die Analyse nicht relevant. Eine relative Kardinalität bzw. Populationsfrequenz der s-ten Box ist folgendermaßen definiert:

$$p(BX_s(g)) = ||BX_s(g)||/r.$$

Durch Aufsummieren aller relativen Kardinaltiäten $p(BX_s(g))$ unter Berücksichtigung der Rény-Entropie ist der FIV in den Gleichungen 4.5 und 4.6 definiert.

$$FIV(C_i, q, g) = log(\sum_{s=1}^{u} p(BX_s(g))^q * log(p(BX_s(g))^q), \quad (4.5)$$

$$FIV(C_i, q, g) = log(\sum_{s=1}^{u} p(BX_s(g))^q). \quad (4.6)$$

Die Gleichungen unterscheiden sich leicht in der Verwendung. Während Gleichung 4.5 für den Einsatz $q = 1$ konzipiert ist, wird für alle anderen Werte von q Gleichung 4.6 verwendet. Der Parameter q bestimmt, welche Art der Analyse durchgeführt werden soll. Berechnungen mit $q = 0$ entsprechen der klassischen Box Counting Methodik und das Ergebnis ist die Anzahl der besetzen Boxen. Andernfalls integriert die Metrik zusätzlich die Verteilung bzw. Frequentierung der Boxen, wobei größere q die Frequentierungen entsprechend anders berücksichtigen.

4.4 Fraktales Ähnlichkeitsmaß

Das fraktale Ähnlichkeitsmaß (Fractal Similarity Measure, FSM) ist ein quantitatives relatives Vergleichsmaß, das zur Bestimmung der Cluster-Zugehörigkeit eines zu testenden Cluster-Mitglieds dient. Die in Kapitel 2.3.6 vorgestellten Arbeiten [11, 12] nutzen zur Ermittlung der Ähnlichkeit die Steigung einer Regressionslinie. Wir konnten feststellen, dass der Einsatz von Riemann-Summen gegenüber den partiellen Ableitungen zur Ermittlung der Steigungen eine höhere Sensitivität gegenüber den Änderungen aufweist.

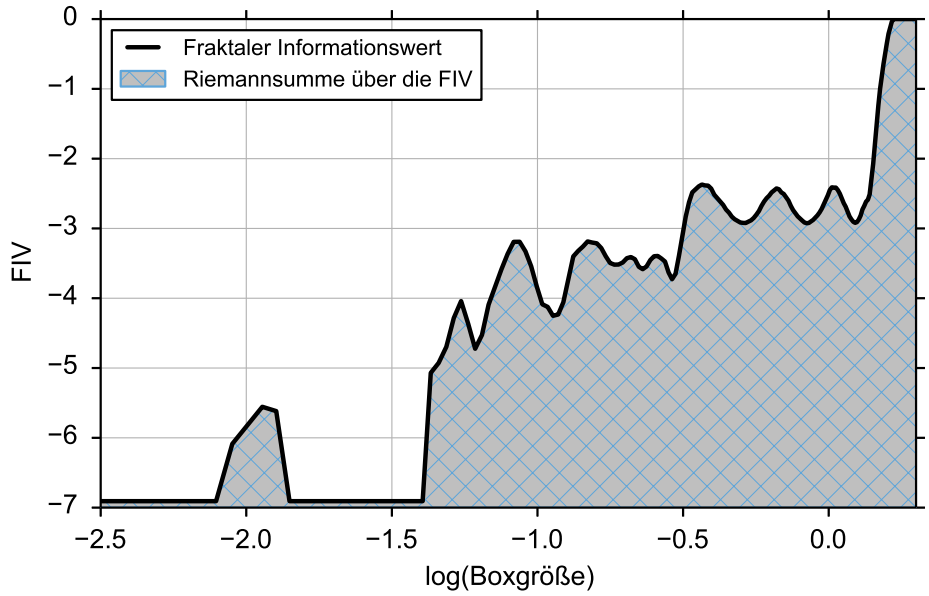


Abbildung 4.4: Visualisierung des Fraktalen Informationswerts und der Riemansumme

Abbildung 4.4 zeigt beispielhaft die Berechnung verschiedener FIV zur detallierten Darstellung mit $o = 50$. Auf der Abszisse aufgetragen ist die logarithmische Boxgröße und die Ordinate zeigt den FIV. In der größten Boxgröße befinden sich als Extremfall alle Elemente in einer Box,

$$||BX(g_{max})|| = 1, \forall \mathbf{di}_j \in DI : g_{max} \geq max(\mathbf{di}_j)$$

bei der kleinsten Boxgröße befinden sich alle Elemente jeweils ein einer eigenen Box:

$$||BX(g_{min})|| = n, \mathbf{di}_j \in DI : g_{min} < min(\mathbf{di}_j)$$

Insgesamt ist der Verlauf der FIV bei passender Wahl der minimalen, maximalen und dazwischen liegenden Boxgröße im Clustering Verfahren charakteristisch und erkennbar an der Form eines vertikal gespiegelten Buchstaben

„z“. Abbildung 4.4 zeigt diese Form, wobei aufgrund der großen Anzahl der dargestellten Boxgrößen die Form horizontal gezerrt ist.

Der Verlauf zwischen minimaler bis maximaler Boxgröße ist bei jedem Cluster individuell. Je ähnlicher sich zwei Cluster sind, desto ähnlicher sind deren graphisch veranschaulichte übereinander gelagerte Verläufe der FIV. Durch eine Maximierung der Anzahl der verwendeten Boxgrößen kann daher eine genauere Charakteristik berechnet werden. Eine Abwägung zwischen der Anzahl der verwendeten Boxgrößen bzw. des Berechnungsaufwands und der benötigten Präzision ist sinnvoll (siehe Kapitel 4.9).

Bei genauer Betrachtung lassen sich die Unterschiede innerhalb eines Clusters anhand der FIV einzelner Boxgrößen erkennen. In Abbildung 4.4 sind auf der Abszisse in Richtung $-\infty$ die größeren, in Richtung 0 die kleineren Boxgrößen aufgetragen. Es existieren wenige Boxen ($g \geq e^{-2.1}$, maximale Boxgröße) die besetzt sind. Je kleiner die Boxgröße wird, desto mehr Variationen treten auf:

$$FSM(C_i, \mathbf{di}_j, q, g) = \sqrt{\mathcal{I}_{g \in G}[(FIV_{after} - FIV_{before})^2]}. \tag{4.7}$$

Die Berechnung des FSM erfolgt nach Gleichung 4.7. Das FSM wird nur im dazugehörigen Algorithmus eingesetzt, der in Kapitel 4.7 beschrieben ist. Der Algorithmus basiert auf der Idee, ein Kandidat $\mathbf{di}_j$ testweise zu einem Cluster $C_i \in C$ hinzuzufügen. Das FSM vergleicht die Werte vor dem testweisen Einfügen $FIV_{before} = FIV(C_i, q, g)$ bzw. danach $FIV_{after} = FIV(C_i \cup \mathbf{di}_j, q, g)$ mit Hilfe der Riemann-Summe unter Einbezug der L^2-Norm in Gleichung 4.7. Mit Hilfe der Riemann-Summe ist es möglich, das Gesamtintegral zu approximieren, um die relativen diskreten Änderungen der Körperform festzustellen. Der Vergleich der beiden Werte dient als Maß und repräsentiert den Einfluss des hinzugefügten Elements bzgl. der Form des hochdimensionalen Körpers.

Zur Visualisierung der Unterschiede nach dem testweisen Hinzufügen von Elementen zeigt Abbildung 4.5 ein Beispiel. Dieses besteht aus zwei Abweichungsgraphen, die eine Cluster-Manipulation mittels Hinzufügen eines passenden Elementes bzw. eines weniger passenden Elements darstellen. Der Wert FIV-Abweichung (Δ) wird pro Boxgröße generiert durch $\Delta = (FIV_{before}/FIV_{after}) - 1$. Wie in der Abbildung dargestellt, verursachen passendere Elemente eine geringer Änderung an der Clusterform und damit an den FIV als ein unpassendes Element.

Abbildung 4.6 zeigt die verwendeten Daten aus Abbildung 4.5 mit dem passenden sowie dem unpassenden Element in einer einzelnen Ansicht. Der verwendete synthetische Datensatz ist in Kapitel 5.2.1 beschrieben. Die FIV-Abweichungen zeigen durch einen überwiegend identischen Graphenverlauf, dass die beiden Testspektren ähnlich aufgebaut sind, jedoch auch Unterschiede (z.B $\log(\text{Boxgröße}) \approx -0.35$) besitzen. In der Betrachtung der Unterschiede der (nicht) passenden Elemente in Abbildung 4.6 fällt auf, dass

sich jeweils eine (charakteristische) Spitze grundsätzlich vom Rest unterscheidet. Da sich die Unterschiede auf zwei Dimensionen konzentrieren sind die Abweichungen in Abbildung 4.5 entsprechend der Analyseskala gering.

4.4.1 Optimierungen

Die Berechnung des FSM kann durch folgende Optimierungen verbessert werden, die auch im Fall der SDSS Daten im Algorithmus implementiert sind.

Ausgleich heterogener Cluster Kardinalitäten

Experimente mit dem FSM zeigen die Eigenschaft, eine Abhängigkeit zur Cluster-Größe zu entwickeln. Zur Laufzeit des Algorithmus werden Daten zu den jeweiligen Clustern hinzugefügt und der Einfluss einzelner Elemente bei einer Cluster-Manipulation sinkt. Dies ist bei der Gesamtbetrachtung des Clusters ein Problem sobald die Cluster ungleichmäßig wachsen bzw. die Verteilung der Objekte nicht homogen ist. Unterschiede in den höheren Größenordnungen verschiedener Cluster verschieben den FSM linear in kleinere Größenordnungen. Im Falle eines stark heterogenen Größenwachstums kann dieser Effekt zu irreführenden Selektionen des Clusters resultieren. Eine Fehlselektion dieser Art erklärt sich durch den geringen Einfluss eines Elements in einem großen Cluster versus dem Einfluss des selben Elements in einem kleineren Cluster. Um diesen Effekt auszugleichen wird der FSM bzgl. der Cluster-Größe normiert, siehe Gleichung 4.8.

$$FSM_{norm} = FSM * r \tag{4.8}$$

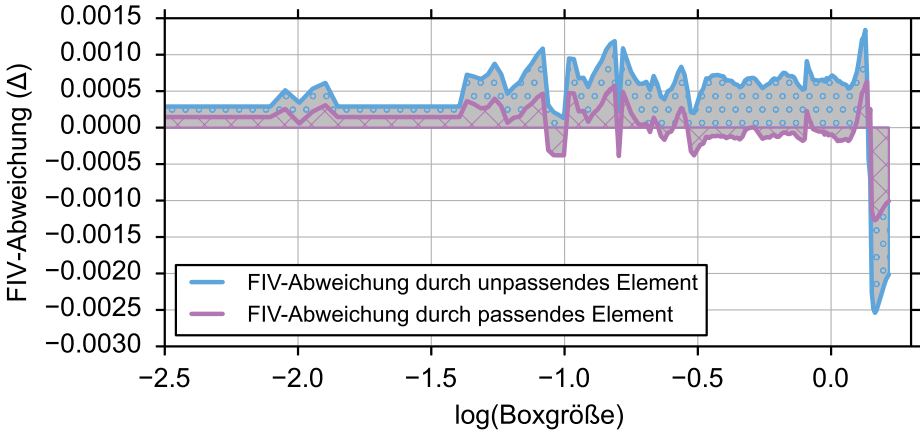


Abbildung 4.5: Visuelle Darstellung der Änderung des FSM nach Manipulation des Clusters

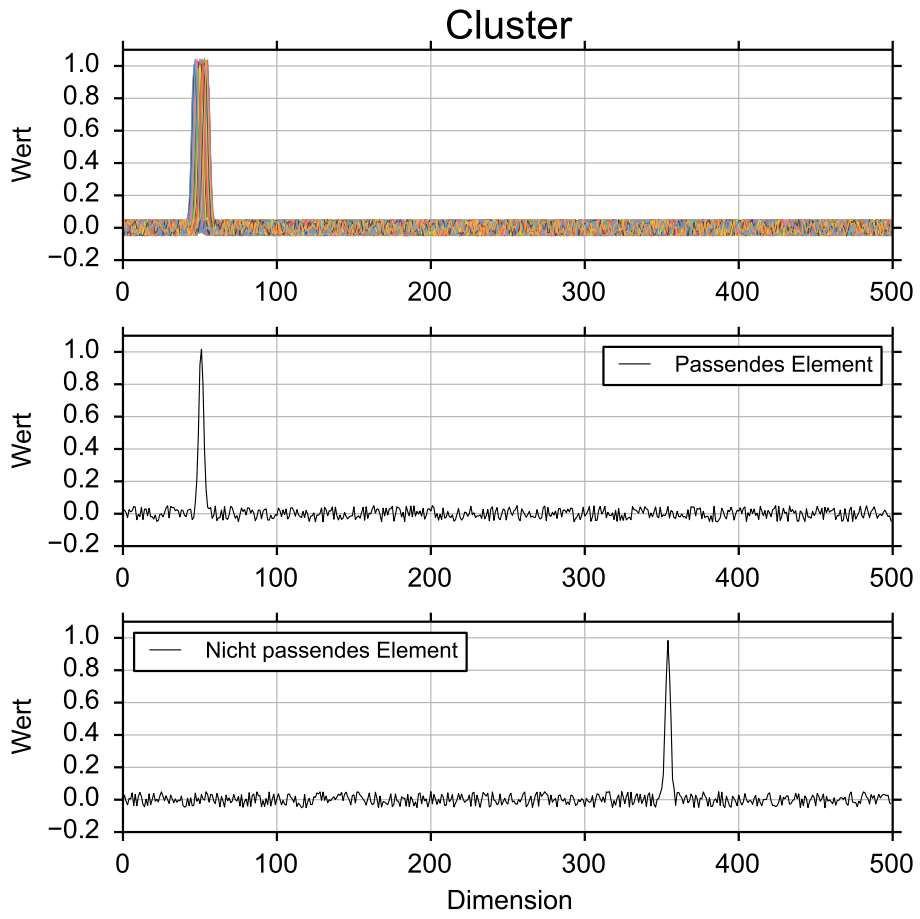


Abbildung 4.6: Datenansicht des Clusters sowie der Testelemente

Durch die Multiplikation wird ein evtl. zu großes Cluster und damit eine zu kleine Änderung ausgeglichen. Die Multiplikation erweitert den Einfluss eines Elements in einem großen Cluster relativ zum Einfluss eines Elements in einem kleinen Cluster. Die Normalisierung des FSM wird sowohl für das FSM_{before} als auch dem FSM_{after} ausgeführt.

Qualität der Cluster-Selektion

Eine absolute Quantifizierung der Selektionsqualität während des Clusterings ist nicht möglich. Primär muss definiert werden, was genau Qualität im Sinne des Clusterings bedeutet. Dies ist eine domänenspezifische Fragestellung, die absolut bemessen nur mit Hilfe eines Goldstandards umgesetzt werden kann. Des Weiteren muss eine auf die Definition aufbauende erstellte Metrik messen, wie gut ein Clustering Ergebnis ist. Da Clustering auch allge-

mein eine anwendungsabhängige Aufgabe darstellt und deren Ergebnis auch stark von den verwendeten Metriken und Algorithmen abhängt [79], ist der Begriff der Qualität für einzelne Clustering-Projekte neu zu definieren. In der Literatur wird der Begriff der Qualität oft mit dem Begriff der Präzision bzw. Korrektheit bzw. Homogenität gleich gesetzt (z.B. in [23, 45]). Dabei wird ein bereits mit Labeln versehener Clustering Datensatz geclustert und die Fehleranzahl gemessen, bzw. bei nicht gelabelten Daten und distanzbasiertem Verfahren der gewichtete mittlere geometrische Durchmesser des Clusters kalkuliert [125]. Es existieren weitere Definitionen von Clustering Qualität, die indivduell bzw. domänenabhängig definiert sind. Im Fall der SDSS Daten existiert zum Zeitpunkt der Erstellung dieser Arbeit kein Clusteringergebnis, das für einen Goldstandard oder als Vergleich genutzt werden kann.

Im Folgenden wird nicht die Qualität des Clustering Ergebnis, jedoch die relative Quantifizierung der Qualität der einzelnen Selektion eines Clusters für ein zu clusterndes Element vorgestellt. Die einzelnen o.g. FSM_{norm} Werte der Cluster stehen pro neuem Element zur Verfügung. Der Vergleich zwischen einzelnen FSM_{norm} Werten zeigt, dass diese sich bei einem am besten passenden Cluster (das Cluster mit dem kleinesten Wert für FSM_{norm}) und einer „zweitbesten“ Wahl um einen Faktor $\theta = 10^a, a > 1$ unterscheiden, falls die Cluster untereinander heterogen sind. Je kleiner dieser Faktor ist, desto weniger vertrauenswürdig ist eine Wahl eines Clusters. Ein geringerer Faktor bedeutet, dass ein Element in zwei unterschiedliche Cluster vergleichbar gut passen würde. Ist die Änderung der Form für verschiedene hochdimensionale Formen der jeweiligen Cluster gering, sind sich die jeweiligen Cluster ähnlich oder das Element passt in keines der beiden Cluster. Der zweite Fall tritt auf, da das am besten passende Cluster bzgl. der gegebenen Cluster nicht automatisch passend ist.

Um die richtige Selektion eines Clusters sichern zu können, wird eine weitere Prüfung eingeführt. Ein Liste von $FSMs$ bestehend aus FSM_{norm} Werten entsteht durch das einmalige testweise Einfügen von zu clusternden Elementen in alle verfügbaren Cluster. Diese Liste von $FSMs$ wird absteigend sortiert und die ersten beiden Werte werden in Relation gesetzt: $FSM_{rel} = FSMs[0]/FSMs[1]$. Dieses relative Maß annotiert mit Hilfe eines definierbaren Schwellenwerts (θ) die Selektion als zuverlässig bzw. unzuverlässig. Eine Messung ist zuverlässig, wenn $\theta < FSM_{rel}$. Im Falle einer zuverlässigen Selektion wird das Element dem Cluster zugewiesen, anderenfalls als Ausreißer deklariert.

4.5 Integration von Unsicherheiten

Unsicherheiten stellen einen Aspekt im Clustering dar, der die Qualität des Ergebnisses einer Analyse beeinflussen kann. Eine Integration von Unsicher-

heiten in einem Clustering Verfahren kann auf zwei sich nicht ausschließende Arten implementiert werden.

- Zum einen ist es möglich, Unsicherheiten im Ergebnis zu integrieren: In der Regel sind Clustering Algorithmen so konstruiert, dass diese eine Dateninstanz genau einem Cluster zuweisen. Durch unter anderem Unsicherheiten in Daten oder nicht vollständig homogene Cluster können einzelne Dateninstanzen zu mehreren Clustern passen. Ein Clustering Algorithmus kann diese Ambiguität erkennen und eine Zuweisung zu einem Cluster entsprechend mit einer Wahrscheinlichkeit oder Toleranz annotieren. Diese Qualität der Detektion von Grenzfällen ist eine Zusatzinformation, deren Einsatz domänenabhängig sinnvoll ist. Jedoch steigt durch die integrierten Ambiguitäten der Cluster Zuweisungen der Berechnungsaufwand, weshalb eine Abwägung des Einsatzes dieser Integration von Unsicherheiten im Einzelfall entschieden werden muss. Darüber hinaus ist auch eine höhere Schwierigkeit bei der Interpretierung der Analyseergebnisse gegeben, bei der definiert werden muss, ob bei einer nicht eindeutigen Zuweisung ein Grenzfall, ein Ausreißer oder eine doppelte Zuweisung angenommen wird.
- Eine weitere Möglichkeit ist die Integration der datenseitigen Unsicherheiten in die Clustering Analyse. In diesem Fall berücksichtigt ein Algorithmus die entsprechenden angegebenen Unsicherheitsmodelle bzw. berechneten Unsicherheiten in der Clustering Analyse.

In dieser Arbeit wird aufgrund des Berechnungsaufwands bzw. der beschriebenen Metriken der zweite Fall integriert. Da diese Arbeit die SDSS Spektren prozessiert und diese jeweils einer Klasse bzw. Template Spektrum [18, 139] zugeordnet werden können, betrachtet diese Arbeit nicht den ersten Fall.

Wie in Kapitel 3.2 beschrieben, beinhaltet der SDSS Datensatz berechnete inverse Varianzen ($1/\sigma^2$). Diese stehen pro Dimension und Spektrum zur Verfügung, womit jeweils ein Vektor $ivar \in \mathbb{R}^d$ gegeben ist. Durch instrumentale oder umgebungstechnisch bedingte Effekte, die zu fehlenden Messwerten führen, sind entsprechende Daten in den Spektren als unzuverlässig bzw. fehlend markiert. Diese fehlenden Werte können nicht zur Kalkulation der FIV bzw. FSM eingesetzt werden und werden durch die Vorverarbeitung mit einer inversen Varianz von $1/\sigma^2 = 0$ markiert. Erkennen die Berechnungsroutinen einen entsprechend markierten Messwert einer zu testenden Dateninstanz, wird dieser mit dem Wert der passenden Dimension des durchschnittlichen Cluster Spektrums ($\overline{DI}_{C_i}$) ersetzt.

Das durchschnittliche Cluster Spektrum ($\overline{DI}_{C_i}$) berechnet sich folgendermaßen. Im Allgemeinen wird das gewichtete Mittel einer Liste von Daten, bestehend aus Element x_i, die aus einer Wahrscheinlichkeitsverteilung mit bekannter Varianz σ_i^2 stammen, wie folgt berechnet. Mit einem Gewicht von

$$w_i = \frac{1}{\sigma_i^2}$$

ist das gewichtete Mittel

$$\overline{x} = \frac{\sum_{i=1}^{n}(x_i \sigma_i^{-2})}{\sum_{i=1}^{n} \sigma_i^{-2}} \tag{4.9}$$

wobei die Varianz des gewichteten Mittels ist definiert als:

$$\sigma_{\overline{x}}^2 = \frac{1}{\sum_{i=1}^{n} \sigma_i^{-2}} \tag{4.10}$$

Das durchschnittliche Cluster Spektrum besteht pro Dimension aus dem gewichteten Mittel aller Spektren eines Clusters unter Berücksichtigung der Unsicherheiten. Für deren Berechnungen werden die Definitionen des gewichteten Mittels mit den inversen Varianzen entsprechend angewandt:

$$\overline{DI}_{C_i} = \frac{\sum_{k=1}^{r} (\mathbf{di}_k \circ \mathbf{ivar}_k)}{\sum_{k=1}^{r} (\mathbf{ivar}_k)}, \tag{4.11a}$$

$$\text{wobei } \mathbf{ivar}_{\overline{DI}_{C_i}} = \sum_{k=1}^{r} (\mathbf{ivar}_k). \tag{4.11b}$$

Die $\circ$ Operation in Gleichung 4.11a entspricht dem Hadamard Produkt. Die Anzahl der Elemente in einem Cluster wird mit $r = ||DI_{C_i}||$ beschrieben. Eine Funktion *komp* liefert die s-te Komponente eines Vektors:

$$komp(\vec{vek}, s) = \vec{vek}^{\,T} * \vec{e}_s, e \in \{0,1\}^d.$$

Für die Berechnung der FIV bedeutet das:

$$\forall \mathbf{di}_j \in DI_{C_i}(testweise),$$
$$\forall s \in 1, \ldots, d \text{ mit } komp(\mathbf{ivar}_j, s) = 0 \text{ gilt:}$$
$$komp(\mathbf{di}_j, s) \leftarrow komp(\overline{DI}_{C_i}, s), \tag{4.12a}$$
$$komp(\mathbf{ivar}_k, s) \leftarrow komp(\mathbf{ivar}_{\overline{DI}_{C_i}}, s). \tag{4.12b}$$

Die Gleichungen 4.12 beschreiben die Überschreibungsoperation, die im Falle einer unzuverlässigen Messung ($komp(\mathbf{ivar}_j, s) = 0$) für sowohl den Wert (4.12a) in der jeweiligen Dimension/Komponente s als auch für die korrespondierende inverse Varianz (4.12b) ausgeführt wird. Die Verwendung dieser Integration der inversen Varianz bzw. des durchschnittlichen Cluster Spektrums wird im Implementierungskapitel 4.8 beschrieben.

4.6 Bootstrapping-Verfahren

Um den Einsatz des Fraktalen Ähnlichkeitsmaßes zu ermöglichen, benötigt der dazugehörige Algorithmus bereits initial erstellte Cluster. Zur Generierung dieser Cluster wurde dieses Bootstrapping-Verfahren entwickelt. Es basiert auf der grundsätzlichen Idee, dass eine Menge bzw. Liste/Tabelle/-Matrix an bereits diskretisierten Dateninstanzen sortiert werden kann um diese mit Hilfe eines Splitting Verfahrens aufzuteilen. Dimensionen werden in einer dazu erstellenden Tabelle durch Spalten, einzelne Instanzen durch Zeilen repräsentiert und die einzelnen Felder der Tabelle beinhalten die Werte der einzelnen Dateninstanzen. Dieser Ansatz kann prinzipiell als eigener Clustering Ansatz betrachtet werden, jedoch ist die Berechnungskomplexität (siehe Kapitel 4.7.4) höher als im Fall des Fraktalen Ähnlichkeitsmaßes bzw. dessen dazugehörigem Algorithmus. Durch die höhere Komplexität wird der Algorithmus nur als Bootstrapping Mechanismus angewendet und dies auf einer randomisiert gewählten Untermenge der Daten $DI_{sub} \subset DI, n_{sub} = ||DI_{sub}||, n_{sub} < n$.

In der Anfangssituation liegt eine Matrix $A : n_{sub}xd$ vor. Durch die Randomisierung befinden sich die Zeilen in der Matrix bzgl. des Splittings einer ungeordneten Reihenfolge. Das erste Ziel besteht daher aus der Sortierung der Zeilen.

Zusammengefasst sortiert ein Algorithmus die entsprechenden Zeilen mit dem sog. Vergleichsmaß für tabellarisch organisierte hochdimensionale Daten. Die Sortierung ordnet die Daten lexikographisch an, wodurch sich die Zeilen vor-/nacheinander in der Matrix befinden, die ähnliche gleiche diskretisierte Werte besitzen. Sind zwei Zeilen identisch, ist deren Reihenfolge instabil und bzgl. des weiteren Verfahrens unerheblich, da durch die Diskretisierung keine Unterschiede bestehen.

Ein erfolgreiches Bootstrapping mit dieser Methode hängt unter anderem von der Wahl der richtigen Boxgröße zur Diskretisierung ab. Die kleinstmögliche sinnvolle Boxgröße (siehe Kapitel 4.9.1) ergibt die größte Anzahl an Untermengen mit einer geringen Anzahl an Elementen in einem Cluster und einer präzisen Form des Clusters aufgrund der Boxgröße. Eine präzise Form eines Clusters, also eine detaillierte Selektion durch ein sog. Splittingmaß von Elementen durch geringe Toleranzen, unterstützt die Heterogenität der Cluster untereinander.

Der Unterschied zwischen dem Vergleichsmaß und dem Splittingmaß besteht in der Aufgabenstellung. Während das Vergleichmaß zur Herstellung einer Ordnung dient und die Anforderungen der Reflexivität, Antisymmetrie, Transitivität und Totalität gegeben sein müssen, muss ein Splittingmaß nur mindestens die Anforderungen der Reflexivität und Antisymmetrie erfüllen. Die Minimierung der Anforderungen ist mit dem Prozess dieses Verfahrens begründet. Durch die Sortierung liegt bereits eine totale Ordnung vor, so dass ein Splittingmaß diese in diesem Fall nicht gewährleisten muss.

Die Umsetzung des Verfahrens in Form des Algorithmus des Bootstrappings ist in Kapitel 4.7.1 skizziert. Kapitel 4.8 diskutiert unter anderem die Implementierung des modifizierten Algorithmus bzw. Sortierungsmaßes, das auch Unsicherheiten mit Hilfe des Vergleichsmaß in die Entscheidungsfindung integrieren kann.

Insgesamt lässt sich die Bootstrapping Methodik auch austauschen, z.B. durch bereits bekannte manuelle erstellte Cluster, durch Cluster aus einem Goldstandard oder durch einen anderen domänenspezifischen Algorithmus ersetzen. Das hier vorgestellte Bootstrapping wurde auf Basis des Anwendungsfalls der SDSS Spektren entwickelt.

Die Anforderung dieses Bootstrappings an eine randomisierte Selektion einer Untermenge aus allen Daten, ist eine anwendungsspezifisch zu lösende Fragestellung und kann nicht allgemein beantwortet werden. Prinzipiell sollte durch eine entsprechend große randomisierte Untermenge sichergestellt werden, dass alle möglichen Cluster abgedeckt sind, was möglich ist sobald eine Abschätzung $n >> k$ (k ist die Anzahl aller finalen Cluster) getroffen werden kann.

4.6.1 Spektrenspezifisches Bootstrapping

Die Bootstrapping Methodik kann im Fall der vorverarbeiteten SDSS-Spektren weiter optimiert werden. Die Optimierung wird ausschließlich im Bootstrapping-Vorgang eingesetzt. Die identischen Eigenschaften der einzelnen Dimensionen der SDSS Spektren (die Intensitäten bzgl. einer Wellenlänge) ermöglichen es, die Dimensionen miteinander zu diskretisieren. Eine Kontraindikation ist der Fall, sobald zwei Dimensionen unterschiedliche Wertebereiche, Wertetypen oder Bedeutungen besitzen.

In den vorherigen Beschreibungen der Methodik wurde ausschließlich die Werteachse diskretisiert, jedoch nicht die Dimensionsachse gebunden mit Hilfe eines Binningverfahrens. Durch das horizontale Binning werden mehrere Dimensionen zu sog. Bins zusammengefasst. Im Falle der SDSS Spektren werden zusätzlich die Unsicherheiten in den Binningprozess integriert, wodurch dieses Binning einem Resampling Verfahren gleicht. Die Integration der Unsicherheiten in Bins ist ein notwendiger Schritt, da diese nicht gebunden werden können und die Unsicherheiten von Dimension zu Dimension (stark) variieren können. Ein Binning der Unsicherheiten ist nicht möglich, da die Dimensionen mit hohen Unsicherheiten in einem Bin, Dimensionen mit niedrigen oder ohne Unsicherheiten beeinflussen und zusätzlich sich keine Rückschlüsse auf die Originaldaten mehr ziehen lassen.

Das Binning im Allgemeinen kann mit Hilfe eines Faktors, dem sog. Binningfaktor $b \in \mathbb{R}$ gesteuert werden. Dieser Faktor beschreibt, wie viele Dimensionen (auch anteilig) miteinander gebunden werden. Im Falle eines nicht vollständigen Bins durch eine Unterrepräsentation der vorhandenen Dimensionen wird dieses Bin nicht in dem jeweiligen Spektrum zugewiesen.

Das Binning erstellt eine neue Matrix

$$A_{bin} : n_{sub}xk_{max}, k_{max} \in \mathbb{N}, k_{max} = \lceil d/b \rceil$$

der o.g. Matrix $A : n_{sub}xd$ folgendermaßen. Eine Zeile der Matrix a_i wird durch eine Abbildung $bin : \mathbb{R}^d \to \mathbb{R}^{k_{max}}$ gebunden, a_{ij} entspricht dabei der j-ten Dimension eines d_i.

$$\mathbf{binned}_{ik} = bin(a_i, k) = \frac{\sum_{j=k*b}^{k*b+b}(\mathbf{factor}_j * \mathbf{a}_{ij} * \mathbf{ivar}_{ij})}{d \mod b} \tag{4.13}$$

$BINNED$ ist die gebundene Matrix mit einer k-ten Komponente/Dimension einer i-ten Zeile $\mathbf{binned}_{ik}$, wobei $1 \leq k \leq \lceil d/b \rceil$. Die Matrix inversen Varianzen der Spektren $IVAR : nxd$ wird ebenfalls gebunden:

$$\mathbf{ivar\text{-}binned}_{ik} = bin(ivar_i, k) = \sum_{j=k*b}^{k*b+b}(ivar_{ij}) \tag{4.14}$$

$\mathbf{ivar\text{-}binned}_{ik}$ ist analog zu $\mathbf{binned}_{ik}$, die Komponente der gebundenen inverse Varianz des eigentlichen Datenwerts der $BINNED$-Matrix. Die Gleichungen 4.13, 4.14 stellen ebenfalls das gewichtete Mittel der Werte aus den Definitionen der Gleichungen 4.9, 4.10 dar.

Ein Binning mit konstantem Binningfaktor ist ein verbreitetes Verfahren in verschiedenen Anwendungsfeldern, z.B. in [84]. Es ist möglich, das beschriebene Verfahren rekursiv mit dabei variierendem Binningfaktor auszuführen. Je größer ein Binningfaktor ist, desto weniger Dimensionen stehen beim Bootstrapping zur Verfügung, desto zeitlich betrachtet schneller kann eine Bootstrapping Iteration durchgeführt werden. Eine hohe Aggregation der Dimensionen bedeutet jedoch einen hohen Abstraktionsgrad und Informationsverlust. Dennoch können mit Hilfe dieses horizontalen Binnings und einem hohen Abstraktionsgrad gravierende Unterschiede zwischen Spektren differenziert werden.

Durch einen pro Iteration sinkenden Binningfaktor ist es möglich eine hierarchische Struktur zu etablieren. In höheren Rekursionstiefen werden feine Details unterschieden, während sich ansonsten gröbere Strukturen durch die gerine Anzahl an Dimensionen differenzieren lassen. Die Blattebene stellt je nach Abbruchkriterium (z.B. $b = 1$) einen entsprechende Detailgrad an Unterscheidung dar.

Der o.g. Vektor $\mathbf{factor}_k \in [0,1]^{\lceil b \rceil}$ repräsentiert dimensionsabhängige Faktoren, die beschreiben welchen Anteil eine Dimension j aus dem ungebundenen Datensatz in die Dimension k des gebunden Datensatz einbringt. Es sei angemerkt, dass aufgrund von $b \in \mathbb{R}$ die Anzahl der Faktoren gleich $\lceil b \rceil$ sein muss. Die einzelnen Komponenten des Vektors passen sich an den

gegebenen Binningfaktor an, sodass die Summe der Komponenten den Binningfaktor ergibt und jedes Element aus $BINNED$ vollständig in die Berechnungen integriert werden kann. $\mathbf{factor}_k$ variiert für jede Dimension k der gebundenen Daten und wird mit Hilfe der Funktionen $createFactors()$ initialisiert bzw. aktualisiert (siehe Kapitel 4.7.1). Prinzipiell ist es möglich, die beschriebene Optimierung auch für andere Anwendungsfelder zu nutzen.

4.6.2 Splitting Maße

Es wurden verschiedene Splitting Maße entwickelt, die mit Hilfe unterschiedlicher Basisinformationen die jeweiligen Untermengen voneinander trennen. Der Vergleich zweier aufeinander folgender Dateninstanzen entscheidet, ob eine weitere neue Untermenge ab dem j-ten Element erstellt wird.

Absolutes Splitting

Im einfachen Fall des absoluten Splittings ist die Entscheidung über eine Aufteilung der Liste bei zwei aufeinanderfolgenden Elementen (Split) abhängig von einem Unähnlichkeitswert. Dieser Wert wird anhand zweier Spektren $\mathbf{di}_i, \mathbf{di}_j$ verglichen und wie folgt berechnet.

$$dissim_{ij} = \sum_{m=1}^{d} \left\lfloor 1 - \frac{|di_{im} - di_{jm}|}{max(di_{im}, di_{jm})} \right\rfloor$$

$dissim_{ij}$ stellt die Summe der ungleichen Werte eines komponentenweisen Vergleichs von $\mathbf{di}_i, \mathbf{di}_j$ dar und $dissim_{rel_{ij}} = dissim_{ij}/d$ entspricht der relativen Darstellung bzgl. der Dimensionen. Bei einer hohen Anzahl von Dimensionen ist es unwahrscheinlich, dass trotz Diskretisierung alle Dimensionen komponentenweise identische Werte aufweisen. Daher existiert ein reeller Schwellenwert $\Theta_{dissim} \in [0, 1]$, der anhand des relativen Unähnlichkeitswerts entscheidet, ob die Untermengen aufgeteilt werden. Nur im Fall $dissim_{rel_{ij}} \geq \Theta_{dissim}$ wird eine Aufteilung vorgenommen. Der Schwellenwert Θ_{dissim} hat die zusätzliche Funktion, Unsicherheiten einzelner Dimension global abzudecken. Mit der prozentualen Definition des Schwellenwerts kann definiert werden, wie gravierend der Anteil an abweichenden bzw. stark unsicheren Dimensionen ist.

Toleranzbasiertes Splitting

Das toleranzbasierte Splitting erweitert das absolute Splitting um eine weitere Toleranz, die für den Vergleich der Dimensionen der Dateninstanzen $\mathbf{di}_i, \mathbf{di}_j$ eingesetzt wird. Mit Hilfe der zusätzlichen Toleranz besteht die Möglichkeit einzelne Dimensionen variieren zu lassen, ohne jedoch die Fehlerto-

leranz für den gesamten Datenvergleich zu tangieren.

$$dissimtol_{rel_{ij}} = \frac{\sum_{m=1}^{k} tol(di_{im}, di_{im})}{d}$$

$$tol(di_{im}, di_{im}) = \begin{cases} 1 & \text{wenn } |di_{im} - di_{jm}| \leq \mathbf{tol}_m \\ 0 & \text{ansonsten} \end{cases}$$

Im Unterschied zum absoluten Splitting existiert im paarweisen Vergleich nur genau dann ein Unterschied, falls $|di_{im} - di_{jm}| \geq \mathbf{tol}_m, \mathbf{tol} \in \mathbb{R}^d$ ist. **tol** repräsentiert einen Vektor mit manuell definierten dimensionabhängigen Varianzen aus dem Wertebereich der Daten. Unsicherheiten können durch die pro Dimension gegebene Toleranz abgedeckt werden.

4.7 Algorithmus

Dieses Kapitel beschreibt die algorithmischen Aspekte der Fractal Similarity Measures. Die vorherigen Beschreibungen skizzieren die formalen Aspekte und die theoretischen Hintergründe der Methodik, während dieses Kapitel den Einsatz der diskutierten Maße in den dazu speziell entwickelten Algorithmen beschreibt.

4.7.1 Bootstrapping Step

Der Bootstrapping Algorithmus arbeitet prinzipiell rekursiv. Der Aufbau der eigentlich rekursiven Idee ist aufgrund beliebig möglicher Rekursionstiefe bzw. zur Vereinfachung einer späteren nebenläufigen Implementierung iterativ mit Hilfe eines FIFO-Puffers gelöst. Sowohl der Bootstrapping Algorithmus als auch der Performance Algorithmus verwenden die selben Datentypen im Hintergrund, womit der Performance Algorithmus die Ergebnisse bzw. Daten aus dem u.g. Verfahren unverändert übernehmen kann. Voraussetzung für den Start des Bootstrapping Algorithmus ist eine bereits randomisiert selektierte Untermenge des Gesamtdatensatzes.

Zur Initialisierung erstellt das in Algorithmus 4.1 beschriebene Verfahren eine Liste ALL, die direkt mit einem Cluster befüllt wird. Dieses Cluster beinhaltet alle Spektren der selektierten Untermenge (siehe Zeile 5). Diese Liste wird pro Iteration abgearbeitet und einzelne Cluster C_i werden folgendermaßen auf eine entsprechende fortführende Aufteilung geprüft.

Die Spektren aus dem aktuell zu bearbeitenden Cluster C_i werden zuerst gebunden (siehe Kapitel 4.6.1), diskretisiert (s. Kapitel 4.3) und anschließend sortiert (Linie 9). Nach der Sortierung werden die Cluster-Mitglieder sequentiell der Reihenfolge nach prozessiert, wobei ein Splitting Maß immer zwei aufeinander folgende Dateninstanzen miteinander vergleicht. Eine weitere leer initialisierte Liste $CLUSTERS_{new}$ steht zur Verfügung um die neu erstellten Cluster der aktuellen Iteration zu speichern.

Existiert z.B. die sortierte Reihenfolge der Dateninstanzen $[M_a, M_b, M_c]$, muss das Maß die Paare M_a, M_b, M_b, M_c vergleichen. Dabei entscheidet das Maß ob die Paarungen ähnlich oder nicht ähnlich sind. Im Algorithmus ist der Fall des absoluten Splittings dargestellt (Linie 13).

Im positiven Fall wird die Menge zwischen den korrespondierenden Elementen in eine eigene Untermenge getrennt (Linien 14, 15). Falls z.B. M_a nicht ähnlich zu M_b sein sollte, so wird eine Trennung so vollzogen, dass sich M_a (und alle davor befindlichen ähnlichen Elemente) in einer Untermenge sowie die nach M_b folgenden Elemente (inkl. M_b) in einer anderen Untermenge befinden. In der Sortiermethode sowie während des Vergleichens bzw. im Splitting Maß werden fehlende Werte und Werte mit hohen Unsicherheiten entsprechend beachtet und mit dem durchschnittlichen Cluster Spektrum gemäß Kapitel 4.5 ersetzt.

Die ggfs. neu erstellten Cluster werden auf eine Mindestgröße ($minSize$) geprüft. Im Pseudocode ist $minSize = 10, binningfactor = d/2$ gesetzt. Ein größerer Binningfaktor ist nicht sinnvoll, da dadurch effektiv nur eine Dimension zur Analyse zur Verfügung steht, die sich nicht gravierend von einem 2D Binning unterscheidet. Die Variation des $minSize$ Parameters ist möglich. Eine Erhöhung dieses Parameters verursacht eine Minimierung der Anzahl der Cluster in ALL, da vermehrt Cluster als zu klein und damit in die $SMALL$ Liste wandern. Analog dazu verhält sich die Minimierung des Parameterwerts.

Im Falle einer entsprechenden Unterrepräsentation von Elementen in einem Cluster wird das Cluster der Liste der neuen Cluster entfernt bzw. in eine Liste zu kleiner Cluster ($SMALL$) hinzugefügt (Linie 22). Am Ende eines Bootstrapping Vorgangs werden die zu kleinen Cluster aufgeteilt und deren Elemente wie initiale Elemente behandelt. Dadurch ist sichergestellt, dass alle Cluster eine minimale Größe und damit eine entsprechende Anzahl an Elementen zur Formung des hochdimensionalen Körpers aufweisen. Eine Iteration ist mit dem Halbieren des Binningfaktors abgeschlossen. Abbruchbedingungen für alle Iterationen ist das Erreichen des Binningfaktors $b = 1$ womit sichergestellt ist, dass auch kleinere Unterschiede bzgl. der Boxgröße berücksichtigt werden. Theoretisch können weitere Abbruchbedingungen sinnvoll sein:

- Falls die Boxgröße sehr klein im Verhältnis zum kleinsten Datenwert aller Dimensionen aller Elemente gewählt wurde, kann die Anzahl der zu kleinen Cluster $||SMALL||$ aufgrund der unterschiedlichen Diskretisierungen schnell wachsen. Ist die Anzahl der zu kleinen Cluster um einen definierbaren Faktor größer als die Anzahl der normalen Cluster, ist ein Abbruch sinnvoll. Dieser kann mit Hilfe einer entsprechenden Bedingung sowie dem Hinzufügen des aktuellen Clusters C_i in die $CLUSTERS_{new}$ Liste umgesetzt werden. Die Implementierung des Kriteriums kann nach den For-Schleifen in Zeile 26 stattfinden. Das

Kriterium muss die zu kleinen Cluster in $CLUSTERS_{new}$ eines vorherigen Clusters C_i durch C_i ersetzen.

- Falls aus z.B. Performancegründen nur eine maximale Anzahl an Clustern erlaubt sein sollte, kann dies auch in Zeile 26 umgesetzt werden. Dieses Kriterium würde die Liste der aktuellen Cluster ALL nicht ersetzen.

Die genannten weiteren Abbruchbedingungen bzw. Optimierungen sind zur Wahrung der Übersicht nicht im Pseudocode dargestellt. In anderen Anwendungsfeldern sind noch weitere Abbruchkriterien denkbar, die z.B. die Qualität der Cluster als Entscheidungsbasis für einen Abbruch nutzen. Cluster die durch Algorithmus 4.1 entstanden sind, stellen die Datenbasis für den Performance Algorithmus dar.

4.7.2 Spektrenspezifisches Bootstrapping

Im Fall der SDSS Spektren findet das Binning mit Hilfe der *bin*-Funktion Anwendung statt. Falls der Algorithmus in einer anderen Domäne verwendet wird, ist der Einsatz des Binnings zu prüfen, da sich die Charakteristika der einzelnen Dimensionen unterscheiden können und ein Binning ausschließen. Dennoch kann der Bootstrapping Algorithmus ohne das Binning in einem solchen Fall eingesetzt werden.

Beim spektrenspezifischen Bootstrapping inklusive Binning existiert der beschriebene Vektor $\mathbf{faktor}_k$, der mit folgender Methodik berechnet wird. Der Vektor wird mit der $createFactors$ Funktion, beschrieben im Algorithmus 4.2, erstellt und aktualisiert. Es sei angemerkt, dass einzelne Dimensionen sich auch auf mehrere Komponenten mehrerer Vektoren aufteilen können, da Dimensionen vollständig aufgeteilt werden. Im primären Aufruf der Funktion $createFactors$ dient der Parameter $\mathbf{factors}_0 \in [0]^d$ zur Initialisierung und Generierung des ersten Vektors. Die Initialisierung bewirkt, das alle Komponenten des ersten Vektors mit 1 gefüllt sind, mit Ausnahme der letzten Komponente die bei einem $b \notin \mathbb{N}$ mit $b \mod (||factor_k|| - 1)$ gefüllt ist.

Die Neuberechnung der Faktoren muss nach jeder Iteration erfolgen und adaptiert sich anhand des vorherigen Vektors ($factors_{k-1}$) so an, dass alle Werte vollständig in das Binning einfließen. Der Algorithmus stellt sicher, dass $\sum_{m=1}^{||factor_k||} factor_{km} = b$. Ein einzelner Faktor hängt immer von den vorherigen Faktoren ab. Der erste Faktor eines Vektors hängt dabei von der letzten Komponente des vorherigen Vektors ab und übernimmt ggfs. den verbleibenden Anteil eines Wertes (siehe Zeile 7). Im Normalfall faktorisiert eine Komponente des $factors$ Vektor in Zeile 12 eine gesamte Dimension für $0 < j < ||\mathbf{factors}||$. Die letzte Komponente des Vektors berechnet sich aus der Differenz des Binningfaktors und den summierten Komponenten des Vektors.

Algorithmus 4.1: Bootstrapping Algorithmus

```
1: binningfactor = d/2
2: minSize = 10
3: ALL = {}
4: SMALL = {}
5: ALL.add(new cluster(allSpectra))
6: repeat
7:     CLUSTERS_new = {}
8:     for each C_i ∈ ALL do
9:         binnedSpecs = sort(grid(bin(C_i, binningfactor), boxsize))
10:        cluster_new = new cluster()
11:        for each DI_j ∈ binnedSpecs do
12:            dissim = dissim(DI_j, DI_{j-1})
13:            if dissim > Θ_dissim then
14:                CLUSTERS_new.add(cluster_new)
15:                cluster_new = new cluster()
16:            end if
17:            cluster_new.add(DI_j)
18:        end for
19:        for each cluster_new ∈ CLUSTERS_new do
20:            if cluster_new.size() < minSize then
21:                CLUSTERS_new.remove(cluster_new)
22:                SMALL.add(cluster_new)
23:            end if
24:        end for
25:    end for
26:    ALL = CLUSTERS_new
27:    binningfactor = binningfactor * 0.5
28: until binningfactor < 1
29: perfStep(CLUSTERS_new, SMALL)
```

4.7.3 Performance Step

Der zweite zentrale Algorithmus in der gesamten Methodik ist der Performance Step Algorithmus. Mittels einer relativ zum Bootstrapping geringen Komplexität ist es mit dem Algorithmus möglich, einzelne Dateninstanzen einem Cluster zuzuweisen und darin effizient zu integrieren. Durch testweises Hinzufügen einzelner Dateninstanzen in alle Cluster unter Einsatz des FSM werden die einzelnen Änderungen an den Clustern quantifiziert. Das Cluster mit der daraus resultierenden geringsten Änderung entspricht dem besten passenden Cluster bzgl. des FSM. Als finalen Schritt integriert der Algorithmus die Dateninstanz in das passende Cluster.

Der Algorithmus beginnt mit der Initialisierung einer Liste für Ausreißer.

Algorithmus 4.2: *createFactors*-Funktion

```
1:  numberOfFactors = Math.ceil(binningFactor)
2:  factors_k = newVector(numberOfFactors)
3:  sumOfFactors = 0
4:  for j = 0; j < factors.length; j = j + 1 do
5:      if j == 0 then
6:          if factors[factors.length − 1] ≠ 1 then
7:              factors[j] = 1 − factors_{k−1}[factors.length − 1]
8:          else
9:              factors[j] = 1
10:         end if
11:     else if binningFactor − sumOfFactors > 1 then
12:         factors[j] = 1
13:     else
14:         factors[j] = binningFactor − sumOfFactors
15:     end if
16:     sumOfFactors = factors[j]
17: end for
```

Die gegebene Liste der zu clusternden Dateninstanzen (DI) wird iterativ abgearbeitet, wobei der Performance Step jedes Element $\mathbf{di}_j$ folgendermaßen prozessiert.

Für jedes Cluster C_i wird ein normierte FSM (siehe Gleichungen 4.7, 4.8) berechnet, die das Element beinhaltet. Das jeweilige Cluster ist nach der Berechnung des FSM unverändert im Vergleich zum Zustand vor dem Einfügen. Eine, für jedes $\mathbf{di}_j$ neu initialisierte, Liste $FSMs_{norm}$ sichert den FSM Wert inklusive einer Zuordnung zum dazugehörigen Cluster. Die Normierung findet auf Basis der Clustergröße inklusive des Elements statt. Nachdem alle Cluster prozessiert wurden, wird die Liste $FSMs_{norm}$ aufsteigend sortiert, damit die kleinsten Änderungen sich am Anfang der Liste befinden, womit in $FSMs_{norm}[0]$ der Wert des am besten passenden Clusters darstellt. Zeile 9 des Algorithmus prüft mit Hilfe des relativen Vergleichs die Signifikanz der Entscheidung. Unterscheiden sich die FSM Werte der zwei am besten passenden Cluster nicht signifikant ($FSM_{rel} < \theta$), kann ein passendes Clustering nicht sichergestellt werden und $\mathbf{di}_j$ wird als Ausreißer klassifiziert. Andernfalls wird das Element in Zeile 12 permanent zum Cluster hinzugefügt.

Durch eine Adaption eines Clusters durch Hinzufügen eines Elements kann der hochdimensionale Körper eines Clusters angepasst werden, da dieser ggfs. in einer oder mehreren Dimensionen verändert wird. Die Integration eines Elements in ein Cluster stellt jeweils die beste zur Laufzeit zur Verfügung stehende Möglichkeit dar. Permutierte Reihenfolgen der selben Daten in der Liste DI können theoretisch bei zusätzlich gegebenen Ähnlichkeit

Algorithmus 4.3: Performance Step Algorithmus

1: $Outliers = \{\}$
2: **for** each $\mathbf{di}_j \in DI$ **do**
3: $FSMs_{norm} = \{\}$
4: **for** each $C_i \in C$ **do**
5: $FSM_{norm} = FSM(C_i, \mathbf{di}_j) * ||(C_i \cup \mathbf{di}_j)||$
6: **add** FSM_{norm} **to** $FSMs_{norm}$
7: **end for**
8: **sort** $FSMs_{norm}$ **ascending**
9: **if** $\frac{FSMs_{norm}[0]}{FSMs_{norm}[1]} > \theta$ **then**
10: **add** $\mathbf{di}_j$ **to** $Outliers$
11: **else**
12: **add** $\mathbf{di}_j$ **to** cluster with value $FSMs_{norm}[0]$
13: **end if**
14: **end for**

zwischen Clustern in unterschiedlichen Ergebnissen resultieren. Ergebnisunterschiede bestehen in diesem Spezialfall aus Zuweisungen des gleichen Elements zu diversen Clustern. Zeile 9 minimiert diesen Effekt und erstellt damit eine Invarianz bzgl. der Einfügereihenfolge. Die Tests mit Permutationen der Reihenfolge gleicher synthetischer und reeller Daten beschrieben in Kapitel 5 ergeben, dass die Reihenfolge des Hinzufügens keinen Unterschied in den Ergebnissen des Clusterings macht. Die in *Outlier* definierten Ausreißer können nach dem Prozessieren aller zuzuweisenden Dateninstanzen solange iterativ zugewiesen werden, bis die Liste konstant bleibt.

4.7.4 Komplexitätsbetrachtung

Da die Komplexität der gesamten Methodik nicht ausschließlich von einem der Algorithmen abhängig ist, wird diese Thematik in einem eigenen Kapitel behandelt. Die Komplexität der Algorithmen lässt sich in eine Speicherkomplexität sowie eine Berechnungskomplexität unterteilen. Nachstehend werden die beide Arten von Komplexitäten pro Algorithmus diskutiert.

Bootstrapping Algorithmus

Die Berechnungskomplexität des Bootstrapping Algorithmus kann aufgrund dessen Verschachtelung nicht einfach abgelesen werden. Der Algorithmus 4.1 teilt sich bzgl. der Komplexität in folgende Teile auf. In der Aufteilung wird von einer durchschnittlichen Clustergröße von $||DI_{C_i}|| = n/k$ ausgegangen.

- Die äußerste Schleife ist für die Veränderung des Binningfaktors verantwortlich. Je nach Binningfaktor bzw. Anpassungsfaktor in Zeile 27 können unterschiedliche Laufzeiten entstehen. Eine Minimierung die-

ses Anpassungsfaktors (aktueller Faktor ist 0.5) senkt entsprechend die Laufzeiten, da der Binningfaktor schneller das Minima von 1 erreicht. Durch den aktuellen Faktor wird die Schleife $\log d$ mal ausgeführt.

- Insgesamt iteriert die Schleife ab Zeile 8 über alle verfügbaren Cluster und endet nach genau k-Iterationen.
- Der im Algorithmus teuerste Anteil an der Komplexität ist die Schachtelung der Funktionen zum Sortieren, Binden und Diskretisieren. Durch eine optimierte Implementierung kann die Komplexität reduziert werden, was der Komplexität des Sortieralgorithmus entspricht. Die aktuelle Implementierung verwendet TimSort [65]. Die Operationen können so in der Implementierung integriert werden, dass keine zusätzlichen Laufzeitfaktoren hinzukommen. Die Komplexität in diesem Anteil ist $n/k \log n/k$.
- Cluster entstehen im Bootstrapping Algorithmus durch die Aufteilung der sortierten Liste. Die Aufteilung entsteht durch iteratives abarbeiten der Liste *binnedSpecs*. Insofern wird diese Operation genau n/k-mal ausgeführt.
- Das Aussortieren von Clustern mit zu geringer Anzahl an Elementen (siehe Zeile 21) ist ausschließlich abhängig von der Anzahl der Cluster und wird genau k-mal prozessiert.

Insgesamt setzen sich die Komplexitäten zusammen:

$$\log dk(\frac{n}{k} \log \frac{n}{k} + \frac{n}{k} d) = \log d(n \log \frac{n}{k} + nd).$$

Die O-Notation zeigt immer den schlechtesten Fall bzw. das Maximum der Laufzeit des Algorithmus. Für die Notation zählt nur der größte Faktor sowie abhängige Konstanten [53, 443 ff.]. Die Komplexität des Bootstrapping Algorithmus entspricht insgesamt:

$$O(n \log n/k \log d + \log dnd).$$

Die Notation zeigt, dass dieser Teil der gesamten Methodik von der Anzahl der Dimensionen logarithmisch abhängt und eine Maximierung der Dimensionen nur einen geringen Einfluss auf die Effizienz des Algorithmus hat. Die Speicherkomplexität dieses Algorithmus ist implementierungsabhängig. Die aktuelle Implementierung hat einen maximalen Speicherverbrauch von $O(ndo)$, was der Größe des Datensatzes multipliziert mit der Anzahl der Boxgrößen plus wenigen Bytes für die entsprechenden Datenstrukturen entspricht.

Performance Step Algorithmus

Ziel bei der Entwicklung des Performance Steps ist eine geringe Komplexität, die nur zu einem geringen Grad von der Dimensionalität abhängt. Dieser Teil

der Methodik bestehend aus Algorithmus 4.3 setzt sich aus den folgenden Komponenten und Komplexitäten zusammen.

- Maßgeblichen Einfluss auf die Komplexität hat die äußere for-Schleife, beginnend ab Zeile 2. Diese lässt die Methodik alle verfügbaren im Bootstrapping noch nicht zugewiesenen Elemente (die Menge DI) prozessieren und muss daher $n - n/k$ iterieren. Die maximale Anzahl an Iterationen beträgt dennoch n, da es möglich ist, dass bereits Cluster existieren, die unabhängig von DI erstellt wurden.
- Einen weiteren Einfluss auf die Komplexität hat die Schleife über alle Cluster (Zeile 4). Diese ist abhängig von der Anzahl der zur Verfügung stehenden Cluster $||C|| = k$, womit die Anzahl der Iterationen exakt k beträgt.
- Der interessanteste Beitrag zur Komplexität ist in Zeilen 5 hinterlegt. Mit Hilfe einer optimierten Datenstruktur ist es möglich, die Operationen Hinzufügen bzw. Löschen von Elementen in ein Cluster mit einer Komplexität von $O(1)$ zu erledigen. Alternativ erfolgt eine Erweiterung der Komplexität bzgl. n um einen entsprechenden Faktor, der bei einer Listendatenstruktur mit binärer Suche im schlechtesten Fall $\log n$ ist. Des Weiteren ist die Berechnung des FSM bzw. der FIV zu beachten. Pro FSM werden zwei FIV über alle Boxgrößen berechnet, d.h. es existiert eine weitere Abhängigkeit der FSM zu der Anzahl der Boxgrößen ($o = ||G||$). Zusammengefasst ergibt die Kombination der Anzahl der Cluster und der Anzahl der Boxgrößen die Komplexität $O(no)$.
- Einen weiteren kleinen Anteil hat die Sortieroperation in Zeile 8, die beträgt aufgrund der Effizienz von TimSort $k \log k$.

Insgesamt ergibt die Komplexität des Performance Step Algorithmus unter der Annahme, dass $k << n$: $O(nkod)$. Die Speicherungskomplexität des Algorithmus entspricht, durch die Verwendung der verschiedenen Boxgrößen bei einer effizienten Implementierung, ebenfalls $O(ndo)$. Durch die Abhängigkeit bzgl. der Anzahl der Boxgrößen (o) kann eine anwendungsabhängige Balance zwischen dem Speicherverbrauch und der Detailstufe in der Analyse gefunden werden. Eine höhere Detailstufe bedeutet eine größere Anzahl an verwendeten Boxgrößen und damit einen höheren Speicherverbrauch. Analog dazu kann experimentell ermittelt werden, wie viele Boxgrößen notwendig sind, um die Cluster im Performance Step mit Hilfe der FSM korrekt zu repräsentieren um den Speicherverbrauch zu senken.

Gesamtkomplexität der Methodik

Die Gesamtkomplexität der Methodik setzt sich aus den beiden Komplexitäten des Performance Steps und des Bootstrappings zusammen. Durch den Aufbau der Methodik wird zuerst eine Untermenge der Gesamtdaten

$n_{bootstrap} < n$ mit Hilfe des Bootstrappings analysiert. n_{Rest} ist die Anzahl der Elemente die nicht mit dem Bootstrapping prozessiert bzw. durch das Bootstrapping als Ausreißer deklariert wurden. Die Gesamtkomplexität entspricht $O(n_{bootstrap} \log n_{bootstrap}/k \log d + \log d n_{bootstrap} d + n_{Rest} kod)$.

4.8 Implementierung des Verfahrens

Dieses Kapitel beschreibt die Details der Implementierung des Verfahrens. Der Fokus des Kapitels liegt auf den Datenstrukturen sowie den optimierten Umsetzungen der einzelnen für diese Methodik zentralen Algorithmen. Sowohl die Vorverarbeitungsschritte (siehe Kapitel 3.4) als auch die Clustering-Algorithmen sind in Java 8 implementiert. Die Selektion einer einheitlichen Entwicklungssprache für beide Softwareanteile ermöglicht die Implementierung gemeinsam genutzter Datentypen sowie deren einfache Integration.

4.8.1 Zentrale Datentypen

Abbildung 4.7 zeigt ein UML-Klassendiagramm mit den für das Verständnis der Methodik relevanten Datentypen. Die dargestellten Datentypen ergeben die Datenhaltungsbasis (sog. plain old Java objects, POJOs [103]) der Referenzimplementierung des in dieser Arbeit entwickelten Verfahrens. Insgesamt beinhaltet die Referenzimplementierung mit dem Namen *INSPECT-J* 108 Klassen, die sowohl für die Datenhaltung als auch für Datenverarbeitung zuständig sind. Ebenfalls enthalten in dieser Zählung sind die I/O-Klassen zum Einlesen der vorprozessierten Daten bzw. Schreiben der Resultate sowie selbstentwickelte Werkzeuge zur Visualisierung der Analyseergebnisse. Die nachfolgenden Abschnitte beschreiben die einzelnen Datentypen der Datenhaltung. Die Datentypen der Datenhaltung besitzen bis auf die POJO üblichen Getter- bzw. Setter-Methoden keine weiteren Methoden.

Neben den Datentypen für die Datenhaltung existieren weitere Klassen, die gegenüber einem POJO erweiterte Funktionen aufweisen. Sortierbare diskreten Spektren erweitern den Basisdatentyp Spectrum. Diese erweiterten Datentypen sind in den folgenden Unterkapiteln diskutiert.

Spectrum

Das zentrale POJO der Implementierung ist der Datentyp Spectrum. Dieser speichert sowohl in der Vorverarbeitung als auch im Clustering Verfahren alle zur Analyse eines Spektrums notwendigen Daten. Die notwendigen Daten eines Spektrums lagern in einem privaten 2D-float-Array (*values*) und dieses beinhaltet die folgenden Informationen:

- Flux Werte in $values[0]$
- Inverse Varianzen in $values[1]$

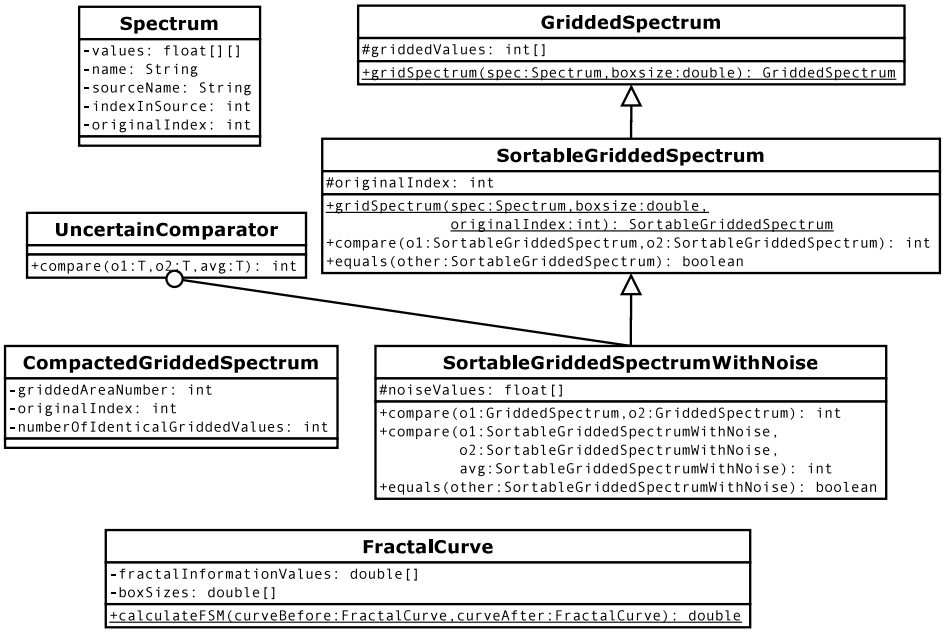


Abbildung 4.7: UML-Klassendiagramm der wichtigsten Datentypen der Datenhaltung

- Zu $values[0]$ and $values[1]$ entsprechende Wellenlänge in $values[2]$

Alle in $values[i]$ befindlichen Unterarrays besitzen genau d Einträge, was der Anzahl der Dimensionen bzw. Messwerte entspricht. Des Weiteren beinhaltetet der Datentyp Metadaten zur Identifizierung der Datenquelle bzw. Quelldatei zum Einlesezeitpunkt ($indexInSourceFile$,$sourceName$) sowie einem zentralen Index $originalIndex$ zur effizienten Referenzierung der Objekte in Listen oder Maps.

Uncertain Comparator

Die generische $UncertainComparator$-Schnittstelle definiert die Möglichkeit einen Datentyp, der Unsicherheiten beinhaltet, zu sortieren. Kapitel 4.5 beschreibt die theoretischen Gedanken der Handhabung unsicherer Daten in diesem Verfahren. Prinzipiell werden Daten, die unsicher sind, durch einen entsprechenden Wert ersetzt. Im Verfahren gelten genau die Daten als unsicher, deren Wert der zugehörigen inversen Varianzen unter bzw. gleich einem bestimmten Schwellwert sind. Eine entsprechende Ersetzung wird im Fall dieser Schnittstelle durch die entsprechende Dimension der avg-Variable vorgenommen. Da eine Sortierung in der Referenzimplementierung immer innerhalb eines Clusters stattfindet, repräsentiert die avg-Variable das durchschnittliche Spektrum des Clusters.

Datentypen sortierbarer diskreter Spektren

Durch die eingesetzten Binning bzw. Diskretisierungsverfahren werden die Float basierten Daten der Spektren zu diskreten Boxen zugewiesen. Die Datentypen der diskreten Spektren teilen sich in in die folgenden Datentypen auf:

- Das sog. *GriddedSpectrum* repräsentiert die Basisform eines diskreten Spektrums und bietet ausschließlich ein Integer-Array (*griddedValues*) zur Speicherung der jeweiligen Boxnummer des diskretisierten Messwerts bzw. der jeweiligen Dimension.
- Eine sortierbare Erweiterung des *GriddedSpectrum* ist das *SortableGriddedSpectrum* und erweitert die Oberklasse durch eine entsprechende Compare-Methode zum Vergleichen. Zur Identifizerung des originalen Spektrums dient das private *originalIndex* Feld.
- Die *SortableGriddedSpectrumWithNoise* Klasse erweitert die sortierbare Variante durch die Handhabung von Unsicherheiten durch die Implementierung der Schnittstelle *UncertainComparator*. Zusätzlich speichert ein Float-Array die entsprechenden, nicht diskretisierten inversen Varianzen des originalen Spektrums.

Die inversen Varianzen werden nicht im Integer Format gespeichert, da diese nicht diskretisiert werden. Inverse Varianzen beschreiben eine Vertrauenswürdigkeit der sowohl der diskretisierten als auch der nicht diskretisierten Flux-Daten. Die Daten werden aus dem originalen Spektrum nur referenziert und nicht kopiert, womit kein signifikanter Aufwand in der Instantiierung eines Objekts entsteht. Dadurch bietet eine Diskretisierung der inversen Varianzen in diesem Kontext keinen Vorteil gegenüber eine Schwellwert-basierten Evaluation während des Sortiervorgangs.

CompactedGriddedSpectrum

Die Berechnung der FIV durch die Gleichungen 4.5, 4.6 wird mit Hilfe des *CompactedGriddedSpectrum* Datentyps optimiert. Innerhalb der Gleichungen sind die boxgrößenabhängigen Populationsfrequenzen $p(BX_s(g))$ zu bestimmen. Eine Berechnung der Populationsfrequenzen ist ohne Komprimierung der zugrunde liegenden Daten ein aufwändiger Vorgang, da geprüft werden muss, welche diskretisierten Spektren identisch bzgl. deren Dimensionen sind. Eine stark komprimierte Version der $(Sortable)GriddedSpectrum(WithNoise)$ Klasse(n) stellt die Klasse $CompactedGriddedSpectrum$ dar. Die kompakte Darstellung beinhaltet pro Spektrum die folgenden Informationen:

- Den originalen Index des Spektrums
- $numberOfIdenticalGriddedValues$ stellt die Anzahl der diskretisierten Spektren dar, die über alle Dimensionen die gleiche Box besetzen.

- Die Variable *griddedAreaNumber* identifiziert eindeutig die Gruppe der diskretisierten Spektren, die über alle Dimensionen die gleiche Box besetzen.

Der pro FIV-Berechnung auftretende Aufwand der Prüfung von identisch diskretisierten Spektren kann damit pro Vergleich zweier Spektren von $O(d)$ als Worst-Case auf insgesamt exakt $O(1)$ minimiert werden, da durch die Komprimierung der Daten kein Vergleich der Dimensionen notwendig ist. Anstatt eines Vergleichs verschiedener diskreter Spektren beinhaltet die Datenstruktur bereits die notwendigen Daten zur Berechnung der jeweiligen $p(BX_s(g))$, die aus einer Gruppe bzw. der Anzahl der identischen Einträge dieser Gruppe besteht. Das folgende Kapitel 4.8.2 beschreibt den implementierten Algorithmus zur Erstellung der *CompactedGriddedSpectrum*-Objekte.

Fractal Curve

Ein weiteres Konzept in der Datenhaltung ist die *Fractal Curve*. Diese stellt die Datenhaltung für die Speicherung von Fraktalen Informationswerten zur Verfügung und wird zur Berechnung des Fraktalen Ähnlichkeitsmaßes verwendet. Die Klasse besteht aus zwei kongruenten Arrays, die für eine bestimmte Boxgröße einen berechneten FIV bereitstellen. Mit Hilfe der Klasse *Fractal Curves* ist es möglich andere von FIV-abhängige Metriken zu entwickeln bzw. effizient zu implementieren. Darüber hinaus beinhaltet diese Klasse die Referenzimplementierungen in Form von statischen Methoden zur Berechnung der FSM-Werte durch den Einsatz zweier *Fractal Curves*.

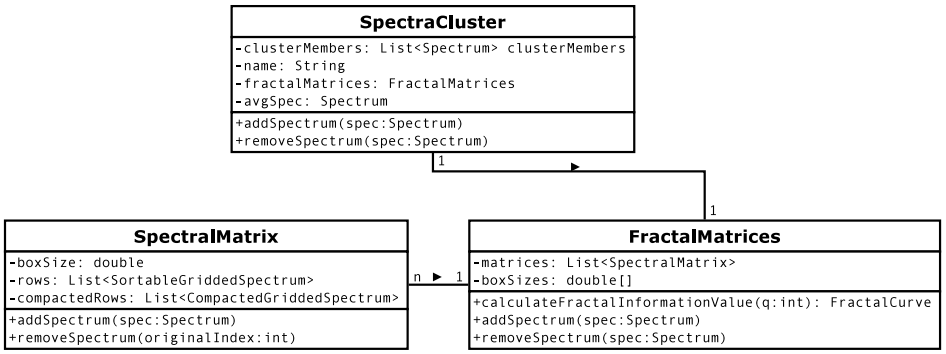


Abbildung 4.8: UML-Klassendiagramm der wichtigsten Klassen der Datenverarbeitung

SpectraCluster

SpectraCluster Objekte bestehen aus der Liste aller Spektren Objekte (*clusterMembers*) sowie einem beschreibenden Namen in Form eines Strings

(*name*) und den fraktalen Matrizen $fractalMatrices$. Jedes Cluster-Objekt beinhaltet alle Spektren des jeweiligen Clusters. Da Cluster in dieser Methodik ausschließlich durch ihre eigenen Spektren bzw. durch ein durchschnittliches Spektrum konstituiert sind und ein Spektrum nur zu exakt einem Cluster zugewiesen wird, entsteht durch diese Art der Datendarstellung kein Overhead bzgl. der Speicherungsvolumen.

Neben den eigentlichen Daten der Spektren enthalten SpectraCluster Objekte eine Repräsentation des Clusters in Form eines durchschnittlichen Cluster-Spektrums *avgSpec*, das durch die Operationen *add* und *remove* invalidiert wird bzw. mit der entsprechenden Getter-Methode cacheoptimiert einmalig berechnet und zurückgegeben wird. Die theoretische Grundlage der Berechnung der Repräsentation diskutiert Kapitel 4.5 und ist in einer privaten Methode innerhalb der Klasse gekapselt.

SpectralMatrix

Die Klasse SpectralMatrix aggregiert die zentralen Datentypen eines Clusters bzgl. einer bestimmen Boxgröße zu einer virtuellen Matrix. Für jede aktive Boxgröße existiert pro Cluster ein SpectralMatrix Objekt im Arbeitsspeicher, das alle diskretisierten Werte der jeweiligen Spektren ($rows$), sowie deren komprimierte Darstellung ($compactedRows$) permanent beinhaltet. Der Speicherverbrauch wird durch die dauerhafte Speicherung der diskretisierten Werte der Spektren erhöht. Der Zugriff auf die o.g. Listen ist im Hintergrund durch Integer-basierten Hashmaps optimiert, sodass der Zugriff auf einzelne diskretisierte Spektren mit Hilfe des $originalIndex$ immer mit der konstanten Komplexität von $O(1)$ erfolgen kann.

Der erhöhte Speicherverbrauch bei einer großen Anzahl von Objekten kann durch aktuelle Entwicklungen im Hard- als auch im Softwarebereich zugunsten einer minimierten Prozessdauer kompensiert werden. Die dauerhaft vorgehaltenen diskretisierten und komprimierten Daten aller Cluster erlauben eine performante Einsortierung neuer bzw. eine performante Löschung der Spektren durch eine möglichst geringe Berechnungskomplexität und damit auch minimierter Laufzeit. Zum Zeitpunkt der Entwicklung dieser Software existieren bereits kommerziell verfügbare Server mit bis zu 12 TB RAM, womit auch mehrere Millionen Spektren mit einer großen Zahl an Boxgrößen ohne Swapping-Effekt im Arbeitsspeicher gehalten werden können. Des Weiteren existieren Methodiken zur partitionierten verteilen globalen Organisation von Speichern, z.B. der sog. Global Address Space Programming Interface (GASPI) [6] oder Open Shared Memory (OpenSHMEM) [25].

FractalMatrices

Die Klasse FractalMatrices kapselt alle Operationen, die für das Verfahren notwendig sind. Ein Objekt dieser Klasse beinhaltet alle SpectralMatrix Ob-

jekte eines Clusters in Form einer Liste *matrices* und alle eingesetzten Boxgrößen zur Berechnung der diskreten Repräsentationen. Die bereitgestellten Methoden werden hauptsächlich von Objekten der Klasse SpectraCluster genutzt und initiieren die jeweiligen Detailoperationen (*add*, *remove*) auf den SpectraMatrix-Objekten. Zusätzlich kann mit Hilfe der entsprechenden Methode die Berechnung einer FractalCurve mit aktuellen Satz an Spektren erfolgen.

4.8.2 Berechnung der komprimierten Spektren

Die beschriebenen Komplexitäten der jeweiligen Algorithmen sind abhängig von der Implementierung einzelner Operationen. Die folgenden Abschnitte stellen die wichtigsten implementierungsspezifischen Prozesse des Verfahrens dar, die einen essentiellen Beitrag zur Komplexität bzw. Funktionalität des Verfahrens beitragen.

Die Berechnung der komprimierten Datenrepräsentationen ist abhängig von der jeweiligen Boxgröße bzw. des SpectralMatrix-Objekts. Diese Berechnungen finden ausschließlich zum Zeitpunkt der Erstellung einer Matrix, d.h. beim Erstellen eines Clusters oder bei einer Manipulation des Clusters statt. Ursachen unabhängig erfolgen diese Berechnung mit Hilfe des folgend diskutierten Musters:

1. Zuerst müssen alle zu berechneten Spektren eines Cluster mit Hilfe der Boxgröße bzw. dem aktuellen Binningfaktor entsprechend diskretisiert werden. Ein Aufruf der Methode *gridSpectrum* in der Klassenhierarchie des *GriddedSpectrum* ermöglicht eine nebenläufige Berechnung aller zu diskretisierenden Spektren unter Voraussetzung eines bereits erfolgten homogenen Binnings. Zusätzlich zur aus den *gridSpectrum*-Aufrufen entstandenen *rows*-Liste stellt eine Integer-basierte HashMap die diskretisierten Spektren für einen effizienten Zugriff zur Verfügung. Die HashMap erzeugt eine Abbildung von einem original Index zu einem diskretisierten Spektrum.
2. Der modifizierte TimSort Algorithmus sortiert die *rows*-Liste unter Berücksichtigung der Unsicherheiten in den Datensätzen. Der Sortiervorgang wird ausschließlich zum Zeitpunkt des Erstellens einer Matrix verwendet. Anschließende Manipulationen des Clusters erfolgen durch ein optimiertes Einsortierungsverfahren.
3. In der sortierten Liste untersucht ein Vergleichsmaß linear die aufeinanderfolgenden Spektren und erstellt für jedes Spektrum ein *CompactedGriddedSpectrum*-Objekt. Dabei wird eine neue Gruppe bzw. eine neue Gruppennummer (*griddedAreaNumber*) von zusammenhängenden Spektren erstellt, sobald zwei aufeinanderfolgende bzgl. der identischen Boxgröße diskretisierten Spektren nicht identische Diskretisierungswerte aufweisen können. Analog zum eigentlichen Sortierungsver-

fahren integriert die Vergleichsmethode das durchschnittliche Cluster-Spektrum um eventuell größere Unsicherheiten zu kompensieren.

Aus den o.g. Schritten sind zwei sortierte Listen entstanden, die mittels der Sortierreihenfolge eine boxgrößenabhängige interne Abstufung bzw. Subclustering der Daten darstellen. Die Sortierung organisiert die Daten in einer Art und Weise, die ähnliche Daten innerhalb der Ordnung nahe zueinander lokalisiert. Das bedeutet zwei aufeinander folgende Einträge können, müssen aber nicht (sehr) ähnlich sein, da auch beliebige Lücken in einzelnen oder mehreren Dimensionen bestehen können. Durch die mehrfache Berechnung dieser Listen unter Einbezug verschiedener Boxgrößen ist die Analysebasis zur effizienten Berechnung der FIV bzw. FSM gegeben.

Die Beschreibung der Vorgehensweise zeigt, dass eine permanente Rekalkulation der Matrizen keine effiziente Alternative zur dauerhaften Speicherung der Daten darstellt, da die Laufzeiten bzw. Komplexitäten einzelner Manipulation signifikant gesteigert werden würden. Während bei einer permanenten Speicherung der Daten die Manipulation eine Berechnungskomplexität $O(1)$ hat, resultiert eine permanente Rekalkulation in einer Komplexität von $O(n \log ng)$, wobei n in diesem Fall der Anzahl der Elemente in dem jeweiligen Cluster $n = |DI_{C_i}|$ und g der Anzahl der Boxgrößen entspricht.

4.8.3 Unsicheres Sortieren

Artikel [65] beschreibt einige Eigenschaften des TimSort-Algorithmus. Dieser effiziente Sortierungsalgorithmus basiert auf dem „Teile und Herrsche“ Prinzip und ist genau betrachtet, eine verbesserte Variante des Mergesort Algorithmus [10, S.127 ff.]. Im Allgemeinen agiert der Mergesort Algorithmus, von dem auch parallele Implementierungen existieren [30], nach dem folgenden Muster.

1. Als Basis zur Sortierung steht ein Array aus Elementen zur Verfügung. Dieses Array wird rekursiv durch halbieren aufgeteilt, bis in der aufgeteilten Menge genau eine Mindestmenge an Elemente repräsentiert sind.
2. Diese Kleinstmengen sortiert der Algorithmus.
3. Durch das sortierte Mischen bzw. Verflechten der jeweiligen Menge mit der Nachbarmenge entsteht mittels der Rekursion die entsprechend sortierte Gesamtmenge

Durch dieses statische Vorgehen in der Sortierung besitzt der MergeSort Algorithmus eine Berechnungskomplexität bzgl. der Vergleiche für den Best, Worst und Average Case von $O(n \log n)$. TimSort verbessert dieses Sortierungsmuster durch die folgenden Kernideen:

- In der Praxis sind die zu sortierenden Listen selten vollständig unsortiert, d.h. es existieren Teilmengen bzw. Unterlisten, die bereits sortiert

sind. Diese können erkannt und deren Vorsortierung bei der Aufteilung berücksichtigt werden, sodass diese nicht nochmals sortiert werden.

- Die sog. Gallop-Funktionalität, die einer exponentiellen Suche entspricht, minimiert die Komplexität des Mischens zweier Unterlisten. Voraussetzung ist dafür eine Mindestgröße der Listen sowie die Vorgabe, dass alle Elemente der ersten Liste kleiner sind als alle Elemente der zweiten Liste. Mit Hilfe dieser Gallop-Funktionalität ist es möglich, zwei Listen mit einer Vergleichkomplexität von $O(\log n)$ zu mischen.
- Eine Erhaltung der Komplexität $O(n \log n)$ für alle Cases ist nur möglich, wenn die zu mischenden Unterlisten die gleiche Kardinalität besitzen. Zusätzlich zur Erkennung der vorsortierten Unterlisten werden diese ggfs. auf eine definierte Mindestlänge mit Hilfe von Insertion Sort erweitert.

Durch die drei genannten Kernideen, die in [] detaillierter diskutiert werden, ergibt sich eine eine Komplexität von $O(n)$ für den besten Fall, $O(n \log n)$ für den schlechtesten und durchschnittlichen Fall. Die Komplexitätseigenschaften des TimSort Algorithmus zeigen, dass dieser sich für das Sortieren großer Datenmengen eignet.

Das „unsichere Sortieren“ von Daten basiert in dieser Arbeit auf der Idee aus Kapitel 4.5. Auch bei TimSort vergleicht der Algorithmus zwei Elemente/Spektren anhand deren (Flux)-Werte. In Java ist der Vergleichmechanismus durch die generische Schnittstelle *Comparator* gekapselt. Um die Unsicherheiten in den Prozess zu integrieren implementiert die *Spectrum*-Klasse die o.g. *UncertainComparator*-Schnittstelle. Diese erweitert die Signatur der *compare* Methode um den Parameter *avg* des durchschnittlichen Spektrums (siehe Abbildung 4.7). Eine Klasse *UncertainTimSort* erweitert alle Signaturen der Methoden der originalen Implementierung mit dem *avg*-Parameter. Zusätzlich sind die Implementierungen so abgeändert, dass bei einer zu hohen Unsicherheit die unsichere Dimension eines Spektrums durch die entsprechende Dimension des durchschnittlichen Spektrums ersetzt wird. Die Detektion einer zu hohen Unsicherheit erfolgt mittels Schwellwertanalyse, deren Schwellwert pro Sortierungsvorgang definiert werden kann.

Die Ersetzung einer Dimension durch die Dimension eines durchschnittlichen Spektrums im Falle der SDSS-Spektren kann aus den folgenden Gründen vorgenommen werden.

- Eine Dimension alleine im hochdimensionalen Raum ist i.d.R. nicht ausschlaggebend für das erfolgreiche Clustering (siehe Kapitel 2.1).
- Durch die Vorverarbeitungsschritte sowie das Binning sind geringe Unsicherheiten bereits behandelt, d.h. diese treten nur auf, wenn innerhalb einer Binninggruppe alle Elemente unsicher sind.
- Mit einer Definitionen eines Schwellwerts der Unsicherheiten (inverse Varianz) von $1/\sigma^2 = 0$ sollen nur die Dimensionen ersetzt werden, die theoretisch „unendlich unsicher“ sind. Tests haben gezeigt, dass

dieser Parameter für die SDSS Spektren eine gute Wahl darstellt, siehe Kapitel 4.9.
- Existieren mehrere Spektren mit einer hohen Unsicherheiten in identischen Dimensionen in einem Cluster, sind diese als gleichwertig zu behandeln.

Neben den Funktionalitäten des Sortierens sind die Unsicherheiten auch beim Einlesen der Daten, für die Visualisierung oder für Exports relevant. Dimensionen einzelner Spektren, die eine hohe Unsicherheit aufweisen, werden für die Visualisierung durch einen Standardwert (in der Implementierung ist der Wert 1) ersetzt. Dies ermöglicht es, visuell zu analysieren, ob der Algorithmus die entsprechende Unsicherheit detektiert hat.

4.8.4 Cluster-Manipulationen

Da ein Cluster ausschließlich durch dessen Spektren repräsentiert wird, bestehten Cluster-Manipulationen in dieser Methodik aus den zwei Cluster-atomaren Operationen *add* und *remove*. Die vorherigen Beschreibungen zeigen, dass die Effizienz des Performance Steps unter anderem auf diesen beiden Operationen basiert. Eine effiziente Implementierung dieser Methoden ist aufgrund dieser Basis ein entscheidender Faktor um die Gesamtkomplexität des Verfahrens zu ermöglichen. Die folgenden Abschnitte skizzieren kurz die Implementierung der beiden Operationen. Sowohl das Hinzufügen als auch das Löschen von Elementen ist ein bzgl. des Designs kaskadierender Vorgang, der zentral in einem SpectraCluster-Objekt initiiert wird. Die Objekte leiten die Methodenaufrufe folgendermaßen durch die Klassenhiearchie: SpectraCluster → FractalMatrices → SpectralMatrix. Die FractalMatrices-Objekte dienen in diesem beiden Fällen zur aggregierten und gekapselten Verteilung der Vorgänge auf die boxgrößenabhängigen Matrix-Objekte.

Hinzufügen von Elementen in ein Cluster

Ziel der *add*-Operation ist das Einfügen eines Spektrums bzw. Elements in ein bestimmte Boxgröße. Voraussetzung für eine erfolgreiche Operation ist, dass das Element exakt die gleiche Anzahl an Dimensionen hat, wie alle anderen Elemente des Clusters, d.h. der Binning-Faktor eines Elements ist identisch. Das Hinzufügen eines Spektrum in ein neues Cluster läuft nach folgendem Prozess in der SpectralMatrix-Klasse ab.

1. **Diskretisierung des Spektrums**: Das bereits durch das Binning prozessierte Spektrum wird mit der entsprechenden Boxgröße diskretisiert.
2. **Einsortieren**: In jedem Cluster existiert eine geordnete Liste an diskretisierten Spektren (*rows*-Variable). Mit Hilfe der Binären Suche unter Berücksichtigung der Unsicherheiten mittels *UncertainComparator*-Schnittstelle wird das Spektrum in die Liste einsortiert.

3. **Komprimieren**: Durch die Betrachtung der benachbarten Elemente in der Liste wird festgestellt, wie aus dem diskreten Element die komprimierte Darstellung berechnet wird. Folgende Fälle können dabei auftreten.

Fall A Das Element ist bzgl. des Comparators identisch zu einem oder beiden benachbarten Elementen: In diesem Fall muss die Anzahl der identischen Elemente in den jeweiligen *CompactedGriddedSpectrum* Objekten modifiziert werden.

Fall B Das Element ist nicht identisch zu den benachbarten Elementen: Die Methode erstellt ein neues *CompactedGriddedSpectrum* Objekt mit einer neuen Gruppennummer in der Liste *compactedRows*.

Zusätzlich erstellt der Prozess des Hinzufügens direkte Links auf die diskretisierten bzw. komprimierten Objekte auf Basis von HashMaps, die nicht explizit in den UML-Diagrammen aufgeführt sind.

Das Komprimieren ist der entscheidende Schritt in dem Hinzufügen eines Elements innerhalb dieser Methodik. Die daraus resultierende Datendarstellung ist die grundlegende Datenbasis bzw. Vorbereitung für die Berechnung der Fraktalen Informationswerte, wie Kapitel 4.8.1 Abschnitt *CompactedGriddedSpectrum* beschreibt. Die Erstellung einer neuen Gruppe (**Fall B**) bedeutet bei einem Box Counting durch $q = 0$ eine signifikante Veränderung des Clusters bzgl. der Boxgröße. Eine detaillierte Berechnung mit dem Parameter $q \neq 0$ unter Annahme des **Fall B** resultiert ebenfalls in eine signifikante Änderung des FIV. Für den **Fall A** detektiert das Verfahren nur mittels $q \neq 0$ eine Veränderung im Cluster.

Löschen von Elementen aus einem Cluster

Die Löschoperation ist simpel gestaltet und kann dadurch performant ausgeführt werden. Mit Hilfe des kaskadierenden Methodenaufrufs resultierend in den Matrix-Objekten löscht der Algorithmus die Spektren aus den jeweiligen Listen. Der eigentliche Löschvorgang besteht aus dem Entfernen der im vorherigen Abschnitt beschriebenen Einträgen in den Listen *rows* und *compactedRows*. Zwei entsprechende HashMaps ermöglichen durch eine Abbildung des originalen Index auf die jeweiligen diskretisierten bzw. komprimierten Objekte eine direkte Löschung des jeweiligen Objekts ohne Suchaufwand. Analog zur Operation des Hinzufügens bestehen wiederum zwei Fälle:

- Das komprimierte Element besitzt eine identische Gruppennummer verglichen mit den benachbarten Elementen. Die Anzahl der identischen Elemente in den jeweiligen *CompactedGriddedSpectrum* Objekten muss minimiert werden.

- Alle (beiden) benachbarten Elementen besitzen nicht die identische Gruppennummer. Die Methode löscht das passende *CompactedGriddedSpectrum* Objekt.

Nach der Behandlung eines Falls ist die Löschoperation abgeschlossen. Die direkten Verlinkungen der Objekte durch die HashMaps ermöglichen eine Komplexität der Löschoperation von $O(1)$.

4.8.5 Binning

Die in Kapitel 4.6.1 theoretisch beschriebene Methodik des Binnings wird in diesem Abschnitt diskutiert. Das Fractal-Similiarity-Measures Verfahren setzt Binning ein, um Dimensionen einzelner Spektren zu aggregieren. Mit Hilfe verschiedener Aggregationsstufen und einer rekursiven Bootstrapping Analyse entsteht ein hierarchisch organisiertes Clustering. Zeile 9 des Algorithmus 4.1 zeigt durch die Verschachtlung der Methodenaufrufe, dass das Binning eines Clusters die primäre modifizierende Aktion des Algorithmus ist. Das Binning ist abhängig vom gewünschten Binningfaktor, der das beschriebene $factors$-Array bestimmt sowie den entsprechenden Unsicherheiten. Die Referenzimplementierung des Vorgehens erfolgt nach diesem konzipierten Prozess, jeder bei einem Binning einer Liste an Spektren für jedes Spektrum die folgenden Schritte abarbeitet:

1. Initialisierung des $factors$ Array zum Einsatz der $createFactors$ Methode
2. Für jede gebinnte Dimension:
   (a) Der Fluxwert einer gebinnten Dimension berechnet sich aus der gewichteten Summe der ursprünglichen Dimensionen. Das Gewicht besteht aus dem Produkt des aktuellen Faktors und der inversen Varianz der passenden Dimensionen entsprechend der Gleichung 4.13.
   (b) Die inverse Varianz der gebinnten Dimension entspricht der Summe der ursprünglichen inversen Varianzen gemäß Gleichung 4.14.
   (c) Adaption der Faktoren durch die $createFactors$ Methode
3. (Optional) Die gebinnten Dimensionen werden bzgl. einer definierten Funktion normiert. Bspw. kann eine Funktion zum Normalisieren die statistischen Momente eines gesamten gebinnten Spektrums analysieren und die Dimension bzgl. des MAD oder Min/Max normieren.

Die Wahl einer Normalisierungsfunktion ist fallabhängig und kann nicht generell festgelegt werden. Unterschiedliche Anwendungsfelder besitzen verschieden organisierte Daten, die ggfs. mit Hilfe der normalisierenden Funktion vergleichbar gemacht werden müssen. Nicht normalisierte Daten führen unter Umständen durch die nicht sichergestellte Vergleichbarkeit und die daraus resultierende fehlerhafte Diskretisierung zu einem fehlerhaften

bzw. ungenauen Clustering-Ergebnis. Eine Funktion $normalize(a_{ij}, a_i) = (a_{ij} - median(a_i))/mad(a_i)$ normalisiert die SDSS Daten.

4.9 Parameterbeschreibung

Die vorherigen Abschnitten diskutieren die einzelnen Methodiken des Verfahrens sowie deren Implementierungen. Darin enthalten sind einzelne unabhängige Parameter deren Modifikation die Qualität des Ergebnisses beeinflussen kann. Auf Basis vorherigen Beschreibungen zeigt dieses Kapitel die Auswirkungen einzelner Parameter auf die Methodik.

4.9.1 Boxgrößen

Eine zentrale Menge an Parametern stellen die Boxgrößen dar. Diese werden sowohl im Bootstrapping als auch im Performance Step verwendet, um eine multiple, unterschiedliche und unabhängige Diskretisierung der zu analysierenden vorprozessierten Daten vorzunehmen. Das Ziel der Diskretisierung in diesem Verfahren ist grundsätzlich eine Gruppierung von Elementen innerhalb einzelner bzw. mehrerer Dimensionen bzgl. jeweils einer Boxgröße.

Für die Referenzimplementierung in diesem gesamten Kapitel sowie in den Tests wird aufgrund der identischen Wertebereiche der verschiedenen Dimensionen der SDSS-Daten eine einzelne Boxgröße verwendet. Die Verwendung des Verfahrens für weitere Anwendungsfelder oder Datensätze muss eine Prüfung beinhalten, ob ggfs. eine dimensionsabhängige Boxgröße aufgrund unterschiedlicher Wertetypen oder Wertebereiche sinnvoll ist.

Unabhängig von der Anzahl der Dimensionen ist die Wahl der einzelnen Boxgrößen. Durch das o.g. Ziel und den theoretischen Überlegungen aus Kapitel 4.3, 4.4 ist die Wahl der Boxgrößen entscheidend für ein korrektes Clustering Ergebnis. Abbildung 4.4 visualisiert die Riemannsumme über die FIV und zeigt beispielhaft den typischen Verlauf der FIV in Abhängigkeit zur Boxgröße. Der typische Verlauf besteht aus den beiden folgenden theoretischen Extrema, die durch das Verfahren direkt gegeben sind.

- Das globale Minimum entsteht sobald jedes Element/Spektrum durch eine sehr kleine Boxgröße in einer eigenen Box für alle Dimensionen ist. Dieser Fall tritt genau dann ein, wenn eine Boxgröße kleiner oder gleich dem kleinsten der Differenz der beiden kleinsten Werte innerhalb einer Dimension entspricht. Einzige Ausnahme ist der Fall zweier oder mehrerer absolut identischer Spektren, die die Anzahl der maximal möglichen Boxen entsprechend reduzieren. Die Wahl des kleinsten Werts als Boxgröße zur Generierung des Minimum ist möglich, jedoch kann durch das Aufrunden in Gleichung 4.4 eine Differenzierung zweier Werte geringer Differenz verloren gehen. Aus diesem Grund sollte die Differenz der zwei kleinsten Werte innerhalb einer Dimension über

alle Spektren hinweg betrachtet als Boxgröße verwendet werden.

- Analog zum Minimum ist das Maximum genau dann erreicht, wenn die Boxgröße größer oder gleich dem Maximalen Wert aller Elemente über alle Dimensionen gewählt wurde. In diesem Fall ist das Aufrunden während des Diskretisierens nicht relevant.

Die Wahl der Boxgrößen sollte zur Abdeckung aller Unterschiede ebenfalls die Boxgrößen der beiden Extrema inkludieren. Des Weiteren sind weitere Boxgrößen einzusetzen, um ein möglichst breites Sampling der diskretisierten Daten zu erhalten. Eine Erweiterung der Anzahl der Boxgrößen bedeutet ein präziseres Sampling bzw. genauere Differenzierung mittels unterschiedlicher Diskretisierungsskalen. Eine Selektion von Boxgrößen zwischen den beiden Extrema erfolgt im Fall der SDSS-Spektren mit Hilfe einer linearen Funktion. Eine Wahl einer Boxgröße kleiner als das notwendige Minimum bzw. größer als das notwendige Maximum bewirkt aufgrund der Extrema keinen Unterschied im Clustering Ergebnis, solange weitere Boxgrößen zwischen den Extrema genutzt werden. Jedoch bewirkt die Wahl einer zu großen kleinsten Boxgröße bzw. einer zu kleinen größten Boxgröße eine unzureichende Abdeckung und ggfs. ein fehlerhaftes Ergebnis.

4.9.2 Binningfaktor

Eine bzgl. der Boxgrößen zusätzliche Diskretisierung findet mit dem in Kapitel 4.6.1 beschriebenen spektrenspezifischen Bootstrapping statt. Der steuerende Binningfaktor entspricht einer rationalen Zahl die den Kompressionfaktor beschreibt. Ein hoher Binningfaktor bedeutet eine hohe Aggregration der Dimensionen, während ein kleinerer Binningfaktor analog eine kleine bzw. keine Aggregation der Dimensionen darstellt ($d \geq b \geq 1$). Ohne zusätzliche Definition wählt das Verfahren automatisch einen Binningfaktor $b = d/2$ und halbiert diesen mit jeder Iteration des Bootstrapping Algorithmus 4.1. Die Wahl eines Binningfaktors außerhalb des o.g. Intervalls wird vom Algorithmus mit einer Exception abgefangen, da ohne Datenquelle nicht weitere Dimensionen entstehen können ($b < 1$) bzw. weniger als eine Dimension ist nicht möglich ($b > d$).

Prinzipiell kann der Minimierungsrate des Binningfaktors angepasst werden. Eine Verkleinerung der Anpassungsrate bewirkt eine Erhöhung der Iterationsschritte des Algorithmus. Daraus resultiert ein verfeinertes Bootstrapping das in Abhängigkeit des verwendeten Datensatzes in einer geringen Anzahl an Clustern endet, da durch die Iterationen weitere Elemente aus dem Clusters ausgeschlossen werden. Die Cluster verkleinern sich dadurch und fallen teilweise unter die Mindestgröße eines Clusters. Umgekehrt bedeutet eine Vergrößerung der Anpassungsrate auf z.B. $b = \frac{b}{8}$ durch die geringe Anzahl eine schnellere Konvergenz des Algorithmus zu dem Minimum $b => 1$ das Überspringen einzelner Analyseschritte. Durch das Überspringen ist es möglich, dass der Algorithmus frühzeitig abbricht. Im Falle der SDSS

Spektren existieren die in Kapitel 4.7.1 beschreibenen optionalen Abbruchbedingungen. Diese stoppen die iterative Aufteilung der Cluster, sobald die Anzahl der zu kleinen Cluster zu stark wächst. Das Überspringen begünstigt in Kombination einen zu frühen Abbruch durch ein $b >> 1$, was einer sehr groben Analyse und damit einem sehr ungenauen Ergebnis entspricht.

Kapitel 5

Evaluierung

Zur Demonstration der Funktionalität des Verfahrens aus Kapitel 4 und der Bestätigung der Berechnungskomplexitätsannahmen wurde die aktuelle Implementierung des FSM Verfahrens einer Evaluierung unterzogen. Wie in Kapitel 2.1.3 beschrieben ist die Validierung der Präzision eines Clustering Algorithmus eine Möglichkeit zur Evaluation. Aufgrund der fehlenden Annotation innerhalb des SDSS Datensatzes, kann dies nur für einen synthetischen Datensatz erfolgen. Dieses Kapitel beschreibt in den folgenden Abschnitten die Evaluation des Algorithmus für die synthetischen und reellen Daten bzgl. der Laufzeit sowie eine Analyse der Präzision für den synthetischen Datensatz.

5.1 Versuchsaufbau und -ablauf

Der Versuchsaufbau ist sowohl für die generierten Daten als auch die observierten SDSS Spektren identisch. Ein Rechencluster bestehend aus vier Knoten mit je 32 physischen bzw. 64 virtuellen Rechenkernen sowie 512 GByte Arbeitsspeicher prozessieren einzelne Aufträge. Diese Aufträge bestehen aus den in den folgenden Kapiteln diskutierten einzelnen Tests. Abbildung 5.1

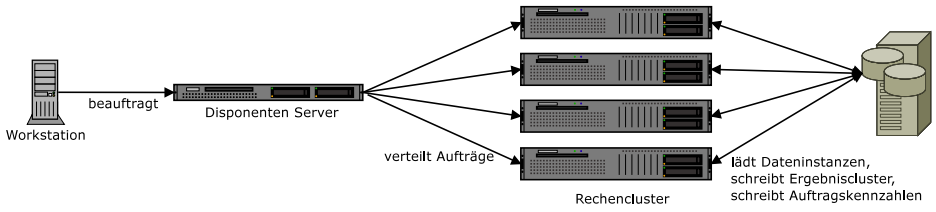


Abbildung 5.1: Schematische Skizze der Organisation der Aufträge

zeigt schematisch die Organisation der Aufträge. Eine Workstation liefert Auftragsparameter an einen Disponentenserver. Dieser nimmt Aufträge an und verteilt diese gemäß der definierten Auslastung auf einen der o.g. Ser-

ver des Rechenclusters. Da das Datenvolumen der genannten Datensätze die Größe des Arbeitsspeichers eines einzelnen Knoten nicht übersteigen, ist in diesem Fall keine verteilte Berechnung notwendig. Ein zentraler Speicher stellt für alle Server die Datensätze bereit.

Nach der Bearbeitung eine Auftrags speichert der Server das Ergebnis sowie die folgenden Auftragskennzahlen in den zentralen Speicher. Das Evaluationsmodul in der Implementierung des FSM Verfahrens protokolliert die folgenden Auftragskennzahlen:

- Laufzeit der vollständigen Analyse ohne Einlesezeit (Die reine Einlesezeit wird nicht beachtet, da diese für die Analyse des Verfahrens durch die reine Abhängigkeit zum angebundenen Speicher nicht relevant ist)
- Laufzeit des Bootstrapping Steps
- Laufzeit des Performance Steps
- Anzahl der Dateninstanzen im Bootstrapping Step
- Anzahl der Dateninstanzen im Performance Step
- Anzahl der zugewiesenen Dateninstanzen
- Anzahl der eingelesenen Dateninstanzen
- Anzahl der Dimensionen
- Durchschnittlich benötigte Zeit zur Kalkulation alle FSM Werte für eine einzelne Dateninstanz.
- Anzahl der verwendeten Boxgrößen im Performance Schritt
- Anzahl der verwendeten Boxgrößen im Bootstrapping Schritt
- Anzahl der gefundenen Cluster
- Im Fall des synthetischen Datensatzes: Prozentuale Fehlerquote

Die Fehlerquote F berechnet sich aus folgender Gleichung:

$$F = \frac{wc}{n}. \tag{5.1}$$

Dabei entspricht n der Gesamtanzahl aller Dateninstanzen sowie wc der Anzahl der falsch zugewiesenen Dateninstanzen. Ein Python [129] Skript produziert auf Basis der Auftragskennzahlen die in den folgenden Kapiteln dargestellten Abbildungen mittels der matplotlib Bibliothek [131]. Mit dem o.g. Verfahren ist es möglich, einen manuellen Aufwand zur Auftragsorganisation und zur darauf folgenden Analyse der Evaluationen zu minimieren.

In allen Evaluationen bearbeitet der Bootstrapping Schritt insgesamt $n_{bootstrap} = 0.2\ n$ Dateninstanzen. Darüber hinaus erfolgt die Selektion der Dateninstanzen für den Bootstrapping Schritt randomisiert und für den Fall des synthetischen Datensatzes zusätzlich stratifiziert, wie in Kapitel 5.2.1 beschrieben.

5.2 Evaluierung anhand synthetischer Daten

Die Evaluation des Fractal Similarity Measures Verfahren erfolgt in diesem Kapitel anhand mehrerer Ausprägungen eines synthetischen Datensatzes. Ziel bei diesen Überprüfungen ist die Prozessierung eines annotierten hochdimensionalen Datensatzes, der auch Unsicherheiten beinhaltet, um eine objektive Analyse der Ergebnisse zu ermöglichen.

5.2.1 Datenbeschreibung

Der synthetische Datensatz besteht aus 4×10^6 Dateninstanzen und ist konzipiert für Laufzeitanalysen sowie zur Validierung der Präzision bzgl. der Annotation. Zur Etablierung einer komprimierten Darstellung sind im Datensatz vier Cluster integriert, die identische Eigenschaften in randomisierten Dimensionen besitzen. Sichergestellt ist die Aufteilung der Dimensionen bzgl. der Cluster, da jedes Cluster in einem Viertel der Dimensionen spezifische Eigenschaften aufweist. Abhängig von der Anzahl der Dimensionen existieren auch überlappende Cluster. Zur Vereinfachung einer stratifizierten randomisierten Selektion während des Bootstrapping Vorgangs sind die Daten entsprechend in Cluster-abhängigen Ordnern organisiert. Mit dieser Organisation kann posteriori eine korrekte Cluster-Zuordnung festgestellt oder falsifiziert werden.

Ein Ziel der Konstruktion des o.g. Datensatzes ist die Demonstration der Detektion geringer Unterschiede bei einer Überschneidung von Eigenschaften im Datenraum über alle Cluster hinweg. Die spezifischen Eigenschaften sind durch entsprechende Spitzen im jeweiligen Graph zu erkennen. Insgesamt besitzen alle Dimensionen in allen synthetischen Spektren ein definiertes gleichverteiltes Rauschen. Zusätzlich ist eine Variation der Dimensionen der spezifischen Eigenschaften eines Clusters ebenfalls durch eine gleichverteilte Dichtefunktion gesteuert. Prinzipiell ist es möglich die o.g. Verteilungen durch weitere Verteilungsfunktionen zu ersetzen.

Insgesamt enthält der Datensatz bis zu 5×10^5 Dimensionen, die in den Ausprägungen 50, 500, 5000 sowie 5×10^4 und 5×10^5 zur Verfügung stehen. Theoretisch besteht die Möglichkeit der Generierung weiterer Datensätze mit einer noch größeren Zahl an Dimensionen. Eine Maximierung der Dimensionen resultiert in dem dargestellten Algorithmus primär in einer Verlängerung der Laufzeit, weshalb für Skalierungstests 500 und 50 Dimensionen zum Einsatz kommen.

Alle Dateninstanzen des Datensatzes sind durch die skizzierten Eigenschaften bereits normiert und besitzen den vorprozessierten Status eines vergleichbaren Spektrums (siehe Abbildung 4.1, Schritt 0). Eine dedizierte Vorverarbeitung ist daher für diesen synthetischen Datensatz im Gegensatz zu den SDSS Daten nicht notwendig. Die Speicherung des synthetischen Datensatzes erfolgt im binären Dateiformat (siehe Kapitel 3.4.1), das ebenfalls

für die SDSS Spektren eingesetzt wird. Dies ermöglicht den Einsatz identischer Leseoperationen für reelle und synthetische Datensätze und eliminiert eventuelle Effekte in der Laufzeit bzgl. Dateneingabe bzw. Lesevorgangs.

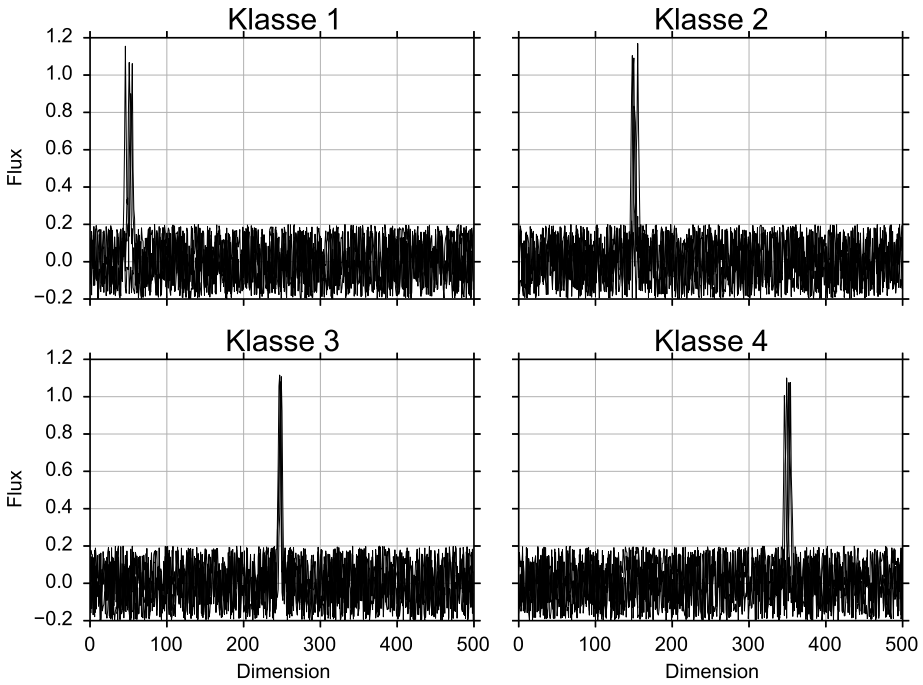


Abbildung 5.2: Visualisierung der vier vom Verfahren zu bestimmenden Klassen. Jede Klasse ist in den Graphen repräsentiert durch 6 randomisiert ausgewählte Dateninstanzen zur Darstellung des Rauschens. Die Abszisse zeigt jeweils die Dimensionen der Dateninstanzen, die Ordinate die entsprechenden generierten Dimensionswerte.

Abbildung 5.2 zeigt die vier zu erkennenden Klassen sowie deren beschriebenen Eigenschaften. Aufgetragen sind die einzelnen Eigenschaften bzw. Dimensionen der Dateninstanzen auf der Abszisse und die fiktiven Messwerte (Flux) auf der Ordinate.

Die nachfolgenden Diskussion zielt auf eine formalere Beschreibung der Datengenerierung. Die Menge der Parameter zur Generierung dieses synthetischen Datensatzes bestehen aus:

- Rausch-Varianz σ^2_{Rausch}
- Signal-Varianz σ^2_{Signal}
- Anzahl der Dimensionen bzw. Signale pro Dateninstanz $numDim$
- Anzahl der Dateninstanzen $numDi$
- Anzahl der Klassen $numC$
- Anzahl der Elemente in einer Datei $numS$

Aus den genannten Parametern resultierten die folgenden Gleichungen, die die Eigenschaften des Datensatzes generieren:

$$posPDim_0 = \frac{0.5\ numDim}{numC + 1}, \tag{5.2}$$

$$posPDist = \frac{numDim - 2\ posPDim_0}{numC}. \tag{5.3}$$

Gleichung 5.3 beschreibt die Anzahl der Dimensionen, die zwischen zwei dimensionslokalen Maxima einer Dateninstanz liegen, basierend auf der initialen Dimension $posPDim_0$ (siehe Gleichung 5.2). Die Maxima entsprechen den skizzierten Eigenschaften einer Klasse.

$$\forall 0 < d \leq D : noise_d = (2\ rand_d\ \sigma^2_{Rausch}) - \sigma^2_{Rausch} \tag{5.4}$$

Gleichung 5.4 zeigt die Zusammensetzung der Rauschkomponente, die in jeder Dimension vorliegt. Dabei ist $rand_d$ eine generierte Zufallszahl aus einer gleichverteilen Wahrscheinlichkeitsdichtefunktion, die Werte $0 \leq rand_d \leq 1$ liefert.

Die Variation der Position der klassenspezifischen Eigenschaften sei bestimmt von $cPos$ in Gleichung 5.5. Dabei sei die Identifikationsnummer einer Klasse i gegeben.

$$cPos = posPDim_0 + (i\ posPDist) + (2\ rand_i\ \sigma^2_{Signal}) - (D\ \sigma^2_{Signal}) \tag{5.5}$$

$$\forall 0 < d \leq D : signal_d = noise_d + e^{-0.2\ (i-cPos)^2} \tag{5.6}$$

Das Rauschen $noise_d$ beschreibt das Rauschen einer Dateninstanz (siehe Gleichung 5.4) sowie $signal_d$ den finalen für das Clustering relevanten Wert einer Dateninstanz in der $d-ten$ Dimension (siehe Gleichung 5.6). Entsprechend der o.g. Gleichungen generiert ein separates Programm die diskutierten 4×10^6 Dateninstanzen.

5.2.2 Laufzeitanalyse

Mit Hilfe der in den vorherigen Kapiteln beschriebenen synthetischen Daten, wurde das Fractal Similarity Measures Verfahren bzgl. der Laufzeit evaluiert. Dieses Kapitel fokussiert auf die Skalierbarkeit des Algorithmus bzgl. der Anzahl der Dateninstanzen und der Dimensionen. Das Fractal Similarity Measures Verfahren ist in die zwei Schritte des Bootstrappings und des Performance Schritts unterteilt, mit jeweils einer eigenen Komplexitätsschätzung. Aufgrund dieser Separation erfolgt die Laufzeitanalyse des Verfahrens zweigeteilt. Die beiden folgenden Abschnitte diskutieren die Laufzeitanalysen der beiden Schritte. Angemerkt sei, dass das Einleseverfahren der Daten bei jeder Messung randomisiert erfolgt.

Bootstrapping Schritt

Dateninstanzbasierte Laufzeitananalyse Der Bootstrapping Schritt erfolgt, wie in Kapitel 4.7.1 dargestellt, mit Hilfe eines rekursiven Verfahrens. Die Messungen wurden für eine Vielzahl von Dateninstanzen durchführt. Die maximale Anzahl an Dateninstanzen ($\approx 8 \times 10^5$) ist gegeben durch die Anzahl der verfügbaren Dateninstanzen (4×10^6) und den definierten relativen Anteil der für das Bootstrapping eingesetzten Spektren (20%). Die Größe der verwendeten Dimensionen liegt bei 50 zur Minimierung der Berechnungszeit. Aufgrund des Berechnungsaufwandes existieren Messpunkte mit $n_x = n_{x-1} * 2$ wobei $n_0 = 2 \times 10^3$ ist. Abbildung 5.3 zeigt im rechten

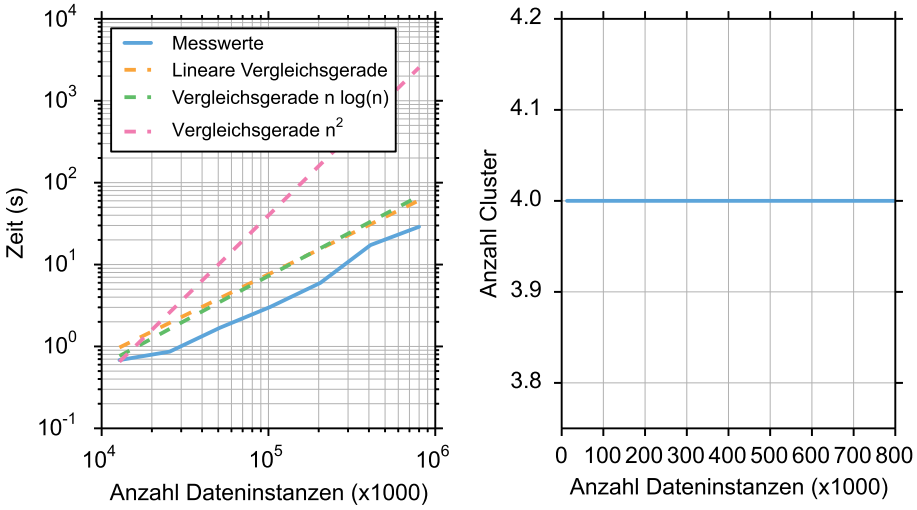


Abbildung 5.3: Visualisierung der Bootstrapping Schritt Skalierungstests. Dargestellt sind die Anzahl der Klassen und die benötigte Rechenzeit auf den Ordinaten, auf der Abszisse jeweils die Anzahl der verwendeten Dateninstanzen. Die Skalierungstests zeigen die ermittelten Messwerte, deren berechnete Lineare Regressionsgerade sowie den Verlauf einer Geraden aus der Klasse der Komplexitätsschätzung.

Teil die erwartete und korrekte Anzahl der gefundenen Cluster, im linken Teil die benötigte Laufzeit des Bootstrappingschritts. In beiden Graphen repräsentiert die Abszisse die Anzahl der Dateninstanzen, die im Bootstrapping Schritt prozessiert sind. Die Darstellung der Laufzeiten zeigt neben den Messwerten weitere Geraden, die verschiedene Komplexitätsklassen repräsentieren. Gleichung 5.7 stellt die Komplexitätsannahme des Bootstrapping Schritts noch einmal dar:

$$O(n \log n/k \log d + \log dnd). \tag{5.7}$$

Dabei steht n für die Anzahl der Dateninstanzen, d für die Anzahl der Dimensionen sowie k für die Anzahl der Cluster.

Insgesamt zeigen die Messwerte einen log-linearen Verlauf, der sich durch die Parallelität zur log-linearen Vergleichsgerade auszeichnet. Ebenfalls zu erkennen ist eine sub-lineare Steigerung der Laufzeit bis ca. 2×10^4 Dateninstanzen.

Dimensionsbasierte Laufzeitananalyse Neben der Laufzeit des Bootstrapping Schritts betrachtet diese Evaluation die Skalierbarkeit bzgl. der Dimensionen. Zur Darstellung dieser Skalierbarkeit existieren die o.g. vier unterschiedlichen Datensätze mit 1×10^4 Dateninstanzen und den Ausprägungen in der Dimensionsanzahl 5×10^x, mit $1 \leq x \leq 5, x \in \mathbb{N}$. Die Laufzeitmessungen wurden für jeden Datensatz durchgeführt. Abbildung 5.4 stellt

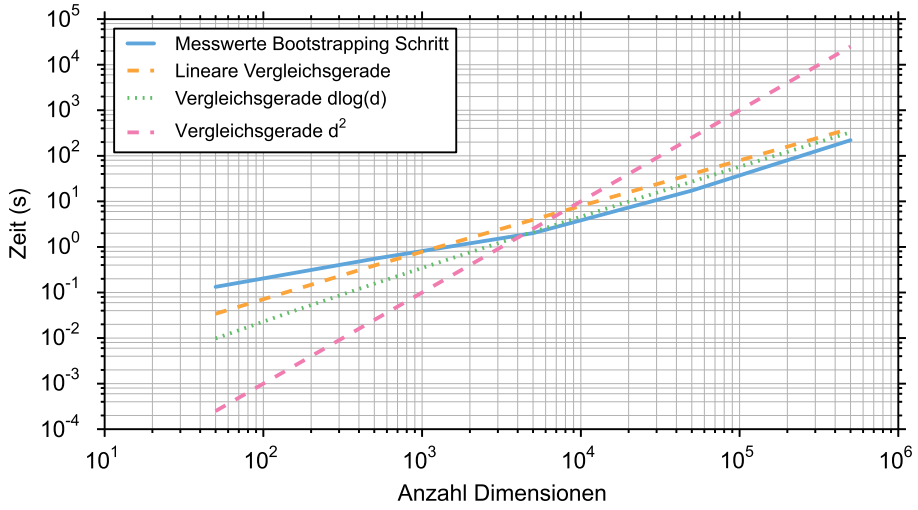


Abbildung 5.4: Darstellung der Skalierbarkeit des Bootstrapping Algorithmus. Die Abszisse stellt die Anzahl der Dimensionen, die Ordinate die benötigte Laufzeit dar.

die Laufzeiten bzgl. der verschiedenen Anzahl an Dimensionen dar. Die logarithmische Darstellung zeigt eine sub-lineare Skalierung des Algorithmus zwischen 50 und 5000 Dimensionen, wobei die Steigerung der Laufzeit zwischen diesen beiden Anzahlen an Dimensionen ca. Faktor 4 entspricht. Explizit erwähnt sei, dass die Anzahl der Dimensionen sich um Faktor 100 unterscheiden. Die anschließenden Tests mit 5×10^4 und 5×10^5 Dimensionen zeigen den eigentlichen erwarteten log-linearen Anstieg der Laufzeit. Abbildung 5.4 zeigt darüber hinaus zur Übersicht Vergleichsgeraden verschiedener Komplexitätsklassen. Dabei ist eine Parallelität des Messgraphen ab 5×10^3 Dimensionen zur log-linearen Gerade im logarithmisch skalierten

Koordinatensystem zu erkennen, was die Komplexitätsannahme bestätigt.

Der zuvor sublineare Anstieg der Laufzeit ist durch ein implementierungsspezifisches Detail zu erklären. In der Implementierung der Sortierfunktion besitzt eine Berechnungskomplexität von $O(d \log d)$ und die Vergleichsoperation bzgl. zweier diskreter Dateninstanzen erfolgt iterativ dimensionsweise. Falls die Aufteilungsoperation (siehe Kapitel 4.6.2) in einer Dimension eine Differenz zweier diskretisierter Dateninstanzen feststellt, bricht diese den Vergleichsvorgang ab. Bei einer kleinen Anzahl an Dimensionen ist in diesem Verfahren die Wahrscheinlichkeit hoch, bei zwei unterschiedlichen Klassen an Dateninstanzen, bei einer geringen Anzahl an Iterationen einen Unterschied festzustellen. Bei einer hohen Anzahl an Dimensionen ist diese Wahrscheinlichkeit geringer, weshalb eine höhere Anzahl an Iterationen und damit eine höhere durchschnittliche Laufzeit pro Vergleich zweier Dateninstanzen anfällt. In diesem Tests steigt der Anteil der Vergleichsoperationen des Sortieralgorithmus ab ca. 5000 Dimensionen und unterbricht den sublinearen Anstieg.

Performance Schritt

Dateninstanzbasierte Laufzeitananalyse Der zweite Schritt im Verfahren nutzt die im Bootstrapping Schritt erstellten Cluster und fügt zu diesen die restlichen 80% der Dateninstanzen hinzu. Daraus resultieren Tests mit einer maximalen Anzahl von $\approx 3.2 \times 10^6$ Dateninstanzen die zu prozessieren sind. Ebenfalls randomisiert ist die Reihenfolge, in der die Dateninstanzen zu einem Cluster hinzugefügt werden. Die Anzahl der Dimensionen sowie die Anzahl der Messpunkte entspricht dem Muster der Evaluation des Bootstrapping Schritts. In Abbildung 5.5 sind die Messwerte der Skalierungstests auf der Ordinate, die Anzahl der Dateninstanzen auf der Abszisse abgebildet. Neben den Messwerten sind Vergleichsgeraden dargestellt, die mögliche Komplexitätsklassen repräsentieren. Klar zu erkennen ist, dass der Verlauf der Messpunkte nicht parallel zur quadratischen Komplexitätsklasse verläuft. Die vergleichbaren Steigungen der linearen bzw. log-linearen Komplexitätsklasse sowie die Steigung der Messpunkte zeigen, dass der Algorithmus superlinear und maximal log-linear mit der Anzahl der Dateninstanzen skaliert. Durch eine minimal höhere Laufzeit bzw. Komplexität während der elementaren Operation des Hinzufügens einer Dateninstanz zu den diskretisierten Gruppen (siehe Kapitel 4.8.1) sind Abweichungen von der Geraden möglich. Diese sind durch die eingesetzte Datenstruktur gegeben und beeinflussen die Zeit, bis eine Dateninstanz einem Cluster zugewiesen wurde. Da das Verfahren im synthetischen Datensatz die vier Cluster findet, können die Daten nur auf genau diese vier Cluster verteilt werden, was bei einer hohen Anzahl an Dateninstanzen zu einem großen Suchraum für die Suche der korrekten Gruppe bzw. hochdimensionalen Zelle führt. Dabei ist es möglich, dass es in der aktuellen Implementierung bei einer so konzentrierten Menge

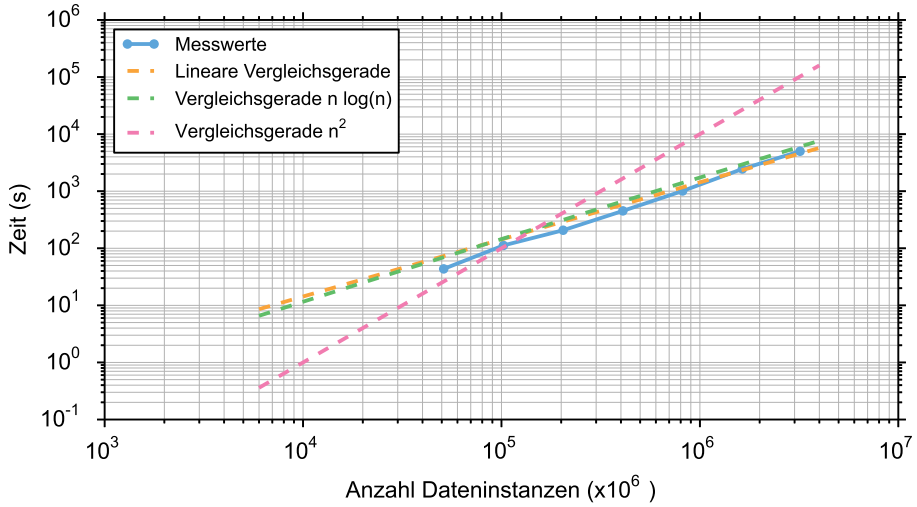


Abbildung 5.5: Visualisierung der Performanz Schritt Skalierungstests. Dargestellt sind die Anzahl der Klassen und die benötigte Rechenzeit auf den Ordinaten, auf der Abszisse jeweils die Anzahl der verwendeten Dateninstanzen. Die Skalierungstests zeigen die ermittelten Messwerte, deren berechnete Lineare Regressionsgerade sowie den Verlauf einer Geraden aus der Klasse der Komplexitätsschätzung.

an Dateninstanzen pro Cluster die Komplexitätsklasse des Hinzufügens auf maximal $\log n_{Cluster}$ ansteigt, wobei $n_{Cluster}$ die Anzahl der Dateninstanzen in einem Cluster entspricht.

Mit Hilfe einer optimierten Implementierung besteht die Möglichkeit, mit Hilfe einer Limitierung der maximalen Anzahl der Dateninstanzen in einem Cluster zur Berechnung der Fraktalen Informationswerte diesen Effekt zu minimieren. Dabei muss die Limitierung bzw. Wahl der Dateninstanzen stratifiziert erfolgen, sodass jede Gruppe (siehe Kapitel 4.8.1) repräsentiert ist. Dateninstanzen die aus einem Cluster entnommen werden, sind zwischenzuspeichern, um diese nach dem Performance Schritt wieder hinzuzufügen. Mit dieser Modifikation ist es möglich, die Variationen zu minimieren. Eine weitaus bessere Lösung würde ein Hashing Verfahren darstellen, dessen Berechnungskomplexität konstant ist und nur eine Gruppennummer bzgl. einer bestehenden Menge an Gruppen zurückliefert.

Dimensionsbasierte Laufzeitananalyse Wie bei der vorherigen Evaluation des Bootstrapping Schritts wurde eine Evaluation zur Ermittlung des Einflusses einer Steigerung der Dimensionen auf die Laufzeit durchgeführt. Die Untersuchungen beinhalten eine Verarbeitung von 1×10^5 Dateninstanzen bei einer variierenden Anzahl an Dimensionen gemäß der erstellen synthetischen Datensätze. Abbildung 5.6 stellt die Laufzeitanalyse

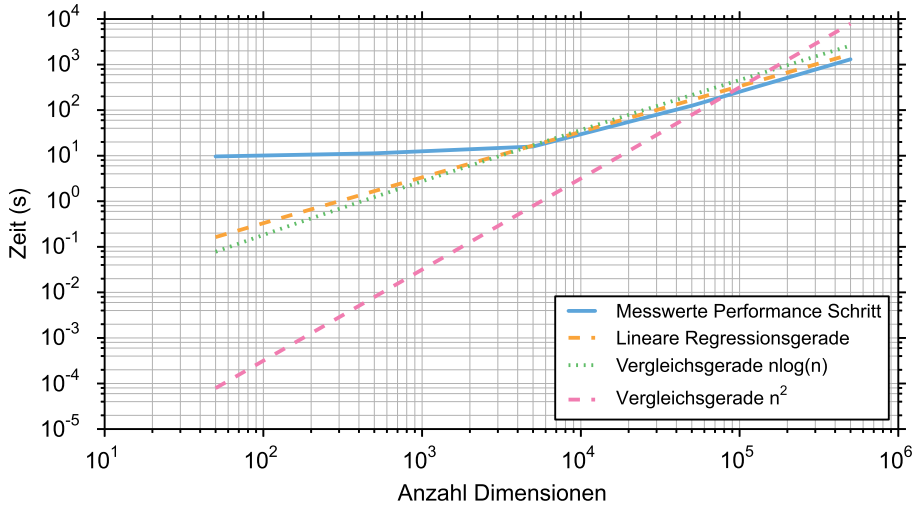


Abbildung 5.6: Darstellung der Skalierbarkeit des Performance Algorithmus. Die Abszisse stellt die Anzahl der Dimensionen, die Ordinate die benötigte Laufzeit dar. Vergleichsgraphen zeigen die evtl. Parallelitäten bzgl. möglicher Komplexitätsklassen.

des Performance Schritts bzgl. der Anzahl an Dimensionen dar. Analog zur Analyse des Bootstrapping Schritts erfolgt die Ausführung der Messungen. Der Verlauf des Messungsgraphen zeigt einen ähnlichen Verlauf zu dem Verlauf des Skalierungsgraphen aus Abbildung 5.5. Das logarithmisch skalierte Koordinatensystem zeigt eine vergleichbare Steigung zwischen der log-linearen sowie der linearen Vergleichsgerade. Zu erkennen ist, dass die Parallelität bzgl. der linearen Vergleichsgerade gegeben ist und die log-lineare Vergleichsgerade verglichen mit der Messungsgeraden divergiert. Dies zeigt, dass die Komplexitätsannahme der linearen Abhängigkeit des Performance Schritts zu der Anzahl der Dimensionen gerechtfertigt ist. Der identische Effekt der steigenden Dimensionen die einen gesteigerten Vergleichsaufwand bei $\approx 5 \times 10^3$ hervorrufen, ist durch das identische Vergleichsverfahren auch im Performance Schritt gegeben.

5.2.3 Parameterstudie

Das Fractal Similarity Measures Verfahren besitzt den essentiellen Parameter der Anzahl der Boxgrößen. Wie in den vorherigen Kapiteln beschrieben, ist die Wahl der einzelnen Boxgrößen und die Selektion der Anzahl der Boxgrößen ein Faktor, der die Qualität und die Laufzeit des Clusterings wesentlich beinflusst. Die folgenden beiden Kapitel beschreiben kurz den evaluierten Einfluss des Parameters auf die Laufzeit sowie auf die Genauigkeit bzgl. des synthetischen Datensatzes. Die Durchführung der folgend

beschriebenen Evaluationen basiert auf dem synthetischen Datensatz mit 500 Dimensionen.

Laufzeiteinfluss

Im Fractal Similarity Verfahren beinflusst die Anzahl der Boxgrößen ausschließlich den Performance Schritt, da der Bootstrapping Schritt die zentralen Berechnungen auf einem einzigen diskretisierten Gitter durchführt. Während des Performance Schritts steuert die Anzahl der Boxgrößen, die Menge der SpectralMatrix-Objekte (siehe Kapitel 4.8.1) und damit den Speicherverbrauch bzw. die Berechnungszeit der FSM bzw. FIV Werte. Dabei muss die Selektion der Boxgrößen unter den in Kapitel 4.9.1 beschriebenen Punkten erfolgen. Die minimale bzw. maximale Boxgröße ist i.d.R. konstant und gegeben durch den Werte der Dimensionen im Datensatz während die Anzahl der Boxgrößen dazwischen variiert. Abbildung 5.7 zeigt Messwert-

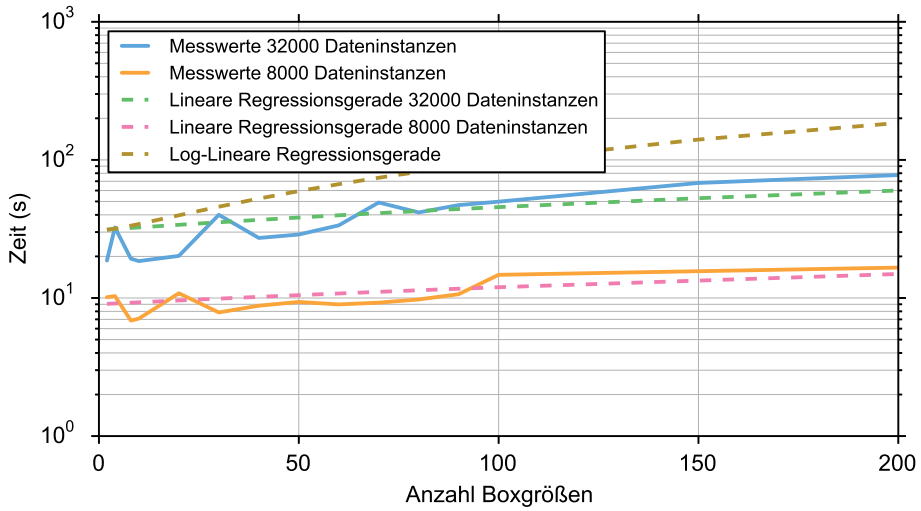


Abbildung 5.7: Visualisierung der Laufzeit mit variierenden Boxgrößen für 8×10^3 und 32×10^3 Dateninstanzen. Die Abszisse zeigt linear skaliert die Anzahl der Boxgrößen, die Ordinate die Laufzeit angegeben in Sekunden.

graphen für 8×10^3 und 32×10^3 Dateninstanzen mit jeweils variierender Anzahl an Boxgrößen. Die Abbildung zeigt für beide Dateninstanzmengen eine linear steigende Laufzeit sowie positive und negative Abweichungen vom linearen Verlauf. In den Messungen erfolgt die Selektion der Boxgrößen nach einem einfachen Muster. Nach der Ermittlung der Minima und Maxima in den Dimensionsdaten ist die Selektion der dazwischenliegenden Boxgrößen ebenfalls linear und äquidistant. Dabei ist es möglich, dass durch die Diskretisierungsoperation (u.a. bestehend aus dem Aufrunden) aus Gleichung 4.4 eine sehr große Menge an diskreten Gruppen für mehrere Boxgrößen ent-

steht. Dies erhöht den jeweiligen Berechnungsaufwand für die Erstellung und Modifikation der Matrix und resultiert in entsprechenden Schwankungen um die in Abbildung 5.7 dargestellten Messungen.

Genauigkeit

Neben dem Einfluss auf die Berechnungszeit besitzt die Anzahl der Boxgrößen auch einen Einfluss auf die Genauigkeit bzw. auf die Fehlerrate. Die Fehlerrate ist definiert durch:

$$F = \frac{\text{Anzahl fehlerhaft zugewiesener Dateninstanzen}}{\text{Anzahl aller Dateninstanzen im Performance Schritt}}. \tag{5.8}$$

Die Fehlerrate F repräsentiert den Anteil der im Performance Schritt fehlerhaft zugewiesenen Dateninstanzen im Verhältnis zu allen Dateninstanzen des Performance Schritts. Die Messungen evaluieren die fehlerhaft klassifi-

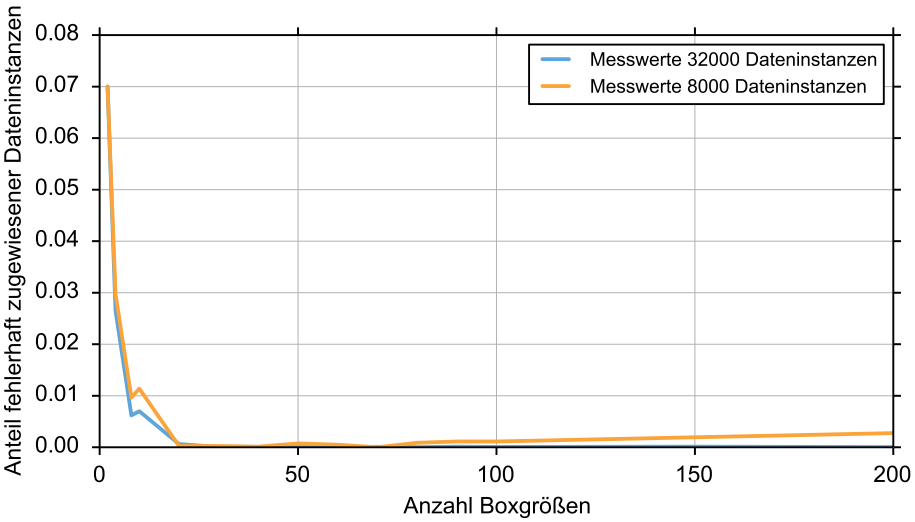


Abbildung 5.8: Visualisierung des Fehleranteils an falsch zugewiesenen Dateninstanzen in Abhängigkeit zur Boxgröße.

zierten Dateninstanzen nach der Beendigung des Performance Schritts durch ein Histogramm. Ein Vergleichsprogramm erstellt das Histogramm auf Basis der erstellten Cluster sowie einer Annotationstabelle, die aus der Erzeugung der Spektren resultiert. Dabei markiert das Vergleichsprogramm die einzelnen Dateninstanzen der erstellten Cluster bzgl. der gegebenen Klassen mit Hilfe der Annotationen und stellt die Majoritätsklasse eines Clusters fest. Alle von der Majoritätsklasse abweichenden Markierungen innerhalb eines Clusters sind als falsch klassifizierte Dateninstanzen einzustufen. Abbildung 5.8 visualisiert die ermittelten Fehler in Abhängigkeit zur Anzahl der Boxgrößen. Dabei wird deutlich, dass eine geringe Anzahl an Boxgrößen

zwar gemäß Abbildung 5.7 schnellere Ergebnisse liefert, diese jedoch ggfs. eine wesentlich höhere Fehlerrate aufweisen. Diese Evaluation zeigt auch, dass die Fehlerrate zwar mit einer höheren Anzahl an Boxgrößen gesenkt, aber nicht eliminiert werden kann. Die Fehlerrate liegt für beide Anzahlen an Dateninstanzen im Promille-Bereich, d.h. es ist durchschnittlich nur eine Dateninstanz fehlerhaft prozessiert worden.

5.3 Evaluierung anhand der SDSS Daten

5.3.1 Laufzeitanalyse

Die Laufzeitanalyse des Fractal Similarity Verfahrens erfolgt analog zu der Evaluation des Verfahrens anhand der synthetischen Spektren zweigeteilt. Die folgenden Untersuchungen zeigen die Ergebnisse der Laufzeitanalyse des Bootstrapping Schritts isoliert von der Analyse des Performance Schritts. Für beide Analysen gilt, dass die jeweils dargestellten Grafiken auf einem reellen Datensatz mit eigenen nicht isolierbaren inhärenten Eigenschaften durchgeführt wurden. In diesem Fall ist eine der Eigenschaften die Vielzahl der Cluster, die ca. $1 - 3\,\%$ der Anzahl der Dateninstanzen beträgt. Implizit bedeutet dies, dass eine isolierte Analyse des Komplexitätsparameters n nicht möglich ist, wie der entsprechende Abschnitt zeigt. Auf Grund der inhärenten Eigenschaft können diese Effekte in diesem Datensatz für eine Analyse nicht isoliert werden.

Bootstrapping Schritt

Die Analyse des Bootstrapping Schritts wurde mit mehreren Durchläufen mit jeweils unterschiedlicher Anzahl an Dateninstanzen durchgeführt. Dabei beträgt das Minimum ca. 3×10^3 sowie das Maximum $\approx 50 \times 10^3$ Dateninstanzen. Die Komplexitätsannahme aus Kapitel 4.7.4 des Bootstrapping Schritts beträgt $O(n \log n/k \log d + \log dnd)$. Eine direkte Abhängigkeit zwischen der Anzahl der Boxgrößen ist nicht gegeben. Ebenfalls nicht gegeben ist eine direkte Abhängigkeit zur Anzahl der Cluster k. Diese ist nur indirekt relevant, da der entsprechende Faktor nur zur Abschätzung der durchschnittlichen Größe eines Clusters beiträgt. Abbildung 5.9 zeigt die kombinierte Ansicht der Laufzeitanalyse des Bootstrapping Schritts. Die eigentliche Laufzeitanalyse, dargestellt im linken Schaubild, zeigt mit Hilfe der logarithmischen Axenskala das Verhältnis der gemessenen Laufzeit zur errechneten Maximallaufzeit, gegeben durch die Komplexitätsannahme. Im Vergleich verhält sich der Verlauf der Messungen proportional zum Verlauf der Komplexitätsannahme.

Während eines Bootstrapping Vorgangs teilt der in Kapitel 4.7.1 skizzierte Algorithmus den vorprozessierten Datensatz rekursiv auf. Jede Aufteilung erfolgt durch die entsprechenden Splitting Maße, die jeweils Daten-

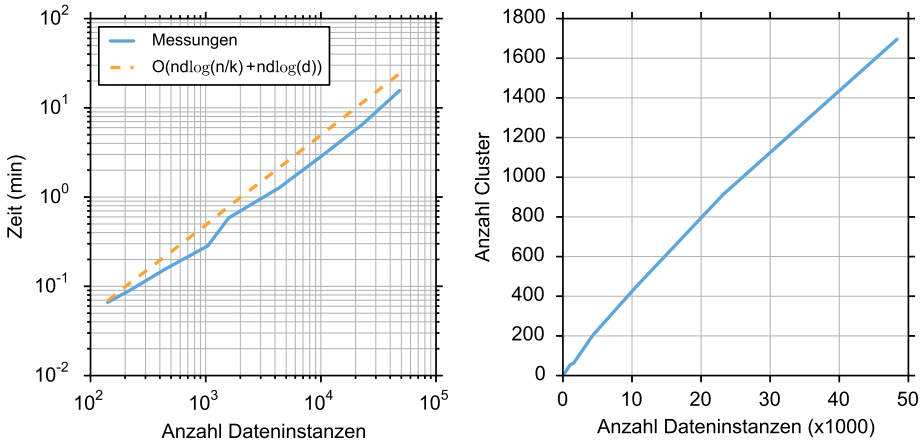


Abbildung 5.9: Laufzeitanalyse des Bootstrapping Schritts. Links dargestellt ist die eigentliche Laufzeitanalyse mit logarithmischen Axen, rechts die im SDSS Datensatz spezifische Eigenschaft des Cluster Wachstums mit linearer Axenskala.Beide Abszissen stellen die Anzahl der Dateninstanzen dar.

satzabhängig entscheiden, ob eine Partionierung vorgenommen wird. Durch diese dynamische Aufteilung sowie durch den Einsatz des randomisierten Einlesevorgangs und der randomisierten Selektion der Eingangsdaten ist es möglich, dass die Aufteilungshierarchie des Bootstrapping Algorithmus sich verändert. Darüber hinaus unterstützt die o.g. Abhängigkeit der Anzahl der Cluster von der Anzahl der Elemente einzelne Varianzen in den Messungen, die durch Sprünge zu erkennen sind. Es existieren zwei Sprünge durch diese Einflüsse im Graphen, die bei einer Anzahl von $\approx 1 \times 10^3$ sowie $\approx 1.6 \times 10^3$ abgebildet sind.

Performance Schritt

Die Laufzeit des Performance Schritts ist in Abbildung 5.10 bis zu einer maximal prozessierten Anzahl von $\approx 28 \times 10^4$ Dateninstanzen dargestellt. Prinzipiell ist bzgl. der Komplexitätsannahme des Performance Schritts eine lineare Abhängigkeit für die Anzahl der Dateninstanzen, die Anzahl der Boxgrößen, die Anzahl der Cluster sowie der Anzahl der Dimensionen gegeben. Angemerkt sei, dass diese Laufzeitanalyse eine Fokussierung auf die Skalierbarkeit bzgl. der Anzahl der Dateninstanzen n besitzt. Beim Einsatz des SDSS Datensatz ist dieser Parameter einfach zu variieren, während eine Modifikation der Anzahl der Dimensionen oder Boxgrößen weitere Seiteneffekte hevorrufen kann, der die Darstellung der Analyseergebnisse komplexer gestaltet. Abbildung 5.10 zeigt mehrere Eigenschaften der Analyse:

1. Variation der Komplexitätsannahme: Gegeben durch den Analysepro-

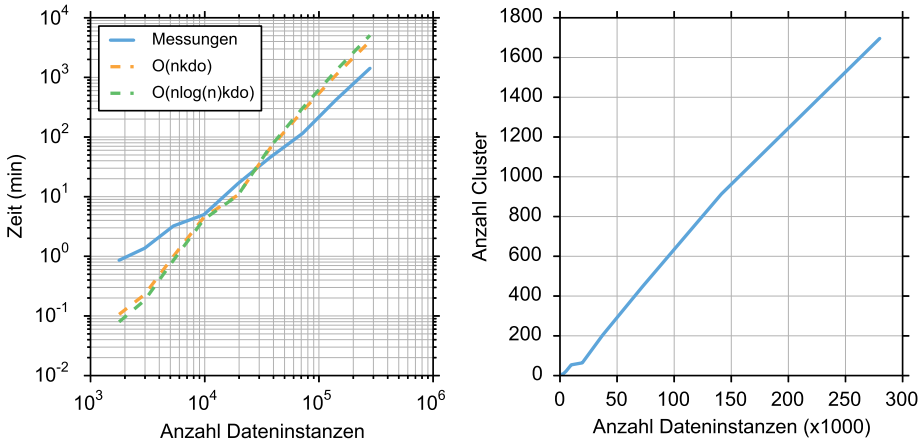


Abbildung 5.10: Laufzeitanalyse des Bootstrapping Schritts. Links dargestellt ist die Laufzeitanalyse mit logarithmischen Achsen, rechts die im SDSS Datensatz spezifische Eigenschaft des Cluster Wachstums mit linearer Axenskala. Beide Abszissen stellen die Anzahl der Dateninstanzen dar.

zess und der dadurch vorgeschalteten Analyse der Dateninstanzen durch den Bootstrapping Algorithmus und die daraus leicht unterschiedlich resultierenden Cluster, sind Variationen in der Linearität analog zu den der Darstellung der Laufzeitanalyse des Bootstrapping Schritts in Abbildung 5.9 repräsentiert. Sowohl die Anzahl der Cluster als auch deren Variation in deren Größe ist als Ursache der Variationen vorhanden.

2. Variation der Messungen: Analog zur Variation der Komplexitätsannahme sind die Abweichungen der Messlinie zu erklären.
3. Linearität: Gemäß der Komplexitätsannahme entsprechen die Messpunkte approximativ einem linearen Verlauf, was in der Parallelität bzgl. der beiden in Abbildung 5.10 links dargestellten Graphen zu entnehmen ist.

Insgesamt kann beobachtet werden, dass die Komplexitätsannahmen auch für Fall eines realen Datensatzes weitestgehend zutreffen.

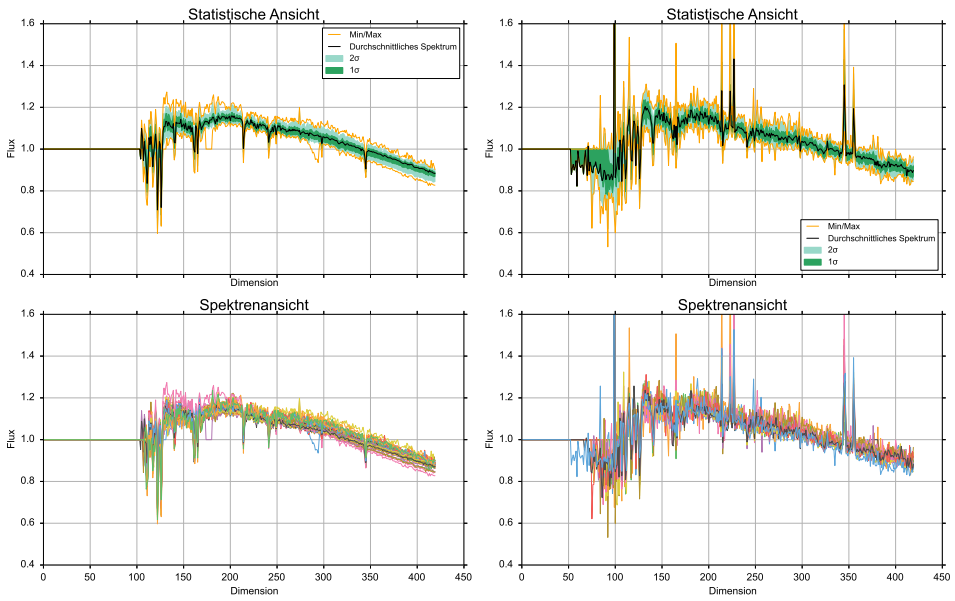
5.3.2 Clustering-Ergebnisse

Die Analyse des SDSS Datensatzes fand nach dem in Kapitel 5.1 beschriebenen Prozess statt. Dieses Kapitel stellt exemplarisch einige erste Ergebnisse vor, die aus einer Cluster-Analyse entstanden sind. Dabei ist die Anzahl der Boxgrößen innerhalb des Bootstrapping Schritts $o = 10$ definiert, wobei die Selektion der einzelnen Boxgrößen auf Basis des entsprechenden Intervalls innerhalb der Minima bzw. Maxima der Daten ermittelt sind. Für eine

bessere Unterscheidung der geringen Unterschiede in der Performance Analyse wurde die Anzahl der Boxgrößen für diesen zweiten Schritt auf $o = 40$ erhöht.

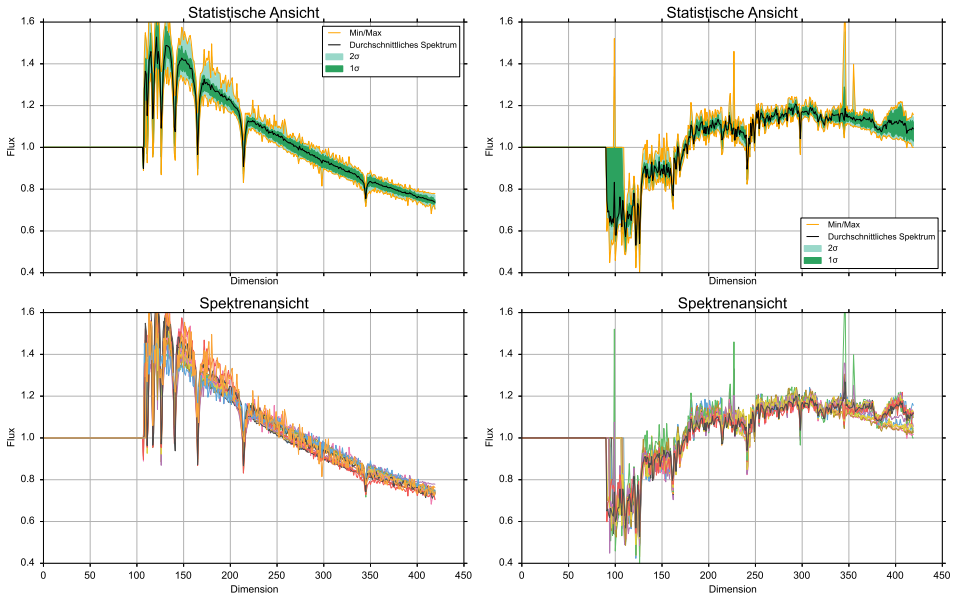
Wie die nachfolgenden Beschreibungen skizzieren, entsteht durch das Fractal Similarity Measures Verfahren eine Vielzahl an unterschiedlichen Clustern, weshalb dieses Kapitel nur einen kleinen Auszug an Clustern zeigt. Die folgenden Abbildungen 5.11 und 5.12 demonstrieren kleinere Cluster mit bis zu 60 Dateninstanzen. Darüber hinaus existieren größere Cluster wie z.B. die in Abbildung 5.15 dargestellten ähnlichen Spektren.

Für die dargestellen Exemplare an Clustern gilt der folgende Aufbau der Abbildungen. Die Abszisse zeigt die Dimensionen in Form einer Nummerierung, die Ordinate die normierten und vorprozessierten Flux-Werte. Eine zweigeteilte Ansicht der Cluster ermöglicht mit Hilfe der Spektrenansicht visuelle Untersuchung der Cluster. In der Spektrenansicht ist jeweils ein Spektrum durch eine eigene Farbe repräsentiert. Die statistische Ansicht markiert orange gefärbt die Extrema der jeweiligen Dimension, während die grünlich markierten Bereiche Konfidenzintervalle zeigen und die durchschnittliche Cluster Spektrum schwarz repräsentiert ist. Die Kurzbeschreibungen der Abbildungen liefern Angaben zu spezifischen Eigenschaften der einzelnen Cluster sowie eine Angabe über die Anzahl der Dateninstanzen bzw. Spektren.



(a) F-Stern ($n = 58$) **(b)** Galaxie Typ 26 ($n = 20$)

Abbildung 5.11: Beispiel Cluster

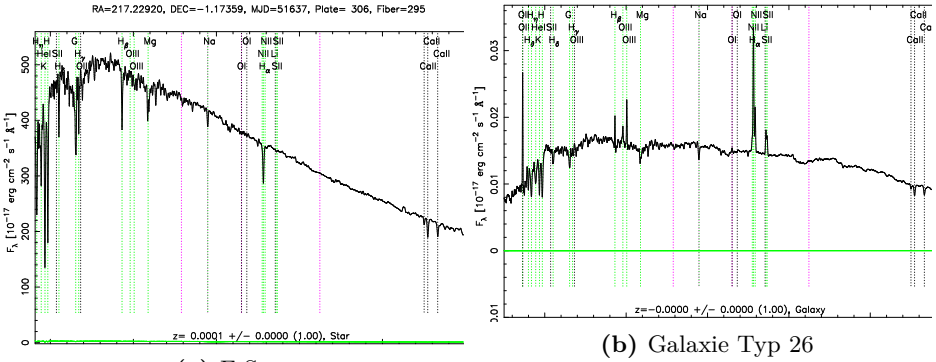


(c) Weißer Zwerg ($n = 21$) **(d)** Galaxie Typ 25 ($n = 19$)

Abbildung 5.12: Weitere exemplarische Cluster

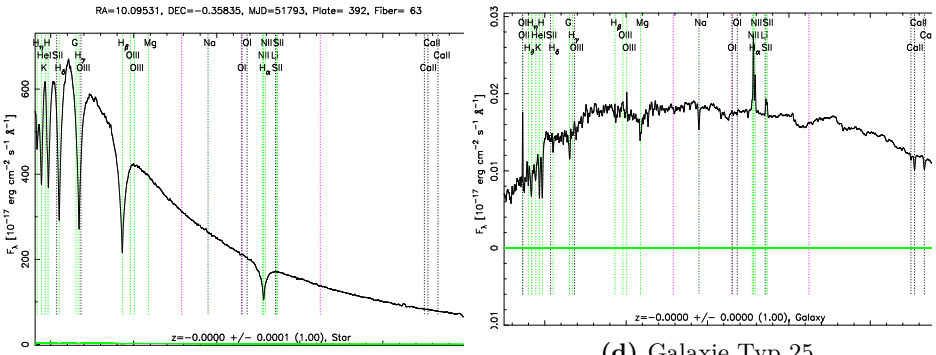
Angemerkt sei, dass die Darstellung der Spektren zur Etablierung einer Vergleichbarkeit entsprechend dem Clustering-Ansatz normiert erfolgt. Dies bedeutet implizit, dass sich bei einer Visualisierung der Verlauf der Daten aus den Dimensionen von einer nicht-normierten Darstellung minimal unterscheiden kann. Die in Abbildung 5.11 und 5.12 visualisierten Cluster könne als unterschiedliche Typen an Objekten klassifiziert werden. Sehr hohe Unsicherheiten ($1/\sigma^2 = 0$) sind dabei in den Abbildungen durch den konstanten Fluxwert 1 gekennzeichnet. Um die Spektren in dieser Arbeit darstellen zu können, wurde ein Binningfaktor für die Spektren in Abbildung 5.11 und 5.12 mit $b = 10$ selektiert und alle weiteren Spektren mit $b = 10$ dargestellt.

Innerhalb des SDSS Projekts wurden manuell Vorlagespektren von Domänenexperten erstellt, die einzelne Objekttypen auszeichnen. Abbildung 5.13 zeigt auszugsweise vier der 33 zur Verfügung gestellten Vorlagespektren [139]. Die identifizierenden Buchstaben der Abbildung 5.11 und 5.12 (a bis d) korrespondieren mit den Buchstaben aus Abbildung 5.13. In den Spektren bzw. Clustern sind hohe Ähnlichkeiten im Verlauf korrespondierender Abbildungen zu erkennen. Diese sind bei einem Vergleich des durchschnittlichen Cluster Spektrums mit den Vorlagespektren auszumachen. Beispielweise ist eine bzgl. der Normierung relative Ähnlichkeit zwischen Abbildung 5.11a und 5.13a ersichtlich, sodass Absorptionslinien mit einem ähn-



(a) F-Stern

(b) Galaxie Typ 26



(c) Weißer Zwerg

(d) Galaxie Typ 25

Abbildung 5.13: SDSS Vorlagespektren, entnommen aus [139]

lichen Muster auftauchen. Dabei ist zu beachten, dass Abbildung 5.13a die Spektren in einer datenoptimierten Form darstellt, die ausschließlich Bereiche mit den typspezifischen Daten fokussiert, während Abbildung 5.11a den gesamten vom Vorprozessierungsverfahren vorgegebenen Messbereich der Observation visualisiert. Dies kann aus Abbildung 5.11a im Dimensionsbereich $1 \leq d \leq 100$ entnommen werden, der ausschließlich aus fehlenden Messwerten oder hohen Unsicherheiten besteht.

Ein derartig großes Intervall an Unsicherheiten kann durch eine hohe Rotverschiebung (d.h. durch die Vorprozessierung) oder durch hohe inverse Varianzen bei diesen observierten Objekten hervorgerufen werden. Rotverschiebungen resultieren nach einer entsprechenden Korrektur (siehe Kapitel 3.4.2) im Fall größerer Wellenlängen am Ende des Messintervalls in einer hohen Unsicherheit in den höheren Dimensionen in einem vorprozessierten Spektrum.

Neben diesen beiden bzgl. der Vorlagentypen kategorisierten F-Stern Spektren, existieren weitere ähnliche durchschnittliche Cluster-Spektren im Vergleich mit den Vorlagespektren wie z.B. ein weißer Zwerg in Abbil-

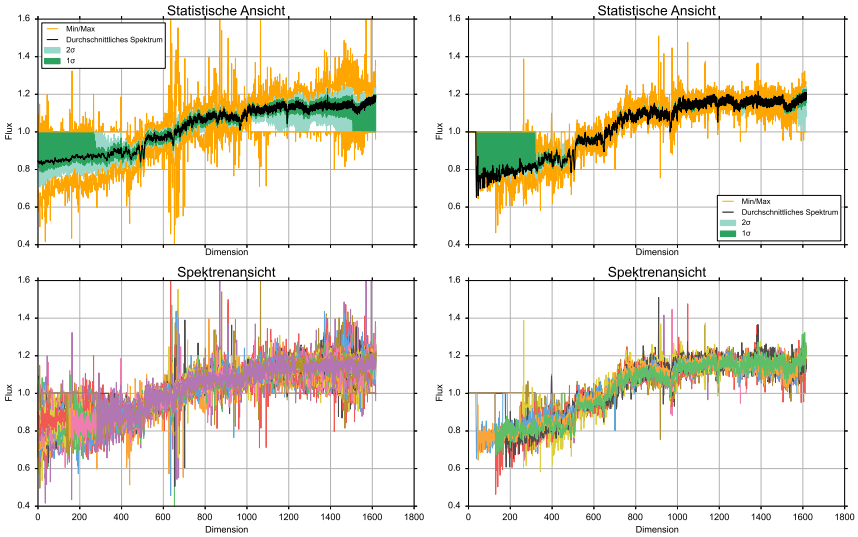
dung 5.13c oder Galaxie Typ 25 in den Abbildungen 5.13d.

Diese Teilergebnisse des Clusterings stellen dar, dass das Fractal Similarity Measures Verfahren eine wertvolle Unterstützung für Domänenexperten bei der Suche nach Klassen, Typen oder Kategorien an observierten Objekten sowie deren Ausreißer sein kann, da das Verfahren automatisiert Cluster findet, die ähnliche oder identische durchschnittliche Spektren bzw. Vorlagespektren repräsentieren.

Die Abbildungen 5.14 repräsentieren zwei Beispiele von Spektren mit einer hohen Anzahl an Dimensionen, die sowohl unsichere als auch weniger unsichere Daten beinhalten. Im Fall von Abbildung 5.14a wird dies im Dimensionsintervall $0 \leq d \leq 400$ sowie $1 \times 10^3 \leq d \leq 1.8 \times 10^3$ deutlich. Die beiden Bereiche weisen in der statistischen Ansicht bei vielen Spektren eine große Unsicherheit auf, was an der hohen Varianz bis zur Visualisierungskonstante der Unsicherheiten zu erkennen ist. In den restlichen Dimensionsintervallen des Clusters existieren keine vergleichbar großen Varianzen. Abbildung 5.14b zeigt ein Cluster mit einer geringeren Anzahl an Unsicherheiten. Die Unterscheidung dieser beiden Cluster ist in den Dimensionen $0 \leq d \leq 400$ möglich. Gegeben durch die errechneten Boxgrößen existiert ein Unterschied im Verlauf sowie in den Rauschverhalten der durchschnittlichen Graphen beider Cluster.

Abbildung 5.15 zeigt zwei Beispiele mit unterschiedlichen Ausprägungen in der Varianzverteilung über die Dimensionen. Während die Varianzen in Abbildung 5.15a variieren, existiert im Cluster aus Abbildung 5.15b eine breitere, aber fast konstante Verteilung der Varianzen in den Dimensionen.

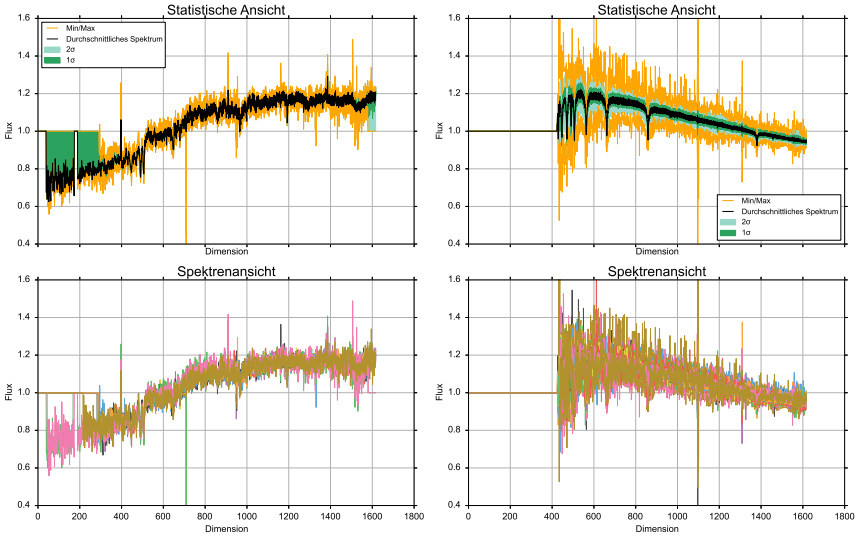
In den dargestellten Clustern ist ersichtlich, dass das Fractal Similarity Measures Verfahren Cluster aus Daten erstellen kann, die Unsicherheiten aufweisen. Cluster enthalten, durch die Boxgrößen definierten und durch die Gitter repräsentierten Rahmen an möglichen Abweichungen zwischen einzelnen Spektren in einzelnen Dimensionen. Durch diese Funktionalität und die inhärenten Dateneigenschaften der Spektren entstehen im Clustering-Prozess Cluster, die auch eine Flexibilität bzw. Varianz in der Dimensionsachse erlaubt. Das bedeutet, dass eventuelle Messfehler in der Rotverschiebung abgefangen werden. Die Spektren eines Clusters besitzen innerhalb der vorgegebenen erlaubten Varianz alle einen ähnlichen Verlauf der Messwerte über die Dimensionen hinweg. Durch die beiden zuvor Sätzen beschriebenen Eigenschaften ist es möglich, dass Cluster in der Spektrenansicht teilweise um einzelne Dimensionen verschoben sind, was die visuelle Unterscheidung bei einer großen Anzahl an Dateninstanzen erschwert. Jedoch ist in der statistischen Ansicht eine vereinfachte Unterscheidung durch die gegebenen durchschnittlichen Spektren möglich. An dieser Stelle sei darauf hingewiesen, dass zum Zeitpunkt der Erstellung dieser Arbeit keine Metrik oder kein Verfahren existiert, das Spektraldaten der Astronomie im Allgemeinen mit einem definierten Fehlermaß analysiert, weshalb die Ergebnisse nicht objektiv bzw. quantitativ überprüft werden können.



(a) Cluster mit hoher Anzahl unsicherer Daten ($n = 196$)

(b) Cluster mit unsicheren Daten ($n = 76$)

Abbildung 5.14: Cluster-Beispiele mit signifikanter Anzahl unsicherer Daten in verschieden Dimensionen. Hohe Unsicherheiten (sehr niedrige inverse Varianzen mit $1/\sigma^2 \approx 0$)



(a) Cluster variabler Varianzen ($n = 50$)

(b) Cluster konstanter Varianzen ($n = 168$)

Abbildung 5.15: Cluster-Beispiele mit verschiedenen Ausprägungen an Varianzen in verschiedenen Dimensionen.

Kapitel 6

Zusammenfassung und Ausblick

6.1 Zusammenfassung

Diese Arbeit stellt einen neuen Ansatz vor, um eine unüberwachte Ähnlichkeitsuntersuchung komplexer Datensätze zu ermöglichen und stellt dafür ein Clustering Verfahren sowie ein Verfahren zur Vorprozessierung von Spektraldaten in der Astronomie zur Verfügung.

Datensätze aus dem industriellen, wissenschaftlichen Bereich insbesondere aus der Astronomie weisen komplexe Eigenschaften auf. Diese Komplexität ist repräsentiert durch mehrere Aspekte. Zum einen erstellen die jeweiligen Datenquellen ein stetig größer werdendes Volumen an Daten in Form einer enormen Anzahl an Dateninstanzen. Zusätzlich bestehen diese Dateninstanzen aus einer Vielzahl an Dimensionen, die adaptierte Analysemethoden und Vorverarbeitungsprozesse zur Vergleichbarkeit erfordern. Eine hochdimensionale Untersuchung weist neben der Berechnungskomplexität bzgl. der Anzahl der Dateninstanzen sowie der Anzahl der Dimensionen einen weitere Aspekt auf.

In der Regel existieren Unsicherheiten in den hochdimensionalen Datensätzen, die in den Clustering Verfahren zu berücksichtigen und zu integrieren sind. Kapitel 2 zeigt die drei genannten Aspekte und diskutiert die dazugehörigen Herausforderungen, die ein Clustering Verfahren bei der Analyse komplexer Datensätze beachten sollte. Darüber hinaus skizziert das Kapitel ausgewählte Repräsentanten einer Unterteilung der aktuellen Clustering Verfahren. Die dargestellten Verfahren zeigen die Behandlung einzelner Aspekte bei der Analyse komplexer Datensätze, jedoch nicht einer Kombination aller Herausforderungen.

Als zentraler Anwendungsfall, des in dieser Arbeit vorgestellten Clustering Verfahrens, dient ein öffentlich verfügbarer Datensatz bestehend aus Spektraldaten. Diese resultieren aus dem Sloan Digitial Sky Survey (SDSS)

Projekt und besitzen neben den eigentlichen Messwerten der einzelnen observierten Objekten auch quantifizierte Unsicherheiten in Form von inversen Varianzen. Kapitel 3 diskutiert die Eigenschaften des Datensatzes und skizziert kurz die Methodik der Datenerfassung und Kalibrierung der Daten. Durch die Rotverschiebungen und die unterschiedlichen Messwertbereiche in den Daten ist eine Vorprozessierung der Daten, die eine Vergleichbarkeit der Spektren untereinander herstellt, notwendig. Aktuelle Datamining Vorhaben nutzen zur Vereinfachung der Entwicklung der Datenverarbeitung das Rahmenwerk Hadoop. Das Kapitel skizziert das Rahmenwerk sowie dessen Komponenten und diskutiert das entwickelte, darauf aufbauendes Verfahren zur Vorprozessierung der SDSS Spektren.

Der im Rahmen dieser Dissertation entwickelte Ansatz des Fractal Similarity Measures Verfahren stellt Kapitel 4 vor. Dieses Verfahren basiert auf dem Konzept der Fraktalen Dimensionen und arbeitet mit einer multiplen Diskretisierung der Eingangdaten zur Erstellung mehrerer Gitter. Die verschiedenen Gitter beinhalten unterschiedlich skalierte Ansichten auf die Daten, die mit Hilfe des in dieser Arbeit vorgestellten Fraktalen Informationswerts, des Fraktalen Ähnlichkeitsmaßes und einem zweigeteilten Algorithmus analysiert werden kann. Das Kapitel beschreibt die theoretischen Grundlagen sowie eine abstrakte Darstellung der Algorithmik.

Komplexitätsannahmen sowie die Funktionalität des Fractal Similarity Measures Verfahrens sind Bestandteil einer Evaluation in Kapitel 5. Diese besteht aus der isolierten Untersuchung der maßgeblichen Komplexitätsanteile der zweigeteilten Algorithmik mit Hilfe eines definierten synthetischen Datensatzes. Die Skalierbarkeit des Verfahrens wird darüber hinaus exemplarisch durch die Prozessierung des SDSS Datensatzes vorgestellt. Auszüge aus den Ergebnissen der Prozessierung belegen die Funktionalität des Verfahrens. Insgesamt zeigt das Verfahren durch die Verarbeitung des SDSS Datensatzes und die daraus resultierenden Ergebnisse, dass das Verfahren in der Lage ist komplexe Datensätze in akzeptabler Zeit zu verarbeiten.

6.2 Ausblick

Der in dieser Dissertation vorgestellte Ansatz zur Analyse von hochdimensionalen und unsicheren Spektraldaten sowie dessen Evaluation stellt eine grundlegende Arbeit dar, die als Basis für Weiterentwicklungen dienen kann. Die folgenden Abschnitte zeigen beispiele möglicher Weiterentwicklungen.

Die Implementierung des Verfahrens der Vorverarbeitung von Spektraldaten ist primär für die Verwendung der SDSS Daten entwickelt, besitzt darüber hinaus die Möglichkeit durch weitere Filter ergänzt zu werden, um weitere Datensätze oder andere Vorverarbeitungsroutinen umzusetzen. Beispielsweise ist es denkbar, einen Cloud-Dienst für Astronomen zu etablieren, der durch eine einfache Modellierung von Vorverarbeitungsprozessen durch

die Verkettung beliebiger Filter ermöglicht.

Das Fractal Similarity Measures Verfahren besitzt aktuell keine Parallelisierung des Verfahrens auf Basis der Dateninstanzen. In einer nicht parallelisierten Version des Verfahren sind keine Synchronisationsschranken nötig, deren Notwendigkeit im Algorithmus erst validiert werden muss. Es ist davon auszugehen, dass eine Parallelisierbarkeit in Form des parallelen Hinzufügens von Dateninstanzen möglich ist, soweit es die atomaren Datenstrukturen, wie z.B. Cluster, dies ermöglichen. Für eine effiziente Umsetzung muss eine klare Unterscheidung zwischen dem testweisen und dadurch temporären bzw. dem permanenten Hinzufügen einer Dateninstanz zu einem Cluster unterschieden werden. Während der temporäre Fall nur eine erweiterte Sicht auf ein Cluster zur Berechnung des Ähnlichkeitswerte auf Basis der Gitterstruktur darstellt, ist das permanente Hinzufügen in jedem Fall mit einer Modifikation der Clusterdaten verbunden. Mit Hilfe einer entsprechend optimierten Implementierung ist es denkbar, eine solche Art der Parallelisierung zu ermöglichen.

Neben der Parallelisierung stellt die Optimierung der Boxgrößensuche eine weitere Entwicklungsmöglichkeit dar. Das in dieser Dissertation vorgestellte Verfahren nutzt einen linearen Anstieg der Boxgrößen innerhalb der datenabhängigen Boxgrößenextrema. Um die Analysedauer des Performance Schritts effizienter zu gestalten, kann untersucht werden, inwieweit die Minimierung der Anzahl der Boxgrößen bei gleichzeitig spezifischer Wahl der Boxgrößen unter Berücksichtigung einer konstanten Fehlerquote die Analysedauer optimiert.

Neben den Weiterentwicklungen besteht die Möglichkeit der Untersuchung weiterer komplexer Datensätze wie z.B. aus dem Bereich der Spracherkennung oder Bildanalyse, wobei ebenfalls ein Vorprozessierungsmethodik zu erstellen ist, die eine Vergleichbarkeit der Daten sichert.

Quellenverzeichnis

Literatur

[1] Hervé Abdi und Lynne J Williams. „Principal component analysis“. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 2.4 (2010), S. 433–459 (siehe S. 7).

[2] Charu C Aggarwal, Alexander Hinneburg und Daniel A Keim. *On the surprising behavior of distance metrics in high dimensional space*. Springer, 2001 (siehe S. 9, 10).

[3] Charu C Aggarwal und Chandan K Reddy. *Data clustering: algorithms and applications*. CRC Press, 2013 (siehe S. 9, 15, 17, 24).

[4] Charu C Aggarwal und Philip S Yu. „A survey of uncertain data algorithms and applications“. In: *Knowledge and Data Engineering, IEEE Transactions on* 21.5 (2009), S. 609–623 (siehe S. 14).

[5] Rakesh Agrawal u. a. *Automatic subspace clustering of high dimensional data for data mining applications*. Bd. 27. 2. ACM, 1998 (siehe S. 28).

[6] Thomas Alrutz u. a. „GASPI–A partitioned global address space programming interface“. In: *Facing the Multicore-Challenge III*. Springer, 2013, S. 135–136 (siehe S. 84).

[7] Michael R Anderberg. *Cluster Analysis for Applications: Probability and Mathematical Statistics: A Series of Monographs and Textbooks*. Bd. 19. Academic press, 2014 (siehe S. 33).

[8] Mihael Ankerst u. a. „OPTICS: ordering points to identify the clustering structure“. In: *ACM Sigmod Record*. Bd. 28. 2. ACM. 1999, S. 49–60 (siehe S. 15).

[9] David Arthur und Sergei Vassilvitskii. „k-means++: The advantages of careful seeding“. In: *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial und Applied Mathematics. 2007, S. 1027–1035 (siehe S. 17).

[10] Duane A Bailey. *Java structures: Data structures in Java for the principled programmer*. McGraw-Hill Science/Engineering/Math, 2002 (siehe S. 86).

[11] Daniel Barbara und Ping Chen. „Fractal mining-self similarity-based clustering and its applications“. In: *Data Mining and Knowledge Discovery Handbook*. Springer, 2010, S. 573–589 (siehe S. 29, 61).

[12] Daniel Barbará und Ping Chen. „Using the fractal dimension to cluster datasets“. In: *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2000, S. 260–264 (siehe S. 29, 61).

[13] Richard Bellman u. a. *Adaptive control processes: a guided tour*. Bd. 4. Princeton university press Princeton, 1961 (siehe S. 6).

[14] Kevin Beyer u. a. „When is “nearest neighbor” meaningful?“ In: *Database Theory—ICDT'99*. Springer, 1999, S. 217–235 (siehe S. 6, 9, 36).

[15] James C Bezdek. *Pattern recognition with fuzzy objective function algorithms*. Springer Science & Business Media, 2013 (siehe S. 18).

[16] James C Bezdek, Robert Ehrlich und William Full. „FCM: The fuzzy c-means clustering algorithm“. In: *Computers & Geosciences* 10.2 (1984), S. 191–203 (siehe S. 12).

[17] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006 (siehe S. 7).

[18] Adam S Bolton u. a. „Spectral classification and redshift measurement for the SDSS-III Baryon Oscillation Spectroscopic Survey“. In: *The Astronomical Journal* 144.5 (2012), S. 144 (siehe S. 66).

[19] Dhruba Borthakur. „HDFS architecture guide“. In: *HADOOP APACHE PROJECT http://hadoop. apache. org/common/docs/current/hdfs design. pdf* (2008) (siehe S. 39).

[20] Paul S Bradley und Usama M Fayyad. „Refining Initial Points for K-Means Clustering.“ In: Citeseer. 1998 (siehe S. 17).

[21] D Britton und SL Lloyd. „How to deal with petabytes of data: the LHC Grid project“. In: *Reports on Progress in Physics* 77.6 (2014), S. 065902 (siehe S. 1).

[22] Tadeusz Caliński und Jerzy Harabasz. „A dendrite method for cluster analysis“. In: *Communications in Statistics-theory and Methods* 3.1 (1974), S. 1–27 (siehe S. 17).

[23] Feng Cao u. a. „Density-Based Clustering over an Evolving Data Stream with Noise.“ In: *SDM*. Bd. 6. SIAM. 2006, S. 328–339 (siehe S. 65).

[24] Chung-I Chang, Nancy P Lin und Nien-Yi Jan. „An axis-shifted grid-clustering algorithm“. In: *Tamkang Journal of Science and Engineering* 12.2 (2009), S. 183–192 (siehe S. 27).

[25] Barbara Chapman u. a. „Introducing OpenSHMEM: SHMEM for the PGAS community“. In: *Proceedings of the Fourth Conference on Partitioned Global Address Space Programming Model*. ACM. 2010, S. 2 (siehe S. 84).

[26] Chun-Hung Cheng, Ada Waichee Fu und Yi Zhang. „Entropy-based subspace clustering for mining numerical data“. In: *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 1999, S. 84–93 (siehe S. 28).

[27] Mark Ming-Tso Chiang und Boris Mirkin. „Experiments for the number of clusters in k-means“. In: *Progress in Artificial Intelligence*. Springer, 2007, S. 395–405 (siehe S. 18).

[28] Mark Ming-Tso Chiang und Boris Mirkin. „Intelligent choice of the number of clusters in K-Means clustering: an experimental study with different cluster spreads“. In: *Journal of classification* 27.1 (2010), S. 3–40 (siehe S. 18).

[29] HAOEN CHUEH, HUNG-JEN CHEN und WEIHUA HAO. „An Axis-shifted Crossover-Imaged Clustering Algorithm“. In: () (siehe S. 27).

[30] Richard Cole. „Parallel merge sort“. In: *SIAM Journal on Computing* 17.4 (1988), S. 770–785 (siehe S. 86).

[31] Aura Conci und Claudia Belmiro Proença. „A fractal image analysis system for fabric inspection based on a box-counting method“. In: *Computer Networks and ISDN Systems* 30.20 (1998), S. 1887–1895 (siehe S. 57).

[32] Thomas M Cover und Peter E Hart. „Nearest neighbor pattern classification“. In: *Information Theory, IEEE Transactions on* 13.1 (1967), S. 21–27 (siehe S. 24, 25).

[33] Jeffrey Dean und Sanjay Ghemawat. „MapReduce: a flexible data processing tool“. In: *Communications of the ACM* 53.1 (2010), S. 72–77 (siehe S. 42).

[34] Jeffrey Dean und Sanjay Ghemawat. „MapReduce: simplified data processing on large clusters“. In: *Communications of the ACM* 51.1 (2008), S. 107–113 (siehe S. 41, 42).

[35] Konstantinos G Derpanis. „Mean shift clustering“. In: *Lecture Notes. http://www. cse. yorku. ca/˜ kosta/CompVis_Notes/mean_shift. pdf* (2005) (siehe S. 19).

[36] PE Dewdney u. a. „SKA1: High Level System Description“. In: *SKA Program Development Office, Tech. Rep* (2011) (siehe S. 3).

[37] Inderjit S Dhillon, Yuqiang Guan und Brian Kulis. „Kernel k-means: spectral clustering and normalized cuts“. In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2004, S. 551–556 (siehe S. 19).

[38] Antonio Di Ieva u. a. „Fractals in the neurosciences, part I general principles and basic neurosciences“. In: *The Neuroscientist* 20.4 (2014), S. 403–417 (siehe S. 57).

[39] Jens Dittrich und Jorge-Arnulfo Quiané-Ruiz. „Efficient big data processing in Hadoop MapReduce“. In: *Proceedings of the VLDB Endowment* 5.12 (2012), S. 2014–2015 (siehe S. 42).

[40] Kieran Jay Edwards und Mohamed Medhat Gaber. *Astronomy and Big Data*. Springer, 2014 (siehe S. 4).

[41] Laszlo Engelman und John A Hartigan. „Percentage points of a test for clusters“. In: *Journal of the American Statistical Association* 64.328 (1969), S. 1647–1648 (siehe S. 52).

[42] Martin Ester u. a. „A density-based algorithm for discovering clusters in large spatial databases with noise.“ In: *Kdd*. Bd. 96. 34. 1996, S. 226–231 (siehe S. 9, 15).

[43] B.S. Everitt u. a. *Cluster Analysis*. Wiley Series in Probability and Statistics. Wiley, 2011 (siehe S. 22).

[44] Kenneth Falconer. *Fractal geometry: mathematical foundations and applications*. John Wiley & Sons, 2004 (siehe S. 56, 57).

[45] Fredrik Farnstrom, James Lewis und Charles Elkan. „Scalability for clustering algorithms revisited“. In: *ACM SIGKDD Explorations Newsletter* 2.1 (2000), S. 51–57 (siehe S. 65).

[46] Paolo Fornasini. *The uncertainty in physical measurements: an introduction to data analysis in the physics laboratory*. Springer Science & Business Media, 2008 (siehe S. 13).

[47] K Foroutan-Pour, Pierre Dutilleul und DL Smith. „Advances in the implementation of the box-counting method of fractal dimension estimation“. In: *Applied mathematics and computation* 105.2 (1999), S. 195–210 (siehe S. 57).

[48] Guojun Gan, Chaoqun Ma und Jianhong Wu. *Data clustering: theory, algorithms, and applications*. Bd. 20. Siam, 2007 (siehe S. 33).

[49] Inc. Gartner. „Hype Cycle for Emerging Technologies, 2014“. In: (2014) (siehe S. 1).

[50] Tilmann Gneiting und Martin Schlather. „Stochastic models that separate fractal dimension and the Hurst effect“. In: *SIAM review* 46.2 (2004), S. 269–282 (siehe S. 57).

[51] Tilmann Gneiting, Donald B Percival u. a. „Estimators of fractal dimension: Assessing the roughness of time series and spatial data“. In: *Statistical Science* 27.2 (2012), S. 247–277 (siehe S. 55, 56).

[52] Sanjay Goil, Harsha Nagesh und Alok Choudhary. „MAFIA: Efficient and scalable subspace clustering for very large data sets“. In: *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 1999, S. 443–452 (siehe S. 28).

[53] Ronald L Graham. *Concrete mathematics:[a foundation for computer science; dedicated to Leonhard Euler (1707-1783)]*. Pearson Education India, 1994 (siehe S. 78).

[54] Peter Grassberger. „Generalized dimensions of strange attractors“. In: *Physics Letters A* 97.6 (1983), S. 227–230 (siehe S. 57, 58).

[55] Sudipto Guha, Rajeev Rastogi und Kyuseok Shim. „CURE: an efficient clustering algorithm for large databases“. In: *ACM SIGMOD Record*. Bd. 27. 2. ACM. 1998, S. 73–84 (siehe S. 24).

[56] Sudipto Guha, Rajeev Rastogi und Kyuseok Shim. „Cure: an efficient clustering algorithm for large databases“. In: *Information Systems* 26.1 (2001), S. 35–58 (siehe S. 24).

[57] James E Gunn u. a. „The 2.5 m telescope of the sloan digital sky survey“. In: *The Astronomical Journal* 131.4 (2006), S. 2332 (siehe S. 33).

[58] Donald Gustafson und William Kessel. „Fuzzy clustering with a fuzzy covariance matrix“. In: *1978 IEEE conference on decision and control including the 17th symposium on adaptive processes*. 17. 1978, S. 761–766 (siehe S. 12).

[59] Isabelle Guyon u. a. *Feature extraction: foundations and applications*. Bd. 207. Springer, 2008, S. 265–313 (siehe S. 32).

[60] Maria Halkidi, Yannis Batistakis und Michalis Vazirgiannis. „Cluster validity methods: part I“. In: *ACM Sigmod Record* 31.2 (2002), S. 40–45 (siehe S. 11).

[61] Maria Halkidi, Yannis Batistakis und Michalis Vazirgiannis. „Clustering validity checking methods: part II“. In: *ACM Sigmod Record* 31.3 (2002), S. 19–27 (siehe S. 11).

[62] Maria Halkidi, Yannis Batistakis und Michalis Vazirgiannis. „On clustering validation techniques“. In: *Journal of Intelligent Information Systems* 17.2 (2001), S. 107–145 (siehe S. 11).

[63] John A Hartigan und Manchek A Wong. „Algorithm AS 136: A k-means clustering algorithm“. In: *Applied statistics* (1979), S. 100–108 (siehe S. 17).

[64] Felix Hausdorff. „Dimension und äußeres Maß“. In: *Mathematische Annalen* 79.1-2 (1918), S. 157–179 (siehe S. 56).

[65] Magnus Lie Hetland. *Python Algorithms: Mastering Basic Algorithms in the Python Language*. Apress, 2010 (siehe S. 78, 86).

[66] Alexander Hinneburg, Charu C Aggarwal und Daniel A Keim. „What is the nearest neighbor in high dimensional spaces?“ In: (2000) (siehe S. 10).

[67] Alexander Hinneburg und Daniel A Keim. „Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering“. In: (1999) (siehe S. 28).

[68] Maximilian Hoecker u. a. „Clustering of Complex Data-Sets Using Fractal Similarity Measures and Uncertainties“. In: *Computational Science and Engineering (CSE), 2015 IEEE 18th International Conference on*. Okt. 2015, S. 82–91 (siehe S. 53).

[69] Joshua Zhexue Huang u. a. „Automated Variable Weighting in k-Means Type Clustering“. In: *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE* 27.5 (2005), S. 657 (siehe S. 19).

[70] Lassi Hyvärinen. „Principal component analysis“. In: *Mathematical Modeling for Industrial Processes*. Springer, 1970, S. 82–104 (siehe S. 7).

[71] Moriteru Ishida, Hiroki Takakura und Yasuo Okabe. „High-performance intrusion detection using optigrid clustering and grid-based labelling“. In: *Applications and the Internet (SAINT), 2011 IEEE/IPSJ 11th International Symposium on*. IEEE. 2011, S. 11–19 (siehe S. 28).

[72] Anil K Jain. „Data clustering: 50 years beyond K-means“. In: *Pattern recognition letters* 31.8 (2010), S. 651–666 (siehe S. 33).

[73] Anil K Jain, M Narasimha Murty und Patrick J Flynn. „Data clustering: a review“. In: *ACM computing surveys (CSUR)* 31.3 (1999), S. 264–323 (siehe S. 9, 52).

[74] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002 (siehe S. 7).

[75] Alexandros Kalousis, Julien Prados und Melanie Hilario. „Stability of feature selection algorithms: a study on high-dimensional spaces“. In: *Knowledge and information systems* 12.1 (2007), S. 95–116 (siehe S. 7).

[76] George Karypis, Eui-Hong Han und Vipin Kumar. „Chameleon: Hierarchical clustering using dynamic modeling“. In: *Computer* 32.8 (1999), S. 68–75 (siehe S. 24–26).

[77] George Karypis und Vipin Kumar. „Metis-unstructured graph partitioning and sparse matrix ordering system, version 2.0“. In: (1995) (siehe S. 25).

[78] L Kaufman und PJ Rousseeuw. „Finding groups in data, 1990“. In: *New York* (1990) (siehe S. 22, 52).

[79] Hans-Peter Kriegel, Peer Kröger und Arthur Zimek. „Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering“. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 3.1 (2009), S. 1 (siehe S. 7, 9, 65).

[80] Hans-Peter Kriegel und Martin Pfeifle. „Density-based clustering of uncertain data“. In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. ACM. 2005, S. 672–677 (siehe S. 14).

[81] K Krishna und M Narasimha Murty. „Genetic K-means algorithm“. In: *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on* 29.3 (1999), S. 433–439 (siehe S. 19).

[82] SD Kuegler, K Polsterer und M Hoecker. „Determining spectroscopic redshifts by using k nearest neighbor regression-I. Description of method and analysis“. In: *Astronomy & Astrophysics* 576 (2015), A132 (siehe S. 48).

[83] K Ashwin Kumar u. a. „Hone: Scaling down Hadoop on shared-memory systems“. In: *Proceedings of the VLDB Endowment* 6.12 (2013), S. 1354–1357 (siehe S. 42).

[84] Marc Lavielle und Kevin Bleakley. „Automatic data binning for improved visual diagnosis of pharmacometric models“. In: *Journal of pharmacokinetics and pharmacodynamics* 38.6 (2011), S. 861–871 (siehe S. 70).

[85] Jian Li, Qian Du und Caixin Sun. „An improved box-counting method for image fractal dimension estimation“. In: *Pattern Recognition* 42.11 (2009), S. 2460–2469 (siehe S. 57).

[86] Wei-keng Liao, Ying Liu und Alok Choudhary. „A grid-based clustering algorithm using adaptive mesh refinement“. In: *7th Workshop on Mining Scientific and Engineering Datasets of SIAM International Conference on Data Mining*. 2004, S. 61–69 (siehe S. 27).

[87] Larry S Liebovitch und Tibor Toth. „A fast algorithm to determine fractal dimensions by box counting“. In: *Physics Letters A* 141.8 (1989), S. 386–390 (siehe S. 57).

[88] Huan Liu und Hiroshi Motoda. *Feature selection for knowledge discovery and data mining*. Bd. 454. Springer Science & Business Media, 2012, S. 43–90 (siehe S. 32).

[89] Stuart P Lloyd. „Least squares quantization in PCM“. In: *Information Theory, IEEE Transactions on* 28.2 (1982), S. 129–137 (siehe S. 16).

[90] Yi Lu u. a. „FGKA: A fast genetic k-means clustering algorithm“. In: *Proceedings of the 2004 ACM symposium on Applied computing*. ACM. 2004, S. 622–623 (siehe S. 19).

[91] Eden WM Ma und Tommy WS Chow. „A new shifting grid clustering algorithm“. In: *Pattern Recognition* 37.3 (2004), S. 503–514 (siehe S. 27).

[92] James MacQueen u. a. „Some methods for classification and analysis of multivariate observations“. In: *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*. Bd. 1. 14. Oakland, CA, USA. 1967, S. 281–297 (siehe S. 16).

[93] Benoit B Mandelbrot. *The fractal geometry of nature*. Bd. 173. Macmillan, 1983 (siehe S. 55, 56).

[94] Dirk Merkel. „Docker: lightweight linux containers for consistent development and deployment“. In: *Linux Journal* 2014.239 (2014), S. 2 (siehe S. 44).

[95] Harvey J Miller und Jiawei Han. *Geographic data mining and knowledge discovery*. CRC Press, 2009 (siehe S. 9, 16).

[96] Glenn W Milligan. „A monte carlo study of thirty internal criterion measures for cluster analysis“. In: *Psychometrika* 46.2 (1981), S. 187–199 (siehe S. 11, 17).

[97] Tood K Moon. „The expectation-maximization algorithm“. In: *Signal processing magazine, IEEE* 13.6 (1996), S. 47–60 (siehe S. 16).

[98] Fionn Murtagh. „A survey of recent advances in hierarchical clustering algorithms“. In: *The Computer Journal* 26.4 (1983), S. 354–359 (siehe S. 33).

[99] Harsha S Nagesh, Sanjay Goil und Alok N Choudhary. „Adaptive Grids for Clustering Massive Data Sets.“ In: *SDM*. SIAM. 2001, S. 1–17 (siehe S. 28).

[100] Mark EJ Newman und Michelle Girvan. „Finding and evaluating community structure in networks“. In: *Physical review E* 69.2 (2004), S. 026113 (siehe S. 17).

[101] Ray P Norris u. a. „The SKA Mid-frequency All-sky Continuum Survey: Discovering the unexpected and transforming radio-astronomy“. In: *arXiv preprint arXiv:1412.6076* (2014) (siehe S. 3).

[102] William L Oberkampf u. a. „Error and uncertainty in modeling and simulation“. In: *Reliability Engineering & System Safety* 75.3 (2002), S. 333–357 (siehe S. 13).

[103] Chris Richardson. „Untangling enterprise java“. In: *Queue* 4.5 (2006), S. 36–44 (siehe S. 80).

[104] ALFRPED Rrnyi. „On measures of entropy and information“. In: *Fourth Berkeley symposium on mathematical statistics and probability*. Bd. 1. 1961, S. 547–561 (siehe S. 58).

[105] Leslie Rutkowski. „Clustering for data mining: A data recovery approach“. In: *Psychometrika* 72.1 (2007), S. 109–110 (siehe S. 18, 19).

[106] Seref Sagiroglu und Duygu Sinanc. „Big data: A review“. In: *Collaboration Technologies and Systems (CTS), 2013 International Conference on*. IEEE. 2013, S. 42–47 (siehe S. 1).

[107] Sherif Sakr u. a. „Big Data Processing Systems: State-of-the-Art and Open Challenges“. In: *Cloud Computing (ICCC), 2015 International Conference on*. IEEE. 2015, S. 1–8 (siehe S. 42).

[108] Nirupam Sarkar und BB Chaudhuri. „An efficient differential box-counting approach to compute fractal dimension of image“. In: *Systems, Man and Cybernetics, IEEE Transactions on* 24.1 (1994), S. 115–120 (siehe S. 57).

[109] Bernhard Schölkopf, Alexander Smola und Klaus-Robert Müller. „Nonlinear component analysis as a kernel eigenvalue problem“. In: *Neural computation* 10.5 (1998), S. 1299–1319 (siehe S. 19).

[110] Shokri Z Selim und Mohamed A Ismail. „K-means-type algorithms: a generalized convergence theorem and characterization of local optimality“. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 1 (1984), S. 81–87 (siehe S. 17).

[111] Claude Elwood Shannon. „A mathematical theory of communication“. In: *ACM SIGMOBILE Mobile Computing and Communications Review* 5.1 (2001), S. 3–55 (siehe S. 58).

[112] Stephen Smee u. a. „The multi-object, fiber-fed spectrographs for SDSS and the Baryon Oscillation Spectroscopic Survey“. In: *arXiv preprint arXiv:1208.2233* (2012) (siehe S. 33–35).

[113] Chaoming Song, Shlomo Havlin und Hernan A Makse. „Self-similarity of complex networks“. In: *Nature* 433.7024 (2005), S. 392–395 (siehe S. 57).

[114] Chris Stoughton u. a. „Sloan digital sky survey: early data release“. In: *The Astronomical Journal* 123.1 (2002), S. 485 (siehe S. 35).

[115] Christiane Stutz. *Anwendungsspezifische fuzzy-clustermethoden*. Infix, 1999 (siehe S. 18).

[116] Robert Tibshirani, Guenther Walther und Trevor Hastie. „Estimating the number of clusters in a data set via the gap statistic“. In: *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 63.2 (2001), S. 411–423 (siehe S. 17).

[117] Vinod Kumar Vavilapalli u. a. „Apache hadoop yarn: Yet another resource negotiator“. In: *Proceedings of the 4th annual Symposium on Cloud Computing*. ACM. 2013, S. 5 (siehe S. 42, 43).

[118] Michalis Vazirgiannis, Maria Halkidi und Dimitrious Gunopulos. *Uncertainty handling and quality assessment in data mining*. Springer Science & Business Media, 2012 (siehe S. 19).

[119] Tom White. *Hadoop: The definitive guide*. Ö'Reilly Media, Inc.", 2012 (siehe S. 39).

[120] Kuo-Lung Wu und Miin-Shen Yang. „Mean shift-based clustering“. In: *Pattern Recognition* 40.11 (2007), S. 3035–3052 (siehe S. 19).

[121] Rui Xu und Don Wunsch. *Clustering*. Bd. 10. John Wiley & Sons, 2008 (siehe S. 6, 9, 15, 17, 22, 24).

[122] Rui Xu, Donald Wunsch u. a. „Survey of clustering algorithms“. In: *Neural Networks, IEEE Transactions on* 16.3 (2005), S. 645–678 (siehe S. 28).

[123] M-S Yang. „A survey of fuzzy clustering“. In: *Mathematical and Computer modelling* 18.11 (1993), S. 1–16 (siehe S. 12).

[124] Ka Yee Yeung, David R. Haynor und Walter L. Ruzzo. „Validating clustering for gene expression data“. In: *Bioinformatics* 17.4 (2001), S. 309–318 (siehe S. 11).

[125] Tian Zhang, Raghu Ramakrishnan und Miron Livny. „BIRCH: an efficient data clustering method for very large databases“. In: *ACM SIGMOD Record*. Bd. 25. 2. ACM. 1996, S. 103–114 (siehe S. 65).

[126] Andreas Züfle u. a. „Representative clustering of uncertain data“. In: *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2014, S. 243–252 (siehe S. 14).

Online-Quellen

[127] The Scipy community. *scipy.cluster.hierarchy.linkage Manual*. Aug. 2015. URL: http://scipy.github.io/devdocs/generated/scipy.cluster.hierarchy.linkage.html#scipy.cluster.hierarchy.linkage (siehe S. 22).

[128] Apache Software Foundation. *Docker Container Executor in YARN*. Aug. 2015. URL: http://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/DockerContainerExecutor.html (siehe S. 44).

[129] Python Software Foundation. *Python*. Nov. 2015. URL: https://www.python.org/ (siehe S. 95).

[130] he HEASARC (High Energy Astrophysics Science Archive Research Center). *The FITS Support Office*. Aug. 2015. URL: http://fits.gsfc.nasa.gov/ (siehe S. 36).

[131] John Hunter u. a. *matplotlib*. Nov. 2015. URL: http://matplotlib.org/ (siehe S. 95).

[132] Mathworks Inc. *Linkage Documentation*. Aug. 2015. URL: http://de.mathworks.com/help/stats/linkage.html (siehe S. 22).

[133] Fermi National Accelerator Laboratory. *Fermilab*. Aug. 2015. URL: http://www.fnal.gov/ (siehe S. 34).

[134] StataCorp LP. *Stata Cluster Linkage Manual*. Aug. 2015. URL: http://www.stata.com/manuals13/mvclusterlinkage.pdf (siehe S. 22).

[135] SKA. *Square Kilometer Array*. Aug. 2015. URL: https://www.skatelescope.org/ (siehe S. 3).

[136] Sloan Digital Sky Survey. *Sloan Digital Sky Survey*. Aug. 2015. URL: http://www.sdss.org (siehe S. 3, 33).

[137] Sloan Digital Sky Survey. *Spectro2d Pipeline Sourcecode*. Aug. 2015. URL: http://das.sdss.org/software/idlspec2d/v5_3_12/ (siehe S. 35).

[138] Sloan Digital Sky Survey. *The Tenth SDSS Data Release (DR10)*. Aug. 2015. URL: https://www.sdss3.org/dr10/ (siehe S. 33, 36).

[139] Sloan Digital Sky Survey. *The Tenth SDSS Data Release Template Spectra*. Aug. 2015. URL: http://www.sdss.org/dr7/algorithms/spectemplates/index.html (siehe S. 66, 110, 111).