Entwicklung eines Separate-and-Conquer basierten Regellern-Algorithmus für interpretierbares Clustering

Development of a separate-and-conquer based rule learning algorithm for interpretable clustering

Bachelor-Thesis von Tobias Viernickel

Tag der Einreichung:

1. Gutachten: Prof. Dr. Johannes Fürnkranz
2. Gutachten: Dr. Frederik Janssen

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich Informatik
Knowledge Engineering Group

Entwicklung eines Separate-and-Conquer basierten Regellern-Algorithmus für interpretierbares Clustering
Development of a separate-and-conquer based rule learning algorithm for interpretable clustering

Vorgelegte Bachelor-Thesis von Tobias Viernickel

1. Gutachten: Prof. Dr. Johannes Fürnkranz
2. Gutachten: Dr. Frederik Janssen

Tag der Einreichung:

Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 29. September 2015

(T. Viernickel)

Inhaltsverzeichnis

1 Einleitung

Im Bereich des maschinellen Lernens wird versucht, neues Wissen aus Erfahrungen zu erlangen. Durch dieses Vorgehen soll es einer Maschine ermöglicht werden, eine unbekannte Aufgabe auf Basis gesammelter Erfahrungen zu bewältigen. Neben vielen weiteren Disziplinen, wie zum Beispiel der Klassifikation, zählt auch das Clustering zu einer der Hauptaufgaben im Bereich des maschinellen Lernens.
Der Name Clustering wird hierbei verwendet, um Methoden zu beschreiben, die Daten einer vorhandenen Datenmenge in Gruppen zusammenfassen. Die Clustering-Methoden besitzen dabei kein Wissen über einen bestehenden Zusammenhang oder eine Gruppenzugehörigkeit dieser Daten. Die Clusteranalyse ist somit eine fundamentale Anwendung der Datenanalyse und gehört zu dem Bereich des unüberwachten Lernens. Neben den Einsatzgebieten in der Mustererkennung [2][3], in der Bildverarbeitung [32],und in der Bioinformatik [14] hat die Clusteranalyse eine reichhaltige Vergangenheit in anderen Bereichen [16], wie zum Beispiel der Biologie, der Geologie, der Geographie und im Marketing [17].
Andere Verfahren, die in der Literatur auftauchen und der Clusteranalyse sehr ähneln [17], sind unüberwachtes Lernen [16], numerische Taxonomie [31], Vektorquantisierung [27], und Lernen durch Beobachtung [26]
Das Ziel der Clusteranalyse ist es, die in den Daten vorhandenen natürlichen Strukturen zu erkennen und wiederzugeben. Da hierbei kein Wissen über Zusammengehörigkeit von Teilmengen der Datenmenge zur Verfügung steht, werden bei der Clusteranalyse die Instanzen oder auch Objekte der Datenmenge zu sogenannten Clustern zusammengefasst. Ein Clustering-Algorithmus untersucht dabei die gegebenen Daten und weist ähnlichen Objekten das gleiche Cluster und unähnlichen Objekten unterschiedliche Cluster zu. Somit wird erreicht, dass alle Objekte im gleichen Cluster möglichst ähnlich sind, während sich Objekte unterschiedlicher Cluster so sehr wie möglich unterscheiden. Die Qualität eines Clusters ist demzufolge stark abhängig von dem verwendeten Ähnlichkeitsmaß und dessen Implementation [17]. Eine Schwierigkeit hierbei ist das Auffinden solcher geeigneter Maße, welche den Zusammenhalt innerhalb eines Clusters und die Trennung zu anderen Clustern bewerten.
An Methoden und Ansätzen für eine Clusteranalyse wird schon lange geforscht. Da diese Forschung schon weit vorangeschritten ist, gibt es schon viele gute Clustering-Algorithmen. Der Fokus lag jedoch meistens nicht auf Interpretierbarkeit, weshalb die Ergebnisse oft nur schwer für einen Benutzer nachvollziehbar oder kaum interpretierbar sind. Dem Anwender werden oft die Anzahl der Cluster und deren Mittelpunkte geliefert, jedoch bleibt die Frage offen, welche Kriterien die einzelnen Gruppen unterscheiden und wie die Werte der Instanzen sein müssen, um in ein entsprechendes Cluster eingeordnet zu werden. Ein anschauliches Beispiel hierzu befindet sich am Anfang von Kapitel 4.

1.1 Problemstellung

Das Hauptziel der Datenanalyse ist es, verwendbares Wissen über Daten zu erhalten. Sind die erhaltenen Ergebnisse einer Clusteranalyse nicht interpretierbar, so wird ein menschlicher Anwender kein neues Wissen aus diesen gewinnen können. Mit zunehmender Interpretierbarkeit nimmt das Verständnis über die Zusammensetzung der einzelnen Cluster zu. Es wird ersichtlich, weshalb eine bestimmte Menge von Instanzen ausgewählt wurde, ein Cluster zu bilden und wie sie sich von Objekten anderer Cluster unterscheiden. Die Interpretierbarkeit der Ergebnisse ist somit entscheidend für den gesamten Prozess der Datenanalyse, da die erlangten Informationen für einen menschlichen Anwender sonst kaum verwendbar wären.
Herkömmliche Clustering-Algorithmen, wie der bekannte K-Means Algorithmus [15], sind in der Lage, aussagekräftige Cluster in den gegebenen Datenmengen zu erkennen. Ein grundlegendes Problem dieser Algorithmen ist jedoch die Interpretierbarkeit [21] der gefundenen Cluster, da diese meist nur durch die jeweiligen Mittelwerte erfasst werden.
Das ursprüngliche Ziel der Clusteranalyse ist das Erkennen und Wiedergeben der natürlichen Strukturen in den Daten. Um zusätzlich zu gewährleisten, dass ein menschlicher Anwender das Clustering verstehen kann, wird das ursprüngliche Ziel mit der Forderung nach Interpretierbarkeit erweitert. Da deren

Güte jedoch schwer zu beurteilen ist [21] und von jedem Menschen anders beurteilt werden kann, sind verschiedenste Ansätze für die Clusteranalyse und Veranschaulichungen der Cluster denkbar. Vertreter, wie der Clustering Rule-based Algorithm (CRA) [33] oder der Fuzzy Rule-Based Clustering Algorithm (FRBC) [23] nutzen unterschiedliche Hilfsmittel und Lösungsansätze, um das erweiterte Ziel der Clusteranalyse zu erfüllen.
Die Fähigkeit, das durch Clustering erlangte Wissen in der Datenanalyse für einen Anwender verständlich und interpretierbar darzustellen, ist jedoch noch nicht ausgereift. Es existieren Ansätze, die noch Raum für Verbesserungen bieten. Außerdem sind noch andere Herangehensweisen denkbar.

1.2 Ziel

Das Ziel dieser Arbeit ist es, die existierenden Clustering-Verfahren, welche ihren Fokus auf das Erzeugen interpretierbarer Ergebnisse legen, zu untersuchen und einen alternativen regelbasierten Clustering-Algorithmus zu entwerfen. Dieser soll sich durch die Vorgehensweise zur Regelfindung von den existierenden Algorithmen unterscheiden und es menschlichen Anwendern ermöglichen, die gefundenen natürlichen Strukturen der Daten leichter zu interpretieren.

1.3 Lösungsidee

Der hier entwickelte Separate-and-Conquer based Clustering Algorithm (SBCA) erzeugt eine Regelmenge, welche die möglichen Cluster beschreibt. Wie am Anfang von Kapitel 4 in einem Beispiel gezeigt wird, sind diese Regeln einfach zu interpretieren und beschreiben jeweils ein Cluster.
Während der Ausführung des Algorithmus soll ein Cluster, das alle Instanzen der Eingabedaten abdeckt, soweit verfeinert werden, bis ein Stopp-Kriterium greift. Anschließend werden die Instanzen im gefundenen Cluster aus der Datenmenge entfernt und das nächste Cluster wird gefunden. So erklärt jede Regel genau ein Cluster.

2 Grundlagen

Dieses Kapitel bietet eine kurze Einführung in die Grundbegriffe überwachten und unüberwachten Lernens und befasst sich außerdem mit unterschiedlichen Darstellungsformen der Daten und der Cluster. Zusätzlich werden die Begriffe Kompaktheit und Separierung in der Clusteranalyse vorgestellt und erläutert, wozu diese benötigt werden.

2.1 Überwachtes und unüberwachtes Lernen

Überwachtes und unüberwachtes Lernen sind die zwei häufigsten Arten von Problemen des maschinellen Lernens. Während beim überwachten Lernen die Trainingsdaten zusätzliche Informationen für jedes Eingabeobjekt beinhalten, stehen beim unüberwachten Lernen keine weiteren Informationen zur Verfügung.
Im Falle des überwachten Lernens liefert ein Supervisor oder Lehrer zusätzliche Informationen (Zielwerte) zu den Daten. Jedes Trainingsbeispiel der Trainingsmenge besteht dementsprechend aus einem Eingabeobjekt und dem dazugehörigen Zielwert. Das Ziel hierbei ist das Finden einer Funktion, die unbekannten Objekten einen passenden Zielwert zuweist. Dieser Zielwert ist gewissermaßen die *richtige Antwort* der vorliegenden Frage- oder Problemstellung. Die gegebenen Zielwerte der Eingabedaten sind bereits die *richtige Antwort* der Eingabedaten und folgen dabei einem gewissen Konzept. Dieses Konzept soll gelernt werden und die von der zu findenden Funktion erzeugten Zielwerte sollen diesem Konzept folgen. Klassische Vertreter für Probleme des überwachten Lernens sind die Klassifikation und die Regression. Im Falle der Klassifikation wäre die *richtige Antwort*, die Antwort auf die Frage nach der Klassenzugehörigkeit.
Ein typisches Klassifikationsbeispiel ist das Erkennen von Gesichtern. Hierbei stehen dem Gesichtserkennungsalgorithmus im Laufe der Lernphase Bilder mit und ohne Gesichtern zur Verfügung. Dem Algorithmus wird während der Lernphase zusätzlich mitgeteilt, ob auf ein gegebenes Bild ein Gesicht zu sehen ist oder nicht. Die Aufgabe ist es anschließend bei unbekannten Bildern zu entscheiden, ob ein Gesicht zu sehen ist.
Im Gegensatz zum überwachten Lernen existieren beim unüberwachten Lernen keine Informationen über Zielwerte oder Klassen. Die Clusteranalyse und die Komprimierung von Daten sind Beispiele für Aufgaben aus dem Bereich der unüberwachten Probleme. Bei dieser Art von Problem besteht die Trainingsmenge lediglich aus den Eingabeobjekten. Das Ziel ist es, mögliche Strukturen oder Muster in den Daten zu erkennen.
Eine Beispielanwendung der Clusteranalyse ist die Bildsegmentierung, bei der anhand der Eigenschaften der Bildpunkte erkannt wird, ob diese zu einem Objekt zusammengehören. Die Eingabedaten sind hierbei die Bildpunkte.

2.2 Darstellungsform der Daten

Sowohl bei der Clusteranalyse als auch bei anderen Anwendungen aus dem Gebiet des unüberwachten Lernens besteht die Eingabe aus einer Menge von Instanzen ohne vorhandenes Wissen über Zielwerte oder Klassenzugehörigkeiten. Zur Beschreibung einer solchen Instanz x_j verwendet man ein Tupel aus Attributwerten $v_{i,j}$ des zugehörigen Attributes $\mathbf{A}_i$, $i \in 1, ...A$. Dieses Tupel hat die Form $x_j = (v_{1,j}, ..., v_{n,j})$. Im Falle eines diskreten Attributes A_i bestehen die möglichen Werte, die $v_{i,j}$ annehmen kann, aus einer endlichen Wertemenge. Handelt es sich um ein kontinuierliches Attribut, kann $v_{i,j}$ jede reelle Zahl annehmen.
Der gesamte Datensatz, der einem Clustering-Algorithmus zum Lernen zur Verfügung steht, setzt sich somit aus einer Menge von Tupeln aus Attributwerten zusammen.

2.3 Darstellungsform der Cluster

Die Darstellungsform der Cluster und wie die konkrete Vorhersage neuer Objekte bestimmt wird, hängen vom jeweiligen Algorithmus und dessen Implementation ab.

Das Angeben der Mittelpunkte der einzelnen Cluster ist die gebräuchlichste Variante. Sie wird beispielsweise von dem bekannten K-Means Algorithmus benutzt. Der Nachteil hierbei besteht in der Interpretierbarkeit, da es selbst bei einfachen Datensätzen nicht sofort ersichtlich ist, welchem Cluster ein neues Objekt angehört. Oft wird daher auch die Konjunktion logischer Ausdrücke, wie etwa Regeln, zur Beschreibung von Clustern verwendet. Durch diese ist es möglich, die gefundenen Cluster leichter zu interpretieren, da sich die Grenzen der einzelnen Cluster durch Anwendung der Regeln herleiten lassen und man eine genauere Vorstellung von den Objekten eines Clusters bekommt. Regeln lassen sich zudem meist in die menschliche Sprache übersetzen. So kann man beispielsweise passend zu der Bedingung "*farbe(objekt) = grün*" die Frage "*Ist das Objekt Grün?*" stellen. Wird diese Frage für ein konkretes Objekt mit "ja"beantwortet, ist Bedingung für dieses Objekt erfüllt.
Da die Interpretierbarkeit der Ergebnisse in dieser Arbeit eine wichtige Rolle einnimmt, wird auf eine regelbasierte Darstellungsform der Cluster zurückgegriffen.

2.4 Kompaktheit und Separierung in der Clusteranalyse

Um es einem Clustering-Algorithmus zu ermöglichen, Strukturen und Muster in den Daten aufzudecken, muss definiert sein, wann diese vorliegen. Eine Möglichkeit besteht darin, sich ähnelnde Objekte zusammenzufassen, da ihre ähnlichen Eigenschaften eine gewisse Struktur darstellen.
Die Abstände der Objekte zueinander, die sogenannten Intra-Cluster-Distanzen, müssen vorzugsweise gering sein, damit alle Objekte, die dem gleichen Cluster angehören, eng beieinander liegen, und somit möglichst ähnlich sind. Mithilfe dieser Intra-Cluster-Distanzen lassen sich verschiedene Kompaktheitsmaße berechnen. Eine hohe Kompaktheit drückt demzufolge einen großen Zusammenhalt innerhalb eines Clusters aus und muss für jedes Cluster einzeln berechnet werden. Diese Kompaktheitsmaße wurden von Bezdek und Pal [5] vorgeschlagen und können auch zur Evaluation einzelner Bedingungen der Cluster-Regeln angepasst werden.
Im Gegensatz zur Kompaktheit, lässt sich mithilfe der Separierung beschreiben, wie sehr sich die Objekte unterschiedlicher Cluster voneinander unterscheiden. Damit dieser Unterschied möglichst groß ist, müssen die Distanzen zwischen den verschiedenen Clustern, die sogenannten Inter-Cluster-Distanzen, groß sein. Für Separierungsmaße [5] werden diese Distanzen und somit der Abstand zwischen zwei Clustern gemessen und zur Berechnung der Separierung verwendet.

3 Separate-and-Conquer basiertes Regellernen

Gegeben:

- eine Datenbeschreibungssprache, definiert die Form der Daten
- eine Hypothesenbeschreibungssprache, definiert die Form der Regeln
- eine Abdeckungsfunktion *Cover(r,i)*, definiert ob eine Regel *r* die Instanz *i* abdeckt
- eine Menge von Trainingsinstanzen *E* beschrieben in der Datenbeschreibungssprache

Finde:

- eine Hypothese in Form einer Regelmenge *R*, formuliert in der Hypothesenbeschreibungssprache, die jede Instanz abdeckt

Abbildung 1: Definition der Aufgabe des Regellernens bei der Clusteranalyse

Das folgende Kapitel, welches die Grundlagen des Regellernens bei der Clusteranalyse vermittelt, basiert auf der Arbeit von Fürnkranz, Gamberger und Lavrač [11], in dem die Grundlagen des Lernens von Klassifikationsregeln beschrieben werden. Das anschließend gelieferte Wissen über die sogenannten Separate-and-Conquer Algorithmen beruht auf der Arbeit von Fürnkranz [12], in dem alle wesentlichen Bestandteile der Separate-and-Conquer Strategie ausführlich zu finden sind.

3.1 Problemdefinition

Die Problemstellung beim Regellernen für Klassifizierungsaufgaben lässt sich informal wie folgt definieren:

> *Gegeben sei eine Menge von Trainingsbeispielen. Finde eine Menge von Klassifikationsregeln, die zur Vorhersage und Klassifikation neuer Instanzen verwendet werden kann.*

Im Bereich des überwachten Lernens spricht man von Trainingsbeispielen, wenn man Informationen über die zugehörige Klasse einer Instanz besitzt. Da wir uns im Bereich des unüberwachten Lernens befinden und kein Wissen über eventuell vorhandene Klassen verfügen, muss die Definition für das Lernen von Regeln bei einer Clusteranalyse wie folgt abgeändert werden:

> *Gegeben sei eine Menge von Trainingsinstanzen. Finde eine Menge von Clusterregeln, die zur Gruppierung von Instanzen verwendet werden kann.*

Abbildung 1 bietet eine formalere Problemdefinition, welche zusätzlich noch Formalismen für die Beschreibung der Daten (Datenbeschreibungssprache) und der induzierten Regelmenge (Hypothesenbeschreibungssprache) beinhaltet. Mit *Hypothese* wird die Ausgabe des Lerners wegen der hypothetischen Eigenschaft des Findens einer Regelmenge bezeichnet. Durch diese kann nie garantiert werden, dass die Ausgabe beim induktiven Lernen nicht durch neue Kenntnisse, die dem Lerner präsentiert werden, verfälscht wird. Wie auch in [11] können hier die Synonyme Modell oder Theorie verwendet werden. Zuletzt wird noch eine Abdeckungsfunktion benötigt, um zu überprüfen, ob eine Regel eine Instanz abdeckt. Eine Instanz wird von einer Regel abgedeckt, wenn sie die Bedingungen dieser erfüllt. Äquivalent hierzu spricht man vom "*feuern*" einer Regel, wenn eine zu überprüfende Instanz von ihr abgedeckt wird.

3.2 Darstellungsform einer Regel

Ein Regellerner erzeugt aus den Attributwerten, welche die einzelnen Instanzen der Trainingsmenge beschreiben, eine oder mehrere Regeln der Form [11]:

$$IF\ f_1\ AND\ f_2\ AND\ \ldots\ AND\ f_L\ THEN\ Class = c_i.$$

Die Bedingungen einer solchen Regel werden zusammengefasst mit "*Body*" bezeichnet. Die Schlussfolgerung nennt man "*Head*" einer Regel. Die Anzahl L der Bedingungen ist gleichbedeutend mit der Regellänge. Bei Aufgaben aus dem Gebiet des überwachten Lernens hat man üblicherweise eine Menge von Instanzen inklusive dazugehöriger Klasse gegeben. Da diese Klasseninformationen bei der Clusteranalyse fehlen, weist hierbei der Head einer Regel auf das Cluster c_i, welchem die abgedeckten Punkte angehören:

$$IF\ f_1\ AND\ f_2\ AND\ \ldots\ AND\ f_L\ THEN\ Cluster = c_i.$$

Der Body einer Regel besteht sowohl beim überwachten als auch beim unüberwachten Lernen aus logischen Verknüpfungen der Bedingungen. Eine Bedingung f_k verifiziert, ob das zu überprüfende Objekt die in f_k beschriebene Eigenschaft besitzt.
Bedingungen haben normalerweise die Form $A_i < v$, $A_i \geq v$ oder $A_i = v_{i,j}$. Die ersten beiden finden Verwendung bei Attributen mit kontinuierlichen Werten. Letzteres im Falle diskreter Attribute. Während $v_{i,j}$ ein konkreter Attributwert ist, der bei einem gegebenen Objekt vorkommen muss, ist v lediglich eine Grenze, die frei gesetzt werden kann.
Sowohl hier als auch in [11] und in der logischen Terminologie ist der Body einer Regel nur aus Konjugationen von Literalen zusammengesetzt. Der Head besteht aus nur einem Literal.

3.3 Regelmenge

In den meisten Fällen reicht eine einzelne Regel nicht aus, um bei der Klassifikation oder der Clusteranalyse nützliche Ergebnisse zu erzielen, da nur zwischen zwei unterschiedlichen Zuordnungen unterschieden werden kann. Verwendet man hingegen eine Regelmenge, kann man beliebig viele Zielwerte unterscheiden und den Instanzen zuordnen. Um mit einer Regelmenge eine Vorhersage für ein unbekanntes Objekt zu erlangen, muss im Vorfeld festgelegt werden in welcher Form diese vorliegt.
Es gibt verschiedene Möglichkeiten, eine gelernte Regelmenge aufzubauen und zu interpretieren. Die Verwendung einer Entscheidungsliste oder einer ungeordnete Liste sind die am häufigsten verwendeten Formen.
Bei einer Entscheidungsliste, besitzt die gelernte Theorie eine feste Ausführungsreihenfolge. An dieser Stelle müssen die Regeln der Regelmenge in einer geordneten Reihenfolge vorliegen. Zur Vorhersage des Clusters werden die einzelnen Regeln in der Reihenfolge, in der sie in der Regelmenge auftauchen, ausgeführt. Hierbei bestimmt die erste feuernde Regel, welchem Cluster eine Instanz angehört. Alle darauffolgenden Regeln werden nicht mehr betrachtet.
Wird eine ungeordnete Liste verwendet, so hat die Position der Regel in der Menge keinen Einfluss auf das Ergebnis. In diesem Fall könnte eine Zuordnung der Instanz durch Mehrheitsentscheid erfolgen. Hierbei werden die Vorhersagen aller Regeln, die feuern, betrachtet und die Anzahl der Stimmen für die einzelnen Cluster c_i gezählt. Das Cluster, dem das zu untersuchende Objekt zugeordnet wird, ist das mit den meisten Stimmen.

3.4 Separate-and-Conquer Strategie

Separate-and-Conquer Algorithmen wurden für eine Vielzahl verschiedener Lernaufgaben entwickelt. Neben der herkömmlichen Variante, welche Regelmengen für attributwertbasierte Konzeptlernprobleme benutzen, existieren Modifikationen, die Entscheidungslisten oder Regressionsregeln erzeugen.

Algorithmus 1 Separate-and-Conquer Regellernalgorithmus [12]

```
1:  procedure SeCoRuleLearner(Examples)
2:      Theory = ∅
3:      while Positive(Examples) ≠ ∅ do
4:          BestRule = {true}
5:          Rule = BestRule
6:          Covered = cover(Rule, Examples)
7:          while Negative(Covered) ≠ ∅ do
8:              for Condition ∈ Conditions do
9:                  Refinement = Rule ∪ Condition
10:                 if Purity(Refinement, Examples) > Purity(BestRule, Examples) then
11:                     BestRule = Refinement
12:             Rule = BestRule
13:         Theory = Theory ∪ Rule
14:         Examples = Examples \ Covered
15:     return Theory
```

Die Separate-and-Conquer Strategie, welche auch Covering Strategie genannt wird und ihren Ursprung im AQ-Algorithmus [25] hat, findet Anwendung bei vielen regelbasierten Klassifikationsalgorithmen. Die gelernten Regeln ordnen üblicherweise jedem Beispiel die zugehörige Klasse zu. Da die Separate-and-Conquer Strategie sowohl in der Klassifikation als auch bei der Regression [18] Verwendung findet und gute Ergebnisse erzielt, wird sie in dieser Arbeit für den hier entwickelten Clustering-Algorithmus verwendet. Hierbei ermöglicht uns eine Regel die Zuordnung zwischen einem Objekt und einem Cluster.
Die Grundidee dieser Strategie, die auch ihren Namen prägte [28], ist das Entfernen der Daten, die durch eine Regel abgedeckt werden (Separate), und das anschließende Erobern der restlichen Daten (Conquer). Mit Erobern wird hier das Lernen einer neuen Regel bezeichnet.
Ein Separate-and-Conquer basierter Regellern-Algorithmus sucht dabei zuerst eine Regel, welche der Regelmenge hinzugefügt wird. Die von der Regel abgedeckten Daten werden entfernt, um im Anschluss auf den verbliebenen Daten eine weitere Regel zu lernen. Diese Schritte werden wiederholt bis die Regelmenge alle Beispiele abdeckt.
Bei dem von Fürnkranz vorgestellten induktiven Konzeptlernproblem wird davon ausgegangen, dass positive und negative Beispiele eines Zielkonzeptes gegeben sind, welche durch eine feste Anzahl an Attributen und evtl. zusätzlichem Hintergrundwissen beschrieben werden. Das Ziel ist hierbei, das Zielkonzept in Form expliziter Regeln zu beschreiben. Diese Regeln sollen die Instanzen des Zielkonzeptes korrekt erkennen.
Da bei der Clusteranalyse keine positiven und negativen Beispiele existieren, sondern nur Instanzen ohne jegliches Wissen über Gruppenzugehörigkeit, ist das Ziel in diesem Fall, die Daten in Cluster zu unterteilen. Diese werden in Form expliziter Regeln dargestellt, welche ähnliche Instanzen der gleichen Gruppe und unähnliche Instanzen einer unterschiedlichen Gruppe zuweisen.

3.5 Der Separate-and-Conquer Algorithmus

Die klassische Form des Separate-and-Conquer Algorithmus (Algorithmus 1) dient der Klassifikation, und wird in Kapitel 4 für die Clusteranalyse abgeändert. Der Algorithmus startet mit einer leeren Theorie, von der kein Beispiel der Trainingsmenge abgedeckt wird, und fügt dieser nacheinander Regeln hinzu. Für das Lernen einer neuen Regel wird eine allgemeingültige Regel mit dem Body *wahr* durch das Hinzufügen von Bedingungen solange verfeinert, bis nur noch positive Beispiele von ihr abgedeckt werden. Diese gefundene Regel wird anschließend der Theorie hinzugefügt und die von ihr abgedeckten Beispiele werden für die folgenden Schritte aus der Datenmenge entfernt. Dieses Suchen und Hinzufügen neuer Regeln wird wiederholt bis keine positiven Beispiele im Datensatz übrig sind.

In [12] beschreibt Fürnkranz zusätzlich einen generischen Separate-and-Conquer Regellernalgorithmus, welcher genutzt werden kann, um Varianten spezifischer Algorithmen aus der Literatur zu erstellen. Dieser Regellernalgorithmus [19] ist außerdem die Grundlage für die generische Form des in dieser Arbeit entworfenen Clustering-Algorithmus (Algorithmus 2).

3.6 Suchalgorithmus und Suchrichtung

Bei der Separate-and-Conquer Strategie gibt es verschiedene Möglichkeiten, den Hypothesenraum zu durchsuchen. Unterschiedliche Verfahren können dabei aus einer Reihe von Algorithmen, die von einem Hill-climbing Algorithmus bis hin zu einer vollständigen Durchsuchung des gesamten Hypothesenraums reichen, zurückgreifen. Der Nachteil einiger Suchalgorithmen ist ihr Rechenaufwand. Verfahren wie beispielsweise *Beam Search* oder *Exhaustive Search* sind sehr ineffizient. Ein Hill-climbing hingegen verfolgt eine einfache und effiziente Vorgehensweise.
Hill-climbing ist der in Separate-and-Conquer Systemen am häufigsten verwendete Suchalgorithmus. Hierbei wird versucht, die optimale Bewertung für eine Regel zu erlangen. Hierfür wird eine Regel solange verfeinert, bis keine Verbesserung mehr möglich ist. Durch Ausführen dieser lokal optimalen Verfeinerungen wird versucht, das globale Optimum zu finden. Der Vorteil von Hill-climbing ist die Simplizität. Ein Hill-climbing Algorithmus ist einfach aufgebaut und sehr effizient, da ein schrittweises Vorgehen verwendet wird. Ein erheblicher Nachteil von Hill-climbing ist jedoch die Kurzsichtigkeit, da durch die schrittweise Verfeinerung ein lokales Optimum gefunden werden kann, von dem aus das globale Optimum nicht erreichbar ist.
Neben dem Suchalgorithmus ist auch die Suchrichtung entscheidend. Ein Algorithmus kann dabei entweder eine Top-down oder Bottom-up Strategie verwenden. Bei einer Top-Down Strategie wird das Verfeinern einer Regel durch Spezialisierung erreicht. Die initiale Regel ist bei dieser Strategie eine allgemeingültige Regel mit dem Body *wahr*, welche alle Instanzen abdeckt. Anschließend werden der Regel sukzessiv Bedingungen hinzugefügt, wodurch weniger Instanzen von ihr abgedeckt werden.
Die Bottom-up Strategie hingegen durchsucht den Hypothesenraum in entgegengesetzter Richtung durch Generalisierung der zu verfeinernden Regel. Hierbei wird mit einer spezifischsten Regel begonnen, welche schrittweise generalisiert wird, um zunehmend mehr Instanzen abzudecken. Ein Nachteil der Bottom-up Strategie ist die Sensitivität gegenüber der Initialisierung. Für die Wahl einer spezifischsten Regel muss ein zufälliges Beispiel ausgewählt werden. Die gefundenen Regeln bei einer Bottom-up Strategie sind sehr stark von diesen initialen Beispielen abhängig.
Die verwendete Suchrichtung hat eine große Auswirkung auf die gefundenen Regeln und somit auf die resultierenden Modelle. Obwohl viele Clustering-Algorithmen auf der Bottom-up Strategie basieren, verwendet der SBCA eine Top-Down Strategie. Diese benötigt keine zufällige Initialisierung und wird auch in der Klassifikation häufig verwendet.

4 Separate-and-Conquer based Clustering Algorithm

Der hier vorgestellte Algorithmus erzeugt eine Regelmenge zur Einteilung der Daten in Cluster, damit eine entsprechende Interpretierbarkeit erreicht wird. Die gelernten Regeln vereinfachen das Erkennen der Clustergrenzen und ermöglichen es einem Anwender, neue Objekte ohne viel Aufwand dem richtigen Cluster zuzuordnen.
Zur Veranschaulichung sind in Tabelle 1, in Tabelle 2 und in Abbildung 2 die Ergebnisse zweier Clusteranalysen auf einen einfachen synthetischen Datensatz vorgenommen. Dieser lässt sich in vier offensichtliche Cluster unterteilen und soll verdeutlichen, dass die Interpretierbarkeit bereits auf einem einfachen Datensatz Schwierigkeiten bereiten kann. In Tabelle 1 ist eine Darstellung der Cluster durch ihre Mittelpunkte zu sehen. Im Vergleich hierzu befindet sich in Tabelle 2 eine Regelmenge, welche die einzelnen Cluster beschreibt. Die Mittelpunkte sind durch rote Kreise und die Regeln durch grüne Linien in Abbildung 2 visualisiert.

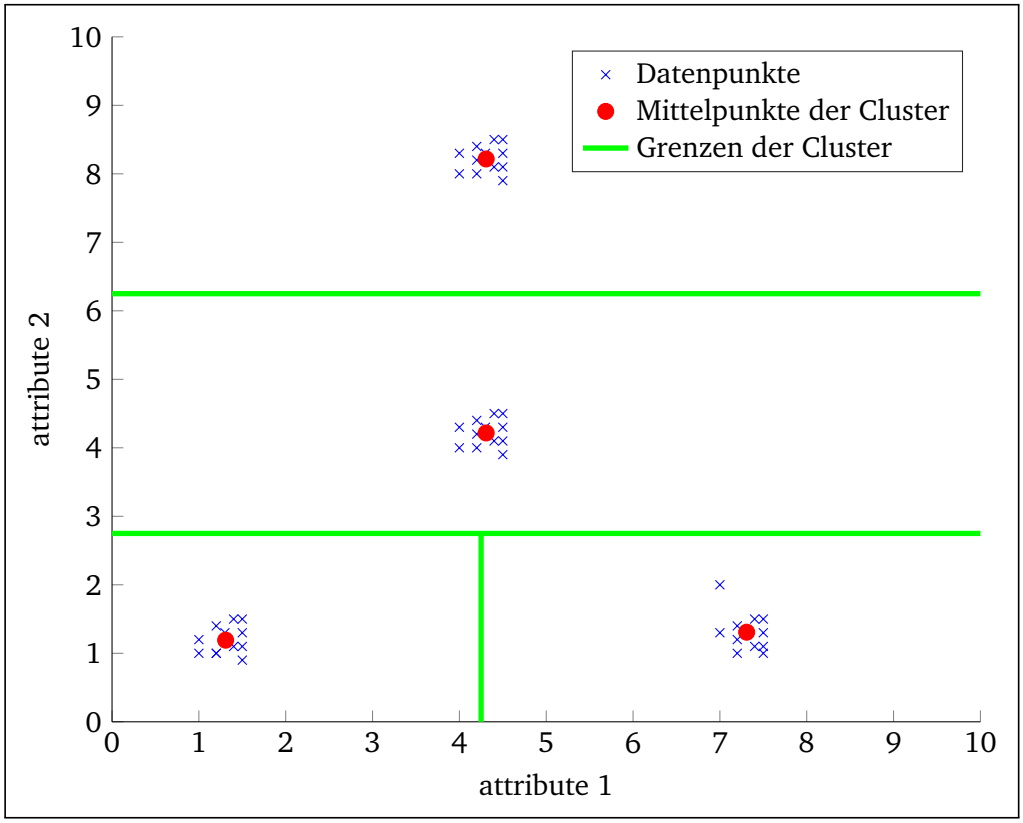


Abbildung 2: Visualisierung der Clusteranalyse eines synthetischen Datensatzes. Die Mittelpunkte und die Grenzen der einzelnen Cluster wurden durch zwei unterschiedliche Clustering-Algorithmen erzeugt. Während bei einem herkömmlichen Verfahren nur die Mittelpunkte angegeben werden können, kann man durch einen regelbasierten Algorithmus die Grenzen der Cluster klar definieren.

Cluster 1	Cluster 2	Cluster 3	Cluster 4
attribute 1: 4.3083 attribute 2: 4.2167	attribute 1: 1.3083 attribure 2: 1.1917	attribute 1: 4.3083 attribure 2: 8.2167	attribute 1: 7.3083 attribure 2: 1.3083

Tabelle 1: Darstellung der Cluster durch Mittelpunkte

attribute 2 $\geq$ 6.25	$\Rightarrow$ Cluster 1
attribute 2 $\geq$ 2.75	$\Rightarrow$ Cluster 2
attribute 1 <4.25	$\Rightarrow$ Cluster 3
else	$\Rightarrow$ Cluster 4

Tabelle 2: Darstellung der Cluster durch eine Entscheidungsliste

Während es einem menschlichen Anwender schwer fällt, ein Objekt durch Mittelpunkte dem richtigen Cluster zuzuweisen, kann bei einer geordneten Entscheidungsliste diese der Reihe nach durchlaufen werden und bei einer passenden Regel das Zielcluster sofort abgelesen werden. Durch einfaches Betrachten der Bedingungen können zudem die ausschlaggebenden Attribute herausgefunden werden.

In diesem Kapitel werden mehrere Maße für Heuristiken oder Abbruchbedingungen verwendet, die auf der Distanz zwischen Instanzen beruhen. Obwohl auch andere Metriken in Frage kommen, wurde für alle Berechnungen der Distanz der euklidischen Abstand verwendet, da dieser einer der am häufigsten verwendeten Distanzmaße ist. Die Attributwerte wurden dazu im Vorfeld normalisiert. So haben zwei Werte des gleichen Attributes einen maximalen Abstand von 1.
Durch Verwendung anderer Metriken wäre es möglich, dass gleich gute oder sogar bessere Ergebnisse erzielt werden, jedoch würden sich die generellen Eigenschaften (z.B. Parameter, Design des Algorithmus) dadurch kaum ändern. Für die Wahl einer optimalen Metrik müssen in zukünftigen Arbeiten weitere Experimente durchgeführt werden

4.1 Konzept des Algorithmus

Der Separate-and-Conquer based Clustering Algorithm wurde entworfen, um ein Clustering zu erzeugen, welches für einen menschlichen Anwender einfach zu interpretieren ist. Um dies zu erreichen, wird eine Regelmenge gelernt. Hierfür wird eine Separate-and-Conquer Strategie verwendet, welche sich durch gute Ergebnisse bereits in der Klassifikation bewiesen hat. Das in [19] vorgestellt Framework bietet zusätzlich die Grundlagen für den Entwurf eines auf Separate-and-Conquer basierenden Algorithmus.
Der SBCA ist eine Abwandlung des Separate-and-Conquer Algorithmus und untersucht den Hypothesenraum in Top-Down Richtung mittels eines Hill-Climbing Suchalgorithmus. Die gefundenen Cluster werden in Form von Regeln in einer geordneten Entscheidungsliste dargestellt. Jedes Cluster wird hierbei von ausschließlich einer Regel beschrieben.
Es existiert zusätzlich eine generische Form des Algorithmus (Algorithmus 2), welche es ermöglicht, die Suchrichtung und den Suchalgorithmus sowie die Abbruchkriterien frei zu verändern. Um aussagekräftige Cluster in angemessener Laufzeit zu finden, ist es nötig, diese konfigurierbaren Stellen zu optimieren und geeignete Heuristiken zu finden, welche die Qualität einzelner Regeln sinnvoll beurteilen.

4.2 Der Algorithmus

Die generische Form des Separate-and-Conquer based Clustering Algorithm (Algorithmus 2) ist eine abgewandelte Form des generischen Separate-and-Conquer Regellern Algorithmus in [12]. In beiden werden mehrere verschiedene Unterprogramme aufgerufen. Diese können verwendet werden, um unterschiedliche Formen des Algorithmus zu erzeugen.
Algorithmus 2 startet mit einer leeren Theorie, von der keine der Eingabeinstanzen abgedeckt wird. Unter der Annahme, dass die Menge der Eingabedaten nicht leer ist, beginnt der Algorithmus mit der While-Schleife. In dieser wird die Subroutine FindClusterRule aufgerufen. Diese lernt eine einzelne Regel, welche einen Teil der Daten abdeckt und somit ein einzelnes Cluster beschreibt. Alle von dieser Regel abgedeckten Instanzen werden aus der Menge der unabgedeckten Instanzen entfernt. Ist diese Menge anschließend nicht Leer, so wird die gelernte Regel der Theorie hinzugefügt. Falls alle verbleibenden Instanzen durch eine Regel abgedeckt wurden, wird die zuletzt gefundene Regel nicht zur Theorie hin-

Algorithmus 2 Separate-and-Conquer based Clustering Algorithm

```
1:  procedure RULEBASEDCLUSTERINGALGORITHMUS(Instances)
2:      Theory = ∅
3:      UncoveredInstances = Instances
4:      while true do
5:          ClusterRule = FINDCLUSTERRULE(UncoveredInstances)
6:          Covered = COVER(ClusterRule, UncoveredInstances)
7:          UncoveredInstances = UncoveredInstances \ Covered
8:          if UncoveredInstances = ∅ then
9:              exit while
10:         Theory = Theory ∪ ClusterRule
11:         if CLUSTERSTOPPINGCRITERION(Instances, UncoveredInstances) then
12:             exit while
13:     Theory = SETDEFAULTRULE(Theory)
14:     return Theory
15:
16: procedure FINDCLUSTERRULE(Instances)
17:     BestRule = ∅
18:     while Instances ≠ ∅ do
19:         Conditions = INITIALIZECONDITION(Instances)
20:         Candidate = ∅
21:         BestValue = −∞
22:         for Condition ∈ Conditions do
23:             Value = EVALUATECONDITION(Condition, Instances)
24:             if Value > BestValue then
25:                 BestValue = Value
26:                 Candidate = Condition
27:         if STOPPINGCRITERION(Candidate, Instances) then
28:             exit while
29:         BestRule = BestRule ∪ Candidate
30:         Instances = COVER(Rule, Instances)
31:     return BestRule
```

zugefügt, da diese am Ende des Algorithmus durch die Default-Regel mit dem Body *wahr* ersetzt wird. Diese Default-Regel deckt alle verbleibenden Instanzen ab.
Die durchlaufene While-Schleife sorgt solange für das Lernen neuer Regeln und somit für das Finden weiterer Cluster, bis alle Instanzen abgedeckt sind oder ein optionales Anhaltekriterium (ClusterStoppingCriterion) erfüllt ist.

4.2.1 FindClusterRule

Die Methode *FindClusterRule* ist zuständig für das Finden eines Clusters auf den übergebenen Daten. In dieser können der gewählte Suchalgorithmus (zum Beispiel Hill-Climbing) und die Suchrichtung (zum Beispiel Top-Down oder Bottom-Up) implementiert werden. Die in Algorithmus 2 zu findende Variante dieser Methode durchsucht den Hypothesenraum in Top-Down Richtung und betrachtet nach Spezialisierung einer Regel nur noch die von dieser Regel abgedeckten Instanzen. Dies bedeutet, dass nur die Punkte des aktuellen Clusters weiter untersucht werden, und ist nicht zu verwechseln mit dem Entfernen der von einer Regel abgedeckten Instanzen aus der Datenmenge im eigentlichen Separate-and-Conquer based Clustering Algorithm.
In dieser Variante wird zuerst eine Regel ohne Bedingungen initialisiert, welche die gesamte Datenmen-

ge abdeckt. Solange die aktuelle Regel noch Instanzen abdeckt, wird eine While-Schleife durchlaufen, welche diese Regel schrittweise um einzelne Bedingungen erweitert.
In der Schleife werden zuerst die möglichen Bedingungen, um die man die aktuelle Regel erweitern könnte, durch *InizializeConditions* gefunden. Anschließend werden diese durch *EvaluateCondition* bewertet. Die Bedingung mit der höchsten Bewertung ist der Kandidat, welcher der aktuellen Regel hinzugefügt werden soll. Wenn das *StoppingCriterion* die While-Schleife nicht unterbricht, wird dieser Kandidat der Regel hinzugefügt. Zuletzt werden die von dieser neuen Regel nicht abgedeckten Instanzen für weitere Schleifendurchläufe entfernt.
Wenn das *StoppingCriterion* die While-Schleife beendet, bedeutet das, dass das aktuelle Cluster nicht weiter verändert wird. Die Prozedur *FindClusterRule* wird verlassen und liefert die gelernte Regel dem Separate-and-Conquer based Clustering Algorithm.

4.2.2 InitializeCondition

InizializeConditions wird in *FindClusterRule* aufgerufen und liefert mögliche Split-Punkte zurück, an denen die übergebene Datenmenge geteilt werden kann. Diese Split-Punkte bilden später zusätzlich die Grenzen der einzelnen Cluster und sind in Abbildung 2 als grüne Linien erkennbar. Die Split-Punkte werden in Form von Bedingungen erstellt. Abhängig davon, ob eine Instanz diese Bedingung erfüllt, liegt sie auf der einen oder anderen Seite der Grenze.
Um solche Split-Punkte zu finden, werden zuerst Attribute ausgewählt, an denen geteilt werden soll. Die einfachste Möglichkeit ist es, jedes vorkommende Attribut zu wählen. Anschließend werden für jedes dieser gewählten Attribute sinnvolle Grenzwerte bestimmt. Die genaue Bestimmung solcher Werte kann auf unterschiedliche Weise geschehen und hängt unter anderem davon ab, ob ein Attribut numerisch oder nominal ist.
Bei nominalen Attributen haben die möglichen Split-Punkte die Form $A_i = v_{i,j}$. Hierbei kann $v_{i,j}$ jeder Attributwert sein, den ein Attribut A_i annehmen kann und im Datensatz vorkommt. Es ist möglich, für jedes vorkommende nominale Attribut und jeden dazugehörigen Attributwert einen Split-Punkt zu erzeugen.
Im Falle eines numerischen Attributes ist die Wertemenge meist kontinuierlich. Da Split-Punkte der Form $A_i = v_{i,j}$ bei solchen Werten oft nicht sinnvoll sind, werden Grenzen der Form $A_i < v$ oder $A_i \geq v$ definiert. Die Schwellenwerte v können konkrete Werte der Attribute A_i annehmen oder eine frei gewählte Zahl sein.
So können beispielsweise alle vorkommenden Attributwerte aufsteigend sortiert werden: $(v_{i,1}, ..., v_{i,n})$ mit $v_{i,k} \leq v_{i,k+1}$. Die Mittelwerte zweier solcher aufeinanderfolgenden Werte können als Schwellenwert verwendet werden:

$$v = \frac{v_{i,k} + v_{i,k+1}}{2} \tag{1}$$

Anschließend werden die Split-Punkte $A_i < v$ und $A_i \geq v$ für jeden berechneten Mittelwert erzeugt. Um zusätzlich Grenzen zu erhalten, welche alle bzw. keine Datenpunkte abgrenzen, kann man außerdem die Split-Punkte $A_i < v_{i,min}$, $A_i \geq v_{i,min}$, $A_i < v_{i,max}$ und $A_i \geq v_{i,max}$ erzeugen. Hierbei ist $v_{i,min}$ der kleinste und $v_{i,max}$ der größte für das Attribut A_i vorkommende Wert.
An dieser Stelle ist es wichtig zu erwähnen, dass die Wahl des Schwellenwertes v großen Einfluss auf das entstehende Cluster hat. Benutzt man im Datensatz vorkommende Werte, so entstehen sehr spezielle Cluster, bei denen die Grenzen auf den Randpunkten des Clusters liegen. Das Verwenden von Mittelwerten führt zu generelleren Clustern, da die Grenzen in der Mitte zweier Cluster liegen würden. Dieses Verengen bzw. Ausweiten der Grenzen wiederum hat jedoch keine direkte Auswirkung auf die Kompaktheit und Separierung der Cluster, da hierdurch die gleichen Instanzen weiterhin dem gleichen Cluster zugeordnet werden.
In dieser Arbeit und in den später folgenden Experimenten wurden für jedes Attribut und für jeden dazugehörigen Attributwert ein möglicher Split-Punkt bestimmt. Im Falle numerischer Attribute wurde wie beschrieben, die Mittelwerte zweier aufeinanderfolgender Werte als möglicher Split-Punkt erstellt.

4.2.3 EvaluateCondition

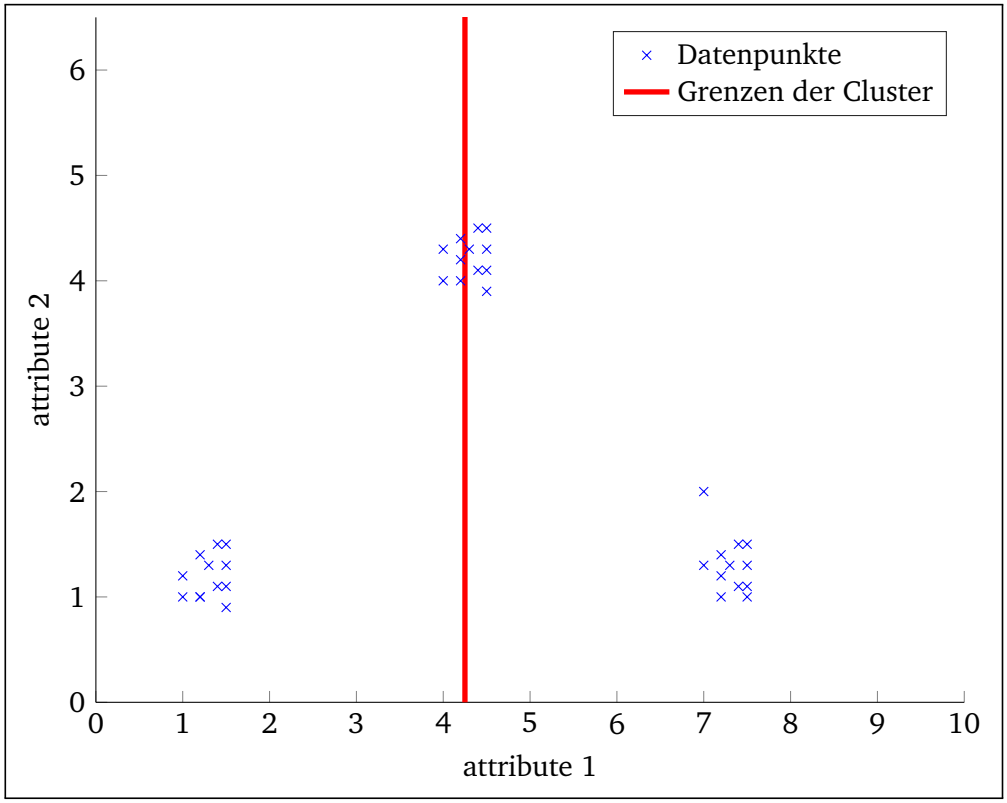


Abbildung 3: Eine schlechte Grenze kann durch eine ungünstige Heuristik ausgewählt werden und führt zu schlechten Ergebnissen bei der Clusteranalyse

In jeder Schleifeniteration der *FindClusterRule* Methode wird die aktuelle Regel um eine Bedingung erweitert. Da bei *InizializeConditions* üblicherweise mehrere Bedingungen erzeugt werden, müssen diese bewertet werden, um anschließend nur die beste Bedingung hinzuzufügen. Die Bewertung wird in *EvaluateCondition* auf Basis eines geeigneten Maßes durchgeführt. Jeder Bedingung wird ein Zahlenwert zugewiesen. Je größer dieser Wert ist, desto besser ist eine Bedingung im Bezug auf die verwendete Bewertungsformel, die sogenannte Heuristik.
Die Zielvorgabe bei Clustering-Algorithmen ist, eine große Inter-Cluster Gleichheit und gleichzeitig eine geringe Intra-Cluster Gleichheit zu erreichen. Dies entspricht einer großen Kompaktheit der einzelnen Cluster und einer großen Separierung unterschiedlicher Cluster. *EvaluateCondition* bewertet die einzelnen Bedingungen hinsichtlich dieser Zielvorgabe.
Zu Beachten ist jedoch, dass man davon ausgeht, dass alle Instanzen, die von der aktuellen Regel abgedeckt werden, dem gleichen Cluster angehören. Dies kann sich im späteren Verlauf des Algorithmus noch ändern. Alle restlichen Instanzen können ein oder mehrere Cluster bilden, die zu diesem Zeitpunkt noch unbekannt sind. Außerdem wurden die Instanzen bereits gefundener Cluster, sowie Instanzen, die von vorherigen Bedingungen der aktuellen Regel nicht abgedeckt werden, entfernt. Dies hat zur Folge, dass eine Berechnung der Kompaktheit anderer Cluster hier nicht möglich ist. Ebenso ist auch eine Berechnung der Separierung zweier Cluster an dieser Stelle nur bedingt möglich. Die Heuristik soll somit lediglich beurteilen, ob es sinnvoll ist, die aktuell betrachteten Daten nochmals zu unterteilen. Hierzu sollte das gewählte Maß den Ansprüchen der Clusteranalyse entsprechen.
Die verwendete Heuristik hat zudem sehr großen Einfluss auf das Ergebnis der Clusteranalyse, da von ihr abhängt, wie die Grenzen der einzelnen Cluster verlaufen. Die Verwendung einer ungeeigneten Heuristik, welche zum Beispiel ein vorhandenes Cluster, wie in Abbildung 3 zu sehen, fälschlicherweise teilt, kann schlechte Ergebnisse zu Folgen haben. Entsteht eine solche Grenze, ist es sehr wahrscheinlich, dass

das nachfolgende *StoppingCriterion* bereits die erste Bedingung der ersten Regel zurückweist und somit nur eine einzige Regel mit dem Body *wahr* gelernt wird. Dies würde einem einzigen großen Cluster entsprechen.
Die folgenden Ansätze für ein mögliches Maß bieten einen Anhaltspunkt für die Wahl einer geeigneten Heuristik zur Beurteilung der einzelnen Split-Punkte.
Eine einfache Möglichkeit ist es, die Entfernungen der Instanzen für eine Beurteilung zu verwenden. Hierzu berechnet man die durchschnittliche Distanz d_{in} innerhalb eines Clusters und die durchschnittliche Distanz d_{out} zu den Punkten außerhalb des Clusters. Anschließend kombiniert man beide Werte so, dass eine Heuristik entsteht, die kleine Werte für d_{in} und große Werte für d_{out} bevorzugt.
Zur Berechnung der durchschnittlichen Distanz d_{in} innerhalb eines Clusters berechnet man die Summe der Distanzen aller Instanzen des Cluster zueinander. Für einen Vergleich, der unabhängig von der Clustergröße ist, muss man anschließend durch die quadratische Anzahl der Instanzen in dem Cluster teilen.

$$d_{in} = \frac{1}{|C_1|^2} \sum_{x_i \in C_1} \sum_{x_j \in C_1} \left|\left| x_i - x_j \right|\right|_2 \tag{2}$$

Für die Berechnung der durchschnittlichen Distanz d_{out} zu den Punkten außerhalb des Clusters, berechnet man die Summe der Distanzen von jeder Instanz innerhalb des Clusters zu jeder Instanz außerhalb des Clusters. Anschließend teilt man durch die Anzahl der Instanzen innerhalb der Cluster multipliziert mit der Anzahl außerhalb des Clusters.

$$d_{out} = \frac{1}{|C_1||C_2|} \sum_{x_i \in C_1} \sum_{x_k \in C_2} ||x_i - x_k||_2 \tag{3}$$

Die einfachste Möglichkeit d_{in} und d_{out} zu kombinieren, bietet die Subtraktion. Große Werte für d_{out} erhöhen den Beurteilungswert (*score*), wohingegen große Werte für d_{in} die Beurteilung senken. Dies entspricht dem Anspruch einer großen Inter-Cluster Gleichheit und einer geringen Intra-Cluster Gleichheit.

$$score_a = d_{out} - d_{in} \tag{4}$$

Da dieser Ansatz nur auf sehr einfachen Datensätze gute Ergebnisse liefert, ist es nötig, ihn für eine reale Anwendungen anzupassen. Denkbar ist es, einen Parameter einzuführen, der für eine zusätzliche Gewichtung zwischen d_{out} und d_{in} sorgt.

Eine Möglichkeit ist es, die Abstände innerhalb eines Clusters mit der Distanz zum nächsten Punkt außerhalb des Clusters zu kombinieren. Dabei wird zuerst für jede Instanz des Clusters die Entfernung zu dessen nächsten Nachbarn innerhalb des Clusters bestimmt. Als nächstes wird die größte dieser Distanzen der nächsten Nachbarn bestimmt. Formal lässt sich dies mit folgender Formel beschreiben:

$$d_{in} = \max\left(\min\left(\left\{ ||x_1 - x_i||_2 \,\middle|\, x_1, x_i \in C_1 \wedge x_1 \neq x_i \right\}\right), ..., \min\left(\left\{ ||x_n - x_i||_2 \,\middle|\, x_n, x_i \in C_1 \wedge x_n \neq x_i \right\}\right)\right) \tag{5}$$

Hierbei ist C_1 das Cluster, welches durch die zu bewertende Bedingung entsteht.
Anschließend wird die minimale Distanz vom Cluster C_1 zum nächsten Punkt außerhalb dieses Clusters bestimmt:

$$d_{out} = \min\left(\left\{ ||x_i - x_k||_2 \,\middle|\, x_i \in C_1 \wedge x_k \in C_2 \right\}\right) \tag{6}$$

Hierbei beinhaltet C_2 alle Punkte, die nicht im Cluster C_1 liegen.
Bei dieser Heuristik sind das Verhalten und die Kombinationsmöglichkeiten von d_{in} und d_{out} sehr ähnlich zu der zuvor vorgestellten Heuristik basierend auf den durchschnittlichen Distanzen. Große Werte für

d_{out} und kleine Werte für d_{in} erhöhen den Beurteilungswert. Dieser lässt sich äquivalent wie in Gleichung (4) berechnen. Auch hier ist es aus Performanzgründen denkbar, einen Parameter einzuführen, der für eine Gewichtung zwischen d_{out} und d_{in} sorgt.

Eine weitere Heuristik basiert auf den Distanzen der einzelnen Instanzen in einem Cluster und auf der maximalen Distanz des Clusters. In Kapitel 5 wird diese Heurstik in den Experimenten verwendet und ihr Verhalten genauer untersucht. Grundidee dieser Heuristik ist das Bestrafen der Distanz der unähnlichsten Instanzen innerhalb eines Clusters. Zusätzlich werden die einzelnen Entfernung aller Clusterobjekte zueinander bestraft.
Die beiden Objekte mit der größten Distanz zueinander sind die unähnlichsten Instanzen eines Clusters. Die Distanz dieser Instanzen lässt sich durch

$$d_{max} = \max\left(\left\{\left|\left|x_i - x_j\right|\right|_2 \middle| x_i, x_j \in C_1\right\}\right) \tag{7}$$

berechnen. Aus diesem Wert d_{max} lässt sich zusammen mit den internen Distanzen ein Beurteilungswert wie folgt berechnen:

$$score_b = \frac{1}{|C_1|} \sum_{x_i \in C_1} \sum_{x_j \in C_1} g(x_i, x_j, d_{max}) \tag{8}$$

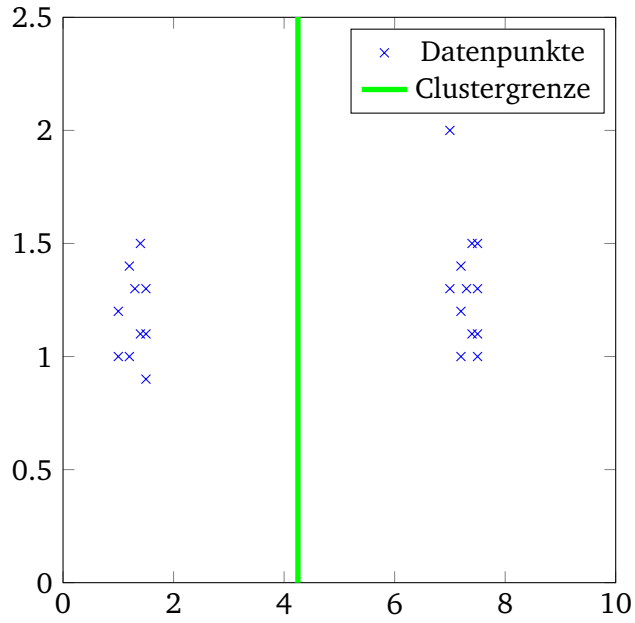
Hierfür wird eine Funktion g(a,b,max) benötigt. Diese ist definiert durch:

$$g(a, b, max) = 1 - ||a - b||_2 \cdot max \tag{9}$$

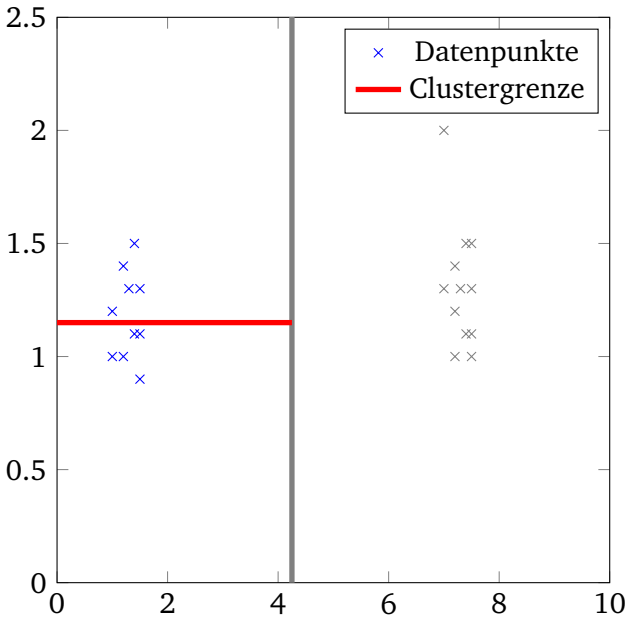
Die Eingabe von g(a,b,max) besteht aus zwei Instanzen a und b und einer maximalen Distanz max. Zuerst wird der Abstand zwischen den beiden Instanzen a und b berechnet und dieser anschließend mit max multipliziert. Die Rückgabewerte der Funktion g liegen in [-∞,1], da von Distanzen größer oder gleich Null ausgegangen wird.
Durch diese Funktion wird erreicht, dass eine niedrige maximale Clusterdistanz für hohe Beurteilungswerte sorgt. Gleichzeitig erhöhen geringe interne Distanzen den Beurteilungswert.
Im SBCA und in den späteren Experimenten wird letztere vorgestellte Heuristik verwendet. Die Bewertung wird wie in Gleichung (8) berechnet.

4.2.4 StoppingCriterion

Nachdem die beste Bedingung zur Verfeinerung der aktuellen Regel ausgesucht wurde, muss festgestellt werden, ob das Hinzufügen dieser Bedingung eine gewünschte Verbesserung der Kompaktheit und Separierung erzielt. Ist dies der Fall, wird sie der Regel hinzugefügt und die While-Schleife *FindClusterRule* wird fortgeführt. Andernfalls wird sie verworfen und die Regel nicht weiter verfeinert. Diese Überprüfung ist die Aufgabe des *StoppingCriterion*.
Bei einem sehr toleranten Kriterium ist die Wahrscheinlichkeit, dass eine Bedingung akzeptiert und somit zur Regel hinzugefügt wird, größer. Da hierdurch die Regeln spezifischer werden, sorgt ein lockeres Kriterium für das Entstehen vieler kleinerer Cluster. Ein strenges Kriterium hingegen lehnt mehr Bedingungen ab und bevorzugt somit allgemeinere Regeln mit weniger Bedingungen. Dies führt zu weniger Clustern, die jedoch mehr Instanzen beinhalten.
Für eine konkrete Form des *StoppingCriterions* gibt es viele denkbare Ansätze. Wie auch bei *EvaluateCondition*, ist es wichtig zu beachten, dass alle Instanzen, die von der aktuellen Regel abgedeckt werden, dem selben Cluster angehören und einige Instanzen bereits aus der Datenmenge entfernt wurden. Die verbleibenden Instanzen bilden eine ungewisse Anzahl an Clustern. Auch hier ist eine Berechnung der Kompaktheit und Separation nur begrenzt möglich.



(a) Eine gefundene Bedingung wird hinzugefügt, da sich eine deutliche Verbesserung der Kompaktheit zeigt. Die beiden vorhandenen Cluster werden genau in der Mitte geteilt.



(b) Das *StoppingCriterion* lehnt das Hinzufügen der Bedingung ab, da sich durch eine weitere Grenze keine deutliche Verbesserung der Kompaktheit ergibt.

Abbildung 4: Darstellung der beiden möglichen Situationen des *StoppingCriterions*

Ein sehr einfaches Maß, welches für das *StoppingCriterion* verwendet werden kann, besteht aus der Berechnung der durchschnittlichen Cluster Distanz. Dieses Maß wird zuerst für alle vorliegenden Instanzen berechnet. Dies entspricht der durchschnittlichen Distanz d_{prev} bevor die neue Bedingung hinzugefügt wird. Anschließend wird d_{new} für alle Instanzen, welche die neue Bedingung erfüllen, berechnet.

$$d_{prev} = \frac{1}{|C|^2} \sum_{x_i \in C} \sum_{x_j \in C} \left|\left| x_i - x_j \right|\right|_2 \qquad d_{new} = \frac{1}{|C_{new}|^2} \sum_{x_i \in C_{new}} \sum_{x_j \in C_{new}} \left|\left| x_i - x_j \right|\right|_2 \tag{10}$$

Die nun vorliegenden Werte d_{prev} und d_{new} können als Werte der Kompaktheit der Cluster interpretiert werden. Durch Vergleichen dieser beiden Werte ist es möglich zu entscheiden, ob die vorhandene Bedingung der aktuellen Regel hinzugefügt werden soll. Das *StoppingCriterion* überprüft dabei folgendes Kriterium:

$$d_{new} > d_{prev} \cdot \pi \tag{11}$$

Der Parameter π regelt hierbei, wie groß eine Verbesserung durch eine neue Bedingung sein muss, damit diese der Regel hinzugefügt wird. Die möglichen Werte dieses Parameters liegen im Bereich]0,1]. Werte außerhalb dieses Bereiches würden auch Bedingungen, die eine Verschlechterung bewirken, akzeptieren. Der Vorteil dieses Parameters ist die Regulierbarkeit der Ergebnisse. Durch setzen des Parameters $\pi = 1$ genügt eine beliebig geringe Verkleinerung der durchschnittlichen Distanz, damit eine Bedingung akzeptiert wird. Dies würde das Entstehen vieler kleiner Cluster zur Folge haben. Werte für π nahe 0 hingegen akzeptieren nur Bedingungen, die eine große Verbesserung verursachen. Da dies nicht sehr oft gegeben ist, würden wenige große Cluster mit vielen Instanzen entstehen.
Ein Nachteil dieses Parameters ist jedoch, dass der gegebene Wertebereich auch viele Werte enthält, die zu ungebräuchlichen Ergebnissen führen. Aus diesem Grund ist es sinnvoll, diesen Wertebereich weiter einzuschränken. Das resultierende Intervall sollte nur Werte enthalten, die auf möglichst vielen Datensätzen gute Ergebnisse liefern. Dennoch sollte es groß genug sein, um eine Regulierung der Größe und

Anzahl der Cluster zu ermöglichen.

Weitere mögliche Ansätze für ein *StoppingCriterion* könnten auf ähnliche Maße wie die *EvaluateCondition* beruhen. So ist es beispielsweise denkbar, auf die Berechnung der Distanz d_{out} zum nächsten Punkt außerhalb des neuen Clusters zurückzugreifen. Dies wurde für eine Heuristik bereits in Gleichung (6) formuliert. Vergleicht man diesen Wert d_{out} mit der größten vorkommenden Distanz d_{in} aller nächsten Nachbarn innerhalb des neuen Clusters, so erhält man ein weiteres *StoppingCriterion*. Auch d_{in} lässt sich wie in Gleichung (6) gezeigt berechnen.

4.2.5 ClusterStoppingCriterion

Das *ClusterStoppingCriterion* ist ein optionaler Bestandteil des eigentlichen Algorithmus. Es untersucht, ob in den verbleibenden Daten weitere aussagekräftige Cluster vorhanden sind. Eine Methode, die dies herausfindet, benutzt dabei die durchschnittlichen Distanzen $d_{uncovered}$ zwischen den verbleibenden, unabgedeckten Instanzen U und vergleicht diese mit den durchschnittlichen Distanzen d_{total} aller Eingabedaten I. Hierzu werden zuerst $d_{uncovered}$ und d_{total} berechnet:

$$d_{uncovered} = \frac{1}{|U|^2} \sum_{x_i \in U} \sum_{x_j \in U} \left|\left|x_i - x_j\right|\right|_2 \qquad d_{total} = \frac{1}{|I|^2} \sum_{x_i \in I} \sum_{x_j \in I} \left|\left|x_i - x_j\right|\right|_2 \tag{12}$$

Da die unabgedeckten Instanzen Teil der Eingabedaten sind, gilt hier $U \subset I$. Nach der Berechnung von $d_{uncovered}$ und d_{total}, müssen diese verglichen werden:

$$d_{uncovered} > d_{total} \cdot \mu \tag{13}$$

Hierbei regelt ein datensatzabhängiger Parameter μ die Strenge des Kriteriums.
Bestehen die verbleibenden Daten U hauptsächlich aus Ausreißern, die eine große Distanz zueinander aufweisen und keine Cluster bilden, so wird man bei $d_{uncovered}$ einen großen Wert feststellen. Überschreitet $d_{uncovered}$ den Wert von d_{total} multipliziert mit einem Parameter μ, so ist dies eine Abbruchsituation für den Separate-and-Conquer based Clustering Algorithm. Während kleine Werte für μ für einen frühzeitigen Abbruch sorgen, versucht der Algorithmus bei großen μ-Werten stetig weitere Cluster zu bilden. Liegt ein Datensatz mit Ausreißern vor, so ermöglicht es das *ClusterStoppingCriterion*, diese durch rechtzeitigen Abbruch der Default-Regel zuzuweisen. In Tabelle 3 ist eine alternative Entscheidungsliste dargestellt, bei der die verbleibenden Instanzen kein aussagekräftiges Cluster bilden. Instanzen, bei denen keine Regel feuert, werden somit keinem Cluster zugeordnet.
Das Weglassen dieses Kriteriums kann dazu führen, dass bei verbleibenden Ausreißern nur wenig Ähnlichkeit festgestellt werden kann und diese in viele kleine Cluster gegliedert werden. Dies hätte eine Verschlechterung der Interpretierbarkeit zu Folge.

Regel 1	⇒ Cluster 1
Regel 2	⇒ Cluster 2
...	⇒ ...
Regel K	⇒ Cluster K
else	⇒ kein Cluster

Tabelle 3: Alternative Entscheidungsliste, bei der die verbleibenden Instanzen kein Cluster bilden

5 Experimente

In den folgenden Experimenten wird der Separate-and-Conquer based Clustering Algorithm hinsichtlich des zu optimierenden Parameters π untersucht. Dieser Parameter wird im *StoppingCriterion* verwendet und ist in Gleichung (11) zu finden. Hierbei regelt π, wie streng das Verhalten des *StoppingCriterions* gegenüber Verbesserungen durch neue Bedingungen ist. Je strenger das Kriterium ist, desto höher ist die Wahrscheinlichkeit, dass eine Bedingung abgelehnt wird. Ziel ist es, einen eingegrenzten Wertebereich des Parameters zu liefern.
Nachdem ein geeigneter Wertebereich für π gefunden wurde, wird aus diesem ein konkreter Wert für Vergleiche des SBCA mit anderen Algorithmen gewählt. Diese Vergleiche werden auf neun Klassifikationsdatensätzen durchgeführt. Diese besitzen eine Klassenzugehörigkeit, die zwar bekannt ist und für die Bewertung verwendet wird, jedoch vom Algorithmus nicht zur Clusterfindung verwendet wird.
Die Klassifikationsdatensätze stammen aus dem Machine Learning Repository der Universität von Kalifornien, Irvine [1]. In Tabelle 4 befindet sich eine Zusammenfassung dieser Datensätze.
Die Verwendung von Clusteringdatensätzen wäre suboptimal, da man bei diesen keine Informationen über eine Zusammengehörigkeit der Daten besitzt und somit keine Kontrollergebnisse für eine Evaluation vorhanden sind. Prinzipiell wäre es möglich, die Ergebnisse Experten vorzulegen und diese dadurch zu evaluieren. Dies ist in der Realität jedoch nur schwer machbar, da hierdurch erheblicher Aufwand und Kosten anfallen würden.

	Syntetic	Iris	Thyroid	Ecoli	Glass	Vowel	Wine	Vehicle	Ionosphere	Sonar
Anzahl der Attribute	2	4	5	7	9	10	13	18	33	60
Anzahl der Klassen	4	3	3	8	6	11	3	4	2	2
Anzahl der Instanzen	48	150	215	336	214	990	178	846	351	208

Tabelle 4: Zusammenfassung der Datensätze, die in den Experimenten verwendet werden

5.1 Messung der Clusterqualität

Sowohl für die Wahl eines Parameters als auch für einen Vergleich unterschiedlicher Algorithmen werden Kriterien benötigt, um Aussagen über die Qualität der erlangten Ergebnisse treffen zu können.
Die typischen Ziele der Clusteranalyse sind eine hohe Intra-Cluster Gleichheit und eine geringe Inter-Cluster Gleichheit zu erreichen. Dies sind interne Kriterien für die Qualität eines Clusters. Zu beachten ist jedoch, dass gute Werte interner Kriterien nicht notwendigerweise gute Ergebnisse bei einer realen Anwendung bedeuten [22]. Hierfür werden externe Kriterien benötigt.
Bei externen Kriterien wird bewertet, wie gut die gefundene Cluster mit der tatsächlichen Zusammengehörigkeit übereinstimmt. Hierbei ist entweder die Klassenzugehörigkeit bekannt oder ein Experte stellt Wissen über den Datensatz zur Verfügung.
Im Folgenden werden die *Purity* und die *Normalized mutual information* als externe Kriterien für die Beurteilung der Qualität eines Clusterings verwendet. *Purity* ist eine einfache und transparente Maßeinheit für die Evaluation. *Normalized mutual information* kann informationstheoretisch interpretiert werden. Das vorhandene Wissen über diese stammt aus [22].
Um die *Purity* zu berechnen, wird jedem Cluster die Klasse, die im Cluster am häufigsten vorhanden ist, zugewiesen. Anschließend wird die *Accuracy* dieser Zuordnung durch Zählen der korrekt zugeordneten Instanzen und Teilen durch die gesamt Anzahl der Instanzen N berechnet. Formal ergibt dies:

$$purity(\Omega, \mathbb{C}) = \frac{1}{N} \sum_{k} \max_{j} |w_k \cap c_j| \tag{14}$$

Hierbei ist $\Omega = \{w_1, w_2, ..., w_K\}$ die Menge der Cluster und $\mathbb{C} = \{c_1, c_2, ..., c_J\}$ die Menge der Klassen. Somit sind w_k alle Elemente des Clusters k und c_j alle Elemente der Klasse j. Ein anschauliches Beispiel für die Berechnung der *Purity* ist in Abbildung 5 gegeben.

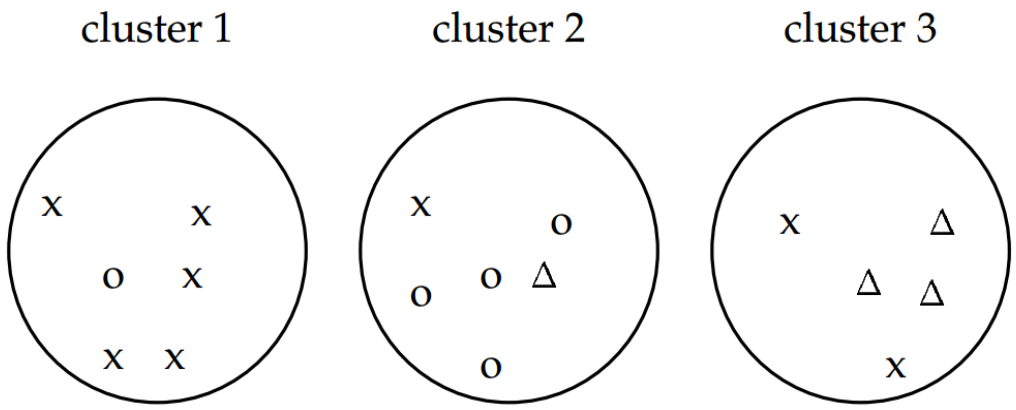


Abbildung 5: *Purity* als externes Kriterium für die Beurteilung der Qualität eines Clusters. Häufigste Klasse und Anzahl der Instanzen der häufigsten Klasse der drei Cluster sind: x,5 (cluster 1); o,4 (cluster 2); Δ,3 (cluster 3). Die *Purity* ist: $\frac{1}{17} \cdot (5+4+3) \approx 0.71$
Quelle: [22], Abbildung 16.4

Der Wertebereich der *Purity* liegt im Intervall [0,1]. Ein hoher Wert lässt hierbei auf ein gutes Clustering schließen. Werte nahe 0 sagen aus, dass Instanzen unterschiedlicher Klassen den gleichen Clustern angehören. Eine Purity von 1 bedeutet, dass jedes einzelne Cluster nur Elemente einer Klasse beinhaltet. Alle Klassen wären somit perfekt getrennt.
Ein Nachteil der *Purity* ist, dass sich diese leicht optimieren lässt. Hierzu muss lediglich die Anzahl der Cluster erhöht werden. Bildet beispielsweise jede Instanz ihr eigenes Cluster, so erzielt dieses Clustering eine *Purity* von 1. Aus diesem Grund ist es mithilfe der *Purity* nicht möglich, eine Abwägung zwischen Qualität der Cluster und Anzahl der Cluster zu treffen.
Ein Maß, welches diese Abwägung ermöglicht, ist die *Normalized mutual information* (NMI):

$$NMI(\Omega, \mathbb{C}) = \frac{I(\Omega; \mathbb{C})}{[H(\Omega) + H(\mathbb{C})]/2} \tag{15}$$

I ist die *Transinformation* (*mutual Information*) und ist wie folgt definiert:

$$I(\Omega; \mathbb{C}) = \sum_k \sum_j P(w_k \cap c_j) \log \frac{P(w_k \cap c_j)}{P(w_k)P(c_j)} \tag{16}$$

$$= \sum_k \sum_j \frac{|w_k \cap c_j|}{N} \log \frac{N|w_k \cap c_j|}{|w_k||c_j|} \tag{17}$$

$P(w_k)$ bzw. $P(c_j)$ aus Gleichung (16) sind die Wahrscheinlichkeiten, dass eine Instanz im Cluster w_k liegt bzw. der Klasse c_j angehört. $P(w_k \cap c_j)$ ist die Wahrscheinlichkeit, dass eine Instanz in der Schnittmenge von w_k und c_j liegt. N ist die Anzahl aller vorhandenen Instanzen. Gleichung (17) ist äquivalent zu Gleichung (16) mit einer Maximum-Likelihood-Schätzung der Wahrscheinlichkeiten.
H ist die Entropie und lässt sich nach einer Maximum-Likelihood-Schätzung der Wahrscheinlichkeiten wie folgt berechnen:

$$H(\Omega) = -\sum_k \frac{|w_k|}{N} \log \frac{|w_k|}{N} \tag{18}$$

Die *Transinformation* in Gleichung (16) misst das Wissen, das wir über die Klassenzugehörigkeiten der Instanzen gewinnen, wenn uns die gefunden Cluster vorliegen. Beträgt dieser Wert 0, so gibt uns das Wissen über die Clusterzugehörigkeit keine Informationen über die Klasse einer Instanz. Im umgekehrten Fall, wenn *I* den Wert 1 erreicht, liefert jedes Cluster die genaue Klasse der Instanzen. Dies geschieht jedoch auch, wenn jede Instanz ein eigenes Cluster bildet. Somit hat die *Transinformation* das gleiche Problem wie die *Purity*. Ideal wäre eine minimale Anzahl an Clustern bei maximaler *Transinformation*. Dies ist das Ziel der NMI.
Wie im Namen enthalten, wird bei der NMI die *Transinformation* normalisiert. Dies geschieht in Gleichung (15) durch den Nenner $[H(\Omega)+H(\mathbb{C})]/2$. Dieser kombiniert die Entropie der gefundenen Cluster und der vorhandenen Klassen. Da die Entropie mit steigender Anzahl an Clustern zunimmt, sinkt die NMI mit steigender Anzahl an Clustern. Der Wertebereich der NMI liegt ebenfalls im Intervall [0,1].
In Tabelle 5 ist ein Vergleich zwischen NMI und *Purity* zu sehen. Dieser wurde auf einen beispielhaften Datensatz mit vier gleich wahrscheinlichen Klassen durchgeführt. Bei nur einem Cluster beträgt die *Purity* $\frac{1}{4}$, da jede der vier Klasse gleich oft im Datensatz enthalten ist. Die NMI beträgt 0, da uns das Wissen über das Cluster einer Instanz keine Informationen über die Klassenzugehörigkeit liefert. Bei einem perfekten Clustering mit vier Cluster beträgt sowohl die *Purity* als auch der NMI 1. Nimmt nun die Anzahl an Clustern zu und gehören alle Instanzen innerhalb eines Clusters weiterhin der gleichen Klasse an, nimmt die NMI ab. Die *Purity* hingegen beträgt weiterhin 1.

	1 Cluster	4 Cluster	6 Cluster	16 Cluster
NMI	0	1	0.91	0.67
Purity	0.25	1	1	1

Tabelle 5: Vergleich zwischen NMI und *Purity* auf einen Datensatz mit 4 Klassen. Bei 1 Cluster sind alle Instanzen in diesem enthalten. Bei 4, 6 und 16 Cluster gehören alle Instanzen innerhalb eines Clusters der gleichen Klasse an.

Für die Evaluierung des SBCA werden die *Purity* und die NMI verwendet. Da die Ergebnisse interpretierbar sein sollen, und die Anzahl der Cluster der Anzahl der Klassen ähneln sollte, wird zusätzlich ein Wert *diffcc* eingeführt:

$$diffcc(c',c) = -||c'|-|c|| \tag{19}$$

Dieser berechnet den Unterschied zwischen der Anzahl der Cluster und der Anzahl der Klassen. Hierbei sind c' die gefundenen Cluster und c die vorhandenen Klassen. Der Maximalwert für *diffcc* liegt bei 0. Dieser liegt vor, wenn die gleiche Anzahl an Cluster und Klassen vorliegen. Existieren viele kleine Cluster mit sehr wenigen Instanzen, so wird *diffcc* einen stark negativen Wert annehmen. Dementsprechend ist ein hoher Wert für *diffcc* zu bevorzugen.
Durch Verwendung von NMI, *Purity* und *diffcc* kann die Qualität der Ergebnisse je nach Präferenz eines Anwenders beurteilt werden.

An dieser Stelle ist nochmals hervorzuheben, dass eine optimale Bewertung der Ergebnisse der Clusteranalyse nur durch Experten möglich ist. Dies ist jedoch in der Realität aus Kosten- und Zeitgründen nicht möglich. Wie auch in [23] oder [29] werden hier deshalb externe Kriterien für die Messung der Clusterqualität verwendet.

5.2 Auswahl eines Wertebereiches des Parameters

Der SBCA verwendet das in Abschnitt 4.2.4 vorgestellte *StoppingCriterion*. Dieses beinhaltet einen Parameter π, welcher die Strenge des Kriteriums regelt. Der Wertebereich des Parameters liegt im Intervall]0,1]. Dieser Wertebereich enthält jedoch viele Werte, die zu ungebräuchlichen Ergebnissen führen. Ein einzelnes Cluster für alle Instanzen oder ein einzelnes Cluster für jede Instanz wären Beispiele solcher

ungebräuchliche Ergebnisse. Im Folgenden soll ein kleinerer Wertebereich bestimmt werden, der auf vielen Datensätze gebräuchliche Ergebnisse liefert. Diese liegen vor, wenn die *Purity* einen Wert nahe 1 annimmt und zugleich der Wert für *diffcc* möglichst groß ist. Dies entspricht einer hohen NMI, da dieser den Informationsgehalt der Cluster mit der Anzahl der Cluster kombiniert.
Zur Bestimmung eines solchen Wertebereiches wird der SBCA mehrere Male auf einen einfachen synthetischen Datensatz und sieben Klassifikations-Datensätze ausgeführt. Der Anfangswert des Parameters π wird auf 0.1 gesetzt. Dieser wird jedes Mal, wenn alle Datensätze durchlaufen wurden, um 0.05 erhöht bis zu einem Maximalwert von 0.8. Nach jedem Durchlauf wird der *diffcc*, die *Purity* und die NMI der erhaltenen Ergebnisse berechnet. Anschließend wird abhängig von den erhaltenen Werten ein geeigneter Wertebereich für den Parameter π abgeleitet.

5.2.1 Ergebnisse

In Abbildung 6 sind die einzelnen Werte der NMI der Ergebnisse für verschiedene Datensätze und Parameter aufgetragen. Analog hierzu befindet sich in Abbildung 7 die Purity dieser Ergebnisse. In Abbildung 9 kann die Differenz zwischen Anzahl der Klassen und Anzahl der Cluster abgelesen werden.
Die erhaltenen Ergebnisse in Abbildung 6, 7 und 9 für die verschiedene π-Werte machen deutlich, dass es nicht möglich ist, einen optimalen Wert für diesen Parameter anzugeben. Während einige Datensätze (z.b. *Iris*) ihren maximalen NMI bereits bei $\pi = 0.4$ besitzen, existieren andere Datensätze (z.B. *Thyroid*), bei denen erst ab einen π-Wert von 0.7 eine NMI-Wert größer 0 zu erreichen ist.
Wie in Abbildung 6 zu sehen ist, erreicht der Mittelwert der NMI-Werte über alle Datensätze sein Maximum bei $\pi = 0.7$. Jedoch lassen sich auf manchen Datensätze bessere NMI-Werte mit einem geringeren Wert für π erzielen.

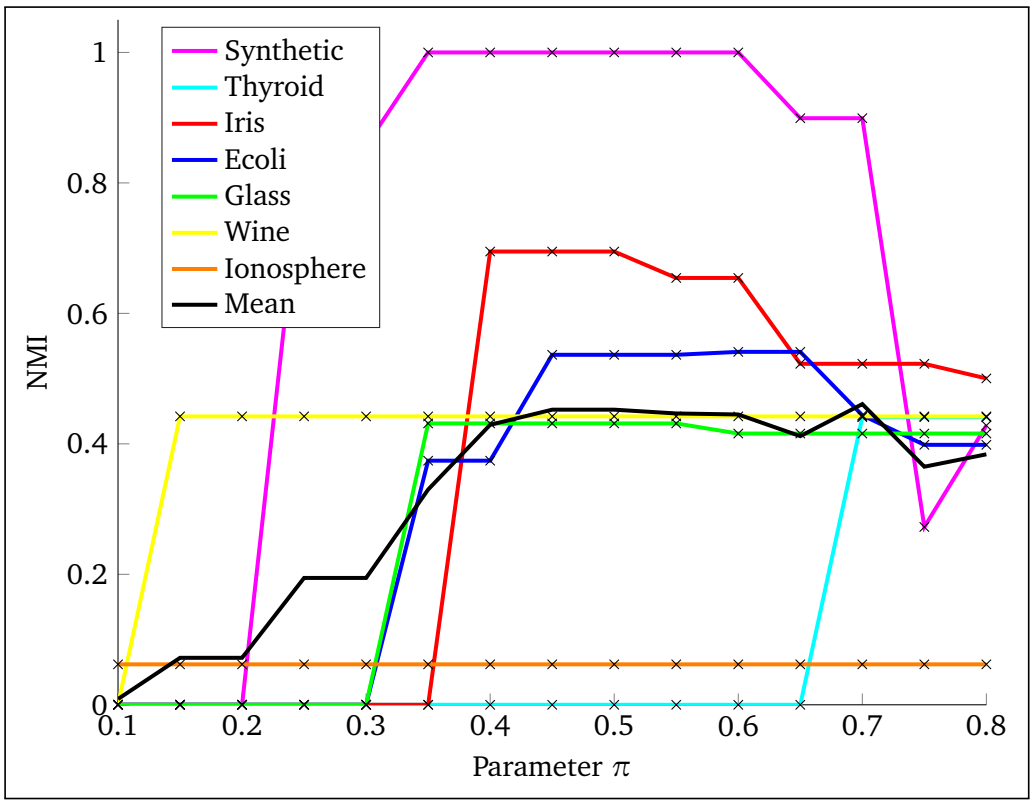


Abbildung 6: NMI der Ergebnisse des Separate-and-Conquer based Clustering Algorithm bei verschiedenen Parameter-Werten

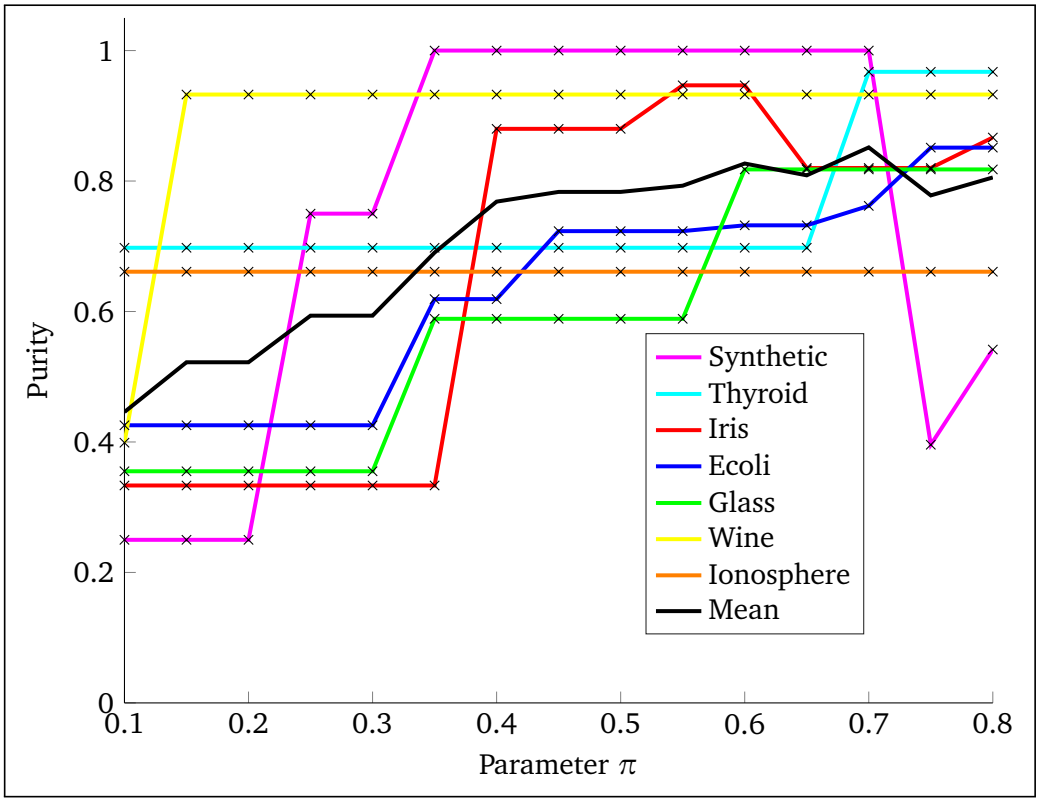


Abbildung 7: *Purity* der Ergebnisse des Separate-and-Conquer based Clustering Algorithm bei verschiedenen Parameter-Werten

Betrachtet man die *Purity* in Abbildung 7, so ist zu erkennen, dass diese mit steigendem π-Wert ebenfalls steigt. Dies liegt an der Funktionsweise dieses Parameters. Durch einen höheren Wert werden mehr Bedingungen zu einer Regel hinzugefügt, wodurch weniger Instanzen von dieser Regel abgedeckt werden und kleinere Cluster entstehen. Diese Cluster sind sehr spezifisch und beinhalten nur sehr ähnliche Instanzen. Da diese meist die gleiche Klasse besitzen, ergibt sich eine hohe *Purity*.
Wie in Abbildung 7 zu sehen ist, erzielt man für $\pi \geq 0.4$ eine hohe *Purity*. Allgemein gilt, dass eine hohe *Purity* durch einen hohen Wert für π erreicht werden kann.

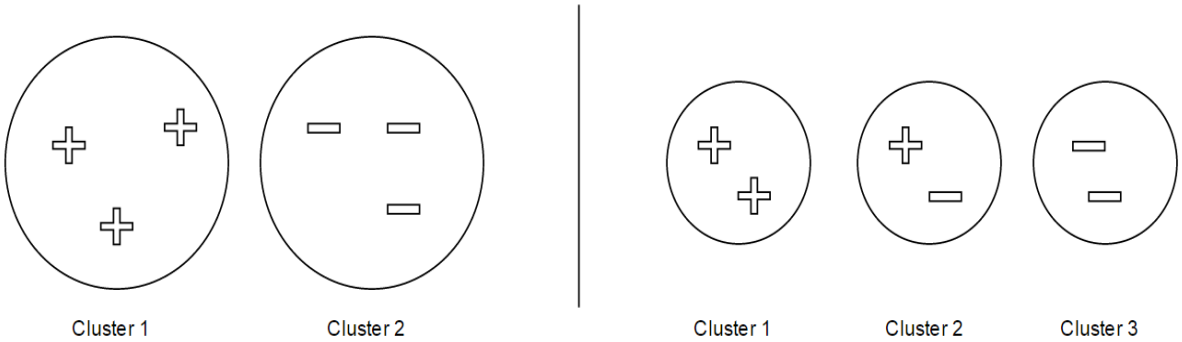


Abbildung 8: Beispiel für eine Senkung der *Purity* durch Erhöhung des Parameters π. Bei den Ergebnissen auf der linken Seite beträgt die *Purity*: $\frac{1}{6} \cdot (3+3) = 1$. Auf der rechten Seite der Grafik sind mögliche Ergebnisse nach Erhöhung des Parameters zu sehen. Da eine Neuzuordnung stattfindet, werden zwei unterschiedliche Instanzen dem gleichen Cluster zugewiesen. Die *Purity* auf der rechten Seite beträgt: $\frac{1}{6} \cdot (2+1+2) \approx 0.83$. Obwohl die Anzahl der Cluster stieg, ist die *Purity* gesunken.

Zu beachten ist jedoch, dass eine Erhöhung von π in einer niedrigeren *Purity* resultieren kann, auch wenn die Anzahl der Cluster zunimmt. Dies liegt daran, dass der SBCA bei unterschiedlichen Parametern eine Neuzuordnung aller Instanzen vornimmt. Ein anschauliches Beispiel befindet sich in Abbildung 8.
Ein invertiertes Verhalten zur *Purity* fällt in Abbildung 9 auf. Während die *Purity* steigt, nimmt die *diffcc* mit steigendem π ab. Da mehr und mehr Cluster entstehen, steigt die Differenz zwischen der Anzahl der gefundenen Cluster und der Anzahl der vorhandenen Klassen. Für $\pi \geq 0.65$ werden für einige Datensätze (z.B. Ecoli) bereits mehr als 20 Cluster gefunden.

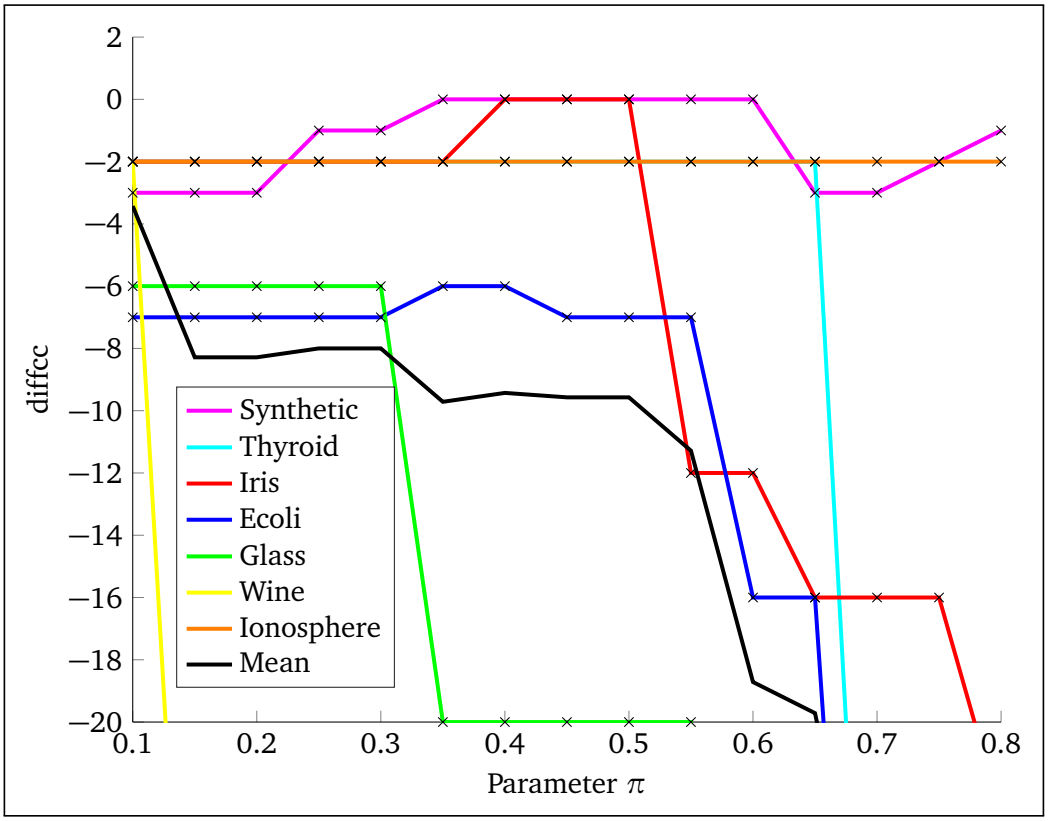


Abbildung 9: *diffcc* der Ergebnisse des Separate-and-Conquer based Clustering Algorithm bei verschiedenen Parameter-Werten

5.2.2 Schlussfolgerung

Obwohl es nicht möglich ist, einen einzigen optimalen Wert für π anzugeben, so ist es dennoch möglich einen Bereich zu bestimmen, in dem sowohl die *diffcc*, die *Purity* als auch die NMI einen hohen Wert erreichen. Im Wertebereich $0.4 \leq \pi \leq 0.6$ werden auf den getesteten Datensätze gute Werte für *diffcc* und *Purity* erzielt. Dieses Verhalten zeigt sich auch bei der NMI. Wie in Abbildung 6 deutlich zu erkennen ist, wird im Bereich $0.4 \leq \pi \leq 0.6$ ein hoher NMI-Mittelwert erzielt. Da das Maximum des NMI-Mittelwertes der verwendeten Datensätze bei $\pi = 0.7$ liegt, wird dieser Wert ebenfalls in den Wertebereich mit aufgenommen.
Somit ergibt sich für den Parameter π ein angepasster Wertebereich, der im Intervall [0.4, 0.7] liegt. Werte außerhalb dieses Intervalles sollten vermieden werden, da sie meist zu schlechten Ergebnissen führen. Die genaue Wahl des Parameters hängt sowohl stark vom verwendeten Datensatz als auch von den Präferenzen des Anwenders ab. Um Ergebnisse mit hoher *Purity* zu erzielen, sollte ein π nahe 0.7 gewählt werden. Steht die Interpretierbarkeit und somit die Anzahl der Cluster im Vordergrund, sollte ein π nahe 0.4 gewählt werden.

5.3 Vergleich mit anderen Algorithmen

In diesem Abschnitt wird eine experimentelle Evaluation durchgeführt, in dem SBCA mit K-Means [15], COBWEB [9], FRBC [23] und dem EM-Algorithmus [8] verglichen wird. K-Means, EM und COBWEB dienen als Referenzwert. Der FRBC ist eine State-of-the-Art Methode für interpretierbares Clustering.
Die EM-Algorithmen gehören zu den partitionierenden Clusterverfahren. Bei diesen Verfahren muss der Anwender die Anzahl der Cluster k vorher definieren. Anschließend werden k Clustermittelpunkte zufällig bestimmt und iterativ verschoben, um eine vorgegebene Fehlerfunktion zu minimieren. EM-Algorithmen bestehen aus zwei sich wiederholenden Schritten, bei denen die Daten zu den unterschiedlichen Clustern zugeordnet werden (Expectation-Schritt) und anschließend die Parameter an die neue Zuordnung angepasst werden (Maximization-Schritt). Diese Schritte werden wiederholt bis keine wesentliche Verbesserung mehr stattfindet.
Der K-Means hat starke Ähnlichkeiten mit dem EM-Algorithmus und zeichnet sich durch seine Einfachheit aus. Auch bei diesem Algorithmus muss der Anwender die Anzahl der Cluster k vorher definieren. Anschließend werden k Clustermittelpunkte zufällig bestimmt. Als nächstes wird jede Instanz dem Cluster, dessen Mittelpunkt am nächsten zur jeweiligen Instanz ist, zugewiesen. Anhand dieser neuen Zuordnung werden die Clustermittelpunkte abhängig von den Instanzen, die einem Cluster angehören, neu Berechnet. Dieses Zuordnen und berechnen der Mittelpunkte wird solange wiederholt, bis keine Instanz einem neuem Cluster zugrordnet wird.
COBWEB ist ein inkrementeller Algorithmus für hierarchisches Clustering, bei dem ein Entscheidungsbaum erzeugt wird. Er beginnt mit einem leeren Wurzelknoten und fügt jede Instanz einzeln dem Baum hinzu. Für das Hinzufügen betrachtet COBWEB vier verschiedene Operationen, die eine neue Instanz einer vorhandenen Klasse zuordnen, eine neue Klasse erzeugen, zwei Klassen kombinieren oder eine vorhandene Klasse teilen.
FRBC benutzt einen Klassifikations-Regellerner zum Finden der einzelnen Cluster. Dies ist möglich, da in jeder Iteration des FRBC eine Menge von Hilfsdaten generiert und zu den Eingabedaten hinzugefügt wird. Diese beiden Datenmengen bilden ein Zwei-Klassen-Problem, welches mithilfe des Regellerners SGERD gelöst wird. Iterativ werden so die einzelnen Cluster gefunden und abgedeckte Instanzen aus der Datenmenge entfernt bis eine Abbruchbedingung greift. Diese besitzt einen Parameter τ als Schwellenwert, der vom Anwender gesetzt werden muss und der die Strenge dieser Abbruchbedingung regelt. Die Vorgehensweise des FRBC ist die, die der Vorgehensweise des SBCA am ähnlichsten ist. Ein Vergleich wird aus zeitlichen Gründen deshalb nur mit dem FRBC als Vertreter für State-of-the-Art Methoden durchgeführt. Eine ausführliche Beschreibung des FRBC befindet sich in Abschnitt 6.2.
Die Ergebnisse des FRBC wurden aus der dazugehörigen Arbeit [23] entnommen. Bei den Datensätzen Thyroid, Ecoli, Glass, Vowel und Vehicle wurden die Ergebnisse des Schwellenwertes $\tau = 0.01$ verwendet, da diese hierdurch eine bessere NMI erzielen. Bei allen anderen Datensätzen wurden die Ergebnisse für $\tau = 0.1$ benutzt. Das Entnehmen der Ergebnisse aus dieser Quelle ist möglich, da der FRBC nicht auf einer zufälligen Initialisierung basiert. Anders als der K-Means, COBWEB und EM wird bei der Initialisierung des FRBC kein sogenannter *Random Seed* benötigt, von dem die erhaltenen Ergebnisse abhängen.
Die verwendeten Implementierungen des K-Means, des COBWEB und des EM-Algorithmus stammen aus Weka, einer Sammlung von Algorithmen für maschinelles Lernen der Universität von Waikato [13]. Da diese wie erwähnt auf einen *Random Seed* beruhen und je nach Initialisierung andere Ergebnisse erzeugen, wird jeder dieser Algorithmen in allen Experimenten mit zehn zufälligen Initialisierungen ausgeführt. Zum Vergleich mit dem SBCA wird lediglich der Beste dieser Durchläufe, im Bezug auf die NMI der Ergebnisse, verwendet.
Der SBCA wird in einer ähnlichen Form wie in den vorherigen Experimenten verwendet. In diesen wurde ein Wertebereich des Parameters bestimmt. Dieser Bereich liegt im Intervall [0.4,0.7]. Wie bereits erwähnt, sorgen hohe Werte für eine höhere *Purity* der Ergebnisse. Werte nahe 0.4 sind dagegen interpretierbarer, da hierdurch weniger Cluster entstehen und dies für einen menschlichen Anwender überschaubarer ist. Da das Ziel des SBCA darin liegt, interpretierbare Ergebnisse zu liefern, wird für

die Vergleiche mit anderen Algorithmen der Wert $\pi = 0.45$ verwendet. Dies kann eine geringere *Purity* und eine geringere NMI zu Folge haben, erleichtert jedoch das Interpretieren der Ergebnisse, da die Anzahl der Cluster im Durchschnitt sinkt.
Trotz des niedrigen π, werden bei manchen Datensätzen (z.B.Ecoli) zu viele kleine Cluster erzeugt. Ein anschaulicher Vergleich und eine Darstellung wären somit nur bedingt möglich. Aus diesem Grund wird für die folgenden Experimente zusätzlich das *ClusterStoppingCriterion* verwendet. Dieses sorgt für das Terminieren des SBCA, wenn es nicht sinnvoll ist, in den verbleibenden Daten weiterhin Cluster zu suchen und verhindert somit, dass sehr viele kleine Cluster entstehen. Das *ClusterStoppingCriterion* beinhaltet einen Parameter μ, der in Gleichung (13) Verwendung findet und der die Anzahl der Cluster und somit *Purity*, *diffcc* und NMI beeinflusst. In zukünftigen Arbeiten und Experimenten mit dem SBCA ist es daher notwendig, einen geeigneten Wertebereich für diesen Parameter zu definieren, um optimale Ergebnisse zu erzielen. In den folgenden Experimenten wird dieser Parameter μ auf 5.5 gesetzt. Dies bedeutet, dass der Algorithmus terminiert, wenn die durchschnittliche Distanz der verbleibenden Instanzen größer als das 5.5-fache der durchschnittliche Distanz aller Eingabedaten ist.

5.3.1 Ergebnisse

Die entstandenen Clusterings des SBCA nach Anwendung auf den neun Klassifikationsdatensätzen sind in den Tabellen 9 bis 17 zu finden. In diesen sind als Benchmark auch die Clusterings des K-Means, COBWEB, EM und FRBC dargestellt.
In diesen Tabellen sind Matrizen dargestellt, die die Genauigkeiten der Clusterings aufzeigen. Obwohl diese Matrizen den Konfusionsmatrizen [20], die in der Klassifikation üblich sind, sehr ähnlich sind, unterscheiden sie sich in einigen Aspekten. Konfusionsmatrizen zeigen die wirkliche und die vorhergesagte Klasse aller Instanzen einer Datenmenge und haben stets die Größe $L \times L$, wobei L die Anzahl der vorhandenen Klassen ist. Die hier verwendeten Matrizen hingegen zeigen die wirkliche Klasse und das Cluster, in das eine Instanz eingeordnet wurde. Sie sind $L \times G$ groß, wobei L die Anzahl der Klassen und G die Anzahl der Cluster ist.
Jede Zelle $c_{i,j}$ der Matrizen enthält die Anzahl der Instanzen der Klasse i, die in Cluster j enthalten sind. Dementsprechend ergibt die Summe der Einträge der Zeile i einer Matrix die Anzahl der Instanzen der Klasse i. Analog ergibt die Summe der Einträge der Spalte j die Anzahl der Instanzen im Cluster j.
Bei den dargestellten Matrizen ist die Gleichheit der Klassen innerhalb eines Clusters wichtig und nicht, welche Klasse in welchem Cluster ist. Dies bedeutet, es ist egal, ob beispielsweise Instanzen der Klasse 1 im fünften Cluster sind. Wichtig wäre jedoch, dass die Mehrheit der Instanzen des fünften Clusters der Klasse 1 angehören. Da die Reihenfolge der Cluster unwichtig ist, wurden in den Tabellen 9 bis 17 die Reihenfolgen der Cluster geändert, damit die Ergebnisse lesbarer werden und ein Vergleich einfacher ist. So kann es beispielsweise vorkommen, dass das fünfte gefundene Cluster in der ersten Spalte mit der Clusterbezeichnung *1* dargestellt wird.
Der Idealfall ist, dass alle Instanzen einer Klasse von dem gleichen Cluster abgedeckt werden und gleichzeitig jedes Cluster nur Instanzen einer Klasse beinhaltet. Dies entspräche einem Eintrag ungleich 0 in jeder Spalte und einem Eintrag ungleich 0 in jeder Zeile.
Wie in den Tabellen zu sehen ist, erzielt der SBCA auf fast allen Datensätze gute Ergebnisse, die mit den anderen Algorithmen vergleichbar sind. Ausnahmen bilden die Datensätze *Vehicle*, *Wine* und *Vowel*, da die Anzahl der gefundenen Cluster hier unangemessen hoch ist.
Für einen präziseren Vergleich der Algorithmen sind in Tabelle 6 die erzielten NMI- und Purity-Werte der Ergebnisse auf den unterschiedlichen Datensätzen angegeben. Zusätzlich ist die Anzahl der gefundenen Cluster angegeben. Bei der NMI und der *Purity* ist zu beachten, dass die im SBCA verwendeten Parameterwerte für diese nicht optimal sind. Durch die Wahl geeigneter Parameter für das *StoppingCriterion* und für das *ClusterStoppingCriterion* können eine höhere NMI und *Purity* auf den meisten Datensätzen erreicht werden.

Datensatz		K-Means	COBWEB	EM	FRBC	SBCA
Iris	# Clusters	3	2	5	3	3
	NMI	0.7419	0.6565	0.6844	**0.7837**	0.6948
	Purity	0.8867	0.6667	**0.9000**	0.8867	0.8800
Thyroid	# Clusters	3	124	3	3	1
	NMI	0.5966	0.2990	**0.8316**	0.6704	0.0000
	Purity	0.8884	**0.9907**	0.9674	0.9163	0.6977
Ecoli	# Clusters	8	1	5	3	5
	NMI	0.6203	0.0000	**0.6595**	0.1919	0.5084
	Purity	**0.8274**	0.4256	0.7798	0.5387	0.6905
Glass	# Clusters	6	9	6	3	8
	NMI	0.3677	0.3411	**0.3892**	0.2236	0.3555
	Purity	**0.5514**	0.5140	**0.5514**	0.4439	0.4907
Vowel	# Clusters	11	2	27	8	281
	NMI	0.4006	0.1875	0.4945	0.3362	**0.5183**
	Purity	0.3667	0.1747	0.5111	0.3111	**0.8172**
Wine	# Clusters	3	156	4	3	39
	NMI	**0.8417**	0.3490	0.7350	0.7681	0.4421
	Purity	0.9494	**0.9831**	0.9270	0.9157	0.9326
Vehicle	# Clusters	4	785	10	3	104
	NMI	0.1540	**0.3365**	0.2701	0.1092	0.2237
	Purity	0.4019	**0.9740**	0.5662	0.3783	0.6005
Ionosphere	# Clusters	2	1	6	2	4
	NMI	0.1349	0.0000	**0.3529**	0.0647	0.0620
	Purity	0.7123	0.6410	**0.8974**	0.6410	0.6610
Sonar	# Clusters	2	1	8	2	7
	NMI	0.0111	0.0000	**0.1291**	0.0001	0.0848
	Purity	0.5577	0.5337	**0.7067**	0.5337	0.5481

Tabelle 6: NMI, Purity und Anzahl der Cluster der unterschiedlichen Algorithmen für verschiedene Datensätze

Trotz der für die NMI und *Purity* suboptimalen Parameter erzielt der SBCA auf den Datensätzen *Ecoli*, *Glass*, *Vowel*, *Vehicle* und *Sonar* eine etwas höhere NMI als der FRBC. Eine bessere *Purity* erzielt der SBCA auf allen Datensätzen außer *Iris* und *Thryoid*. Auf der anderen Seite bildet der SBCA auf den Datensätzen *Vowel*, *Wine* und *Vehicle* eine unangemessen hohe Anzahl an Clustern.
Betrachtet man die Ergebnisse des COBWEB, so stellt man auch hier eine unangemessen hohe Anzahl an Clustern bei einigen Datensätzen fest. COBWEB erzielt zudem nur auf den Datensatz *Thyroid* eine höhere NMI als der SBCA. Bei diesem fand COBWEB jedoch 124 Cluster.
Vergleicht man den SBCA mit dem K-Means und dem EM-Algorithmus, so schneidet der SBCA etwas schlechter ab. Auf den Datensätzen *Iris*, *Ecoli*, *Glass*, *Sonar* und *Vehicle* befinden sich die *Purity* und die NMI der Ergebnisse des SBCA, des K-Means und des EM-Algorithmus etwa im gleichen Bereich. Jedoch erzielt der SBCA in den meisten Fällen eine geringere NMI und *Purity*. Nicht zu vergessen ist hier jedoch, dass die Cluster, die durch den SBCA gefunden wurden, für menschliche Anwender einfacher zu interpretieren sind. Während K-Means und EM lediglich die Mittelpunkte der Cluster liefern, erzeugt der SBCA eine Regelmenge.

Iris

petallength $\geq 3.85 \wedge$ petallength <5.05	$\Rightarrow$ Cluster 2
petallength <3.15	$\Rightarrow$ Cluster 1
else	$\Rightarrow$ Cluster 3

Ecoli

alm1 <0.465	$\Rightarrow$ Cluster 1
alm2 ≥ 0.645	$\Rightarrow$ Cluster 2
gvh $\geq 0.77 \wedge$ aac $\geq 0.48 \wedge$ mcg <0.7	$\Rightarrow$ Cluster 5
alm2 ≥ 0.53	$\Rightarrow$ Cluster 4
else	$\Rightarrow$ Cluster 3

Glass

Mg ≥ 3.295	$\Rightarrow$ Cluster 1
Mg $\geq 2.695 \wedge$ Fe ≥ 0.145	$\Rightarrow$ Cluster 3
Mg $\geq 1.84 \wedge$ K $<0.04 \wedge$ Ca <9.445	$\Rightarrow$ Cluster 6
Mg $\geq 2.7 \wedge$ Na <13.175	$\Rightarrow$ Cluster 4
Na $\geq 14.34 \wedge$ Al $\geq 2.155 \wedge$ Ba ≥ 0.65	$\Rightarrow$ Cluster 7
Si $\geq 73.45 \wedge$ Ca ≥ 9.605	$\Rightarrow$ Cluster 5
Na $\geq 14.34 \wedge$ Al ≥ 2.4	$\Rightarrow$ Cluster 8
else	$\Rightarrow$ Cluster 2

Tabelle 7: Die durch den SBCA erzeugten Regelmengen für die Datensätze *Iris*, *Ecoli* und *Glass*

Zur Veranschaulichung der Ergebnisse sind in Tabelle 7 die durch den SBCA erzeugten Regelmengen der Datensätze *Iris*, *Ecoli* und *Glass* angegeben. Die Regeln wurden in der angegebenen Reihenfolge (von oben nach unten) gefunden und bilden eine geordnete Entscheidungsliste. Ob das Cluster der ersten gefundenen Regel beispielsweise Cluster 1 oder Cluster 2 heißt, spielt dabei keine Rolle, da es sich nur um eine Bezeichnung handelt. Deshalb ist es möglich, die Clusterbezeichnungen so abzuändern, dass sie mit den gegebenen Matrizen übereinstimmen.
Wie bei den erzeugten Regeln des Datensatzes *Iris* in Tabelle 7 sofort zu erkennen ist, ist *petallength* das ausschlaggebende Attribut für die Unterteilung in unterschiedliche Cluster. Mit der ersten Regel werden alle gegebenen Instanzen überprüft. Die Instanzen, deren *petallength* größer oder gleich 3.85 und zugleich kleiner als 5.05 ist, gehören den Cluster mit der Clusterbezeichnung 2 an. Da es sich um eine geordnete Entscheidungsliste handelt, wird die zweite Regel nur auf Instanzen, auf die die erste Regel nicht zutrifft, überprüft. Hat eine der verbleibenden Instanzen eine *petallength* kleiner als 3.15, so wird sie dem Cluster 1 zugewiesen. Alle Instanzen, bei denen keine der bisherigen Regeln gegriffen hat, werden von der Default-Regel abgedeckt und dem letzten Cluster zugewiesen.
Solch ein strukturiertes Vorgehen ist bei einer Regelmenge sehr einfach möglich. Durch Betrachten jeder einzelnen Regel erkennt man das Attribut, welches als Indikator des dazugehörigen Clusters dient. Ein analoges Vorgehen ist bei den Regelmengen der Datensätze *Ecoli* und *Glass* in Tabelle 7 möglich. Im Falle des Datensatzes *Ecoli* wird zuerst überprüft, ob der Attributwert von alm1 eines Objektes kleiner als 0.465 ist. Dies entspricht der ersten gefundenen Regel. Ist diese erfüllt, so wird das Objekt dem Cluster 1 zugewiesen. Ist Regel 1 nicht erfüllt, so werden anschließend die Regeln 2, 3 und 4 der Reihe nach überprüft, bis eine der Regeln erfüllt ist. Das Objekt wird dem zur Regel korrespondierenden Cluster zugewiesen. Erfüllt das Objekt keine dieser Regeln, so wird es dem Cluster 3 zugewiesen. Das Vorgehen für die Regelmenge des Datensatzes Glass ist das gleiche. Die Regeln werden der Reihe nach überprüft, und ein Objekt dem Cluster zugewiesen, dessen Regel erfüllt ist.

Iris

Attribute	Cluster 3	Cluster 1	Cluster 2
sepallength	6.8462	5.006	5.8885
sepalwidth	3.0821	3.418	2.7377
petallength	5.7026	1.464	4.3967
petalwidth	2.0795	0.244	1.418

Ecoli

Attribute	Cluster 2	Cluster 7	Cluster 5	Cluster 3	Cluster 8	Cluster 4	Cluster 1	Cluster 6
mcg	0.3135	0.663	0.6972	0.3722	0.7492	0.6664	0.384	0.6735
gvh	0.3742	0.537	0.4938	0.4848	0.4485	0.7214	0.438	0.681
lip	0.48	1	0.48	0.48	0.48	0.48	0.48	0.48
chg	0.5	0.55	0.5	0.5	0.5	0.5	0.5	0.5
aac	0.4698	0.549	0.5826	0.554	0.4492	0.4266	0.4355	0.7375
alm1	0.2351	0.618	0.7772	0.7692	0.6762	0.464	0.3961	0.449
alm2	0.3262	0.431	0.7875	0.7743	0.6038	0.3496	0.4493	0.3085

Glass

Attribute	Cluster 3	Cluster 1	Cluster 5	Cluster 2	Cluster 6	Cluster 4
RI	1.5181	1.5221	1.5174	1.5173	1.5163	1.5236
Na	13.0778	13.8645	14.2067	13.159	14.4221	12.7263
Mg	3.3975	3.6923	2.3708	3.4984	0.0269	0.3305
Al	1.2957	0.8518	1.9175	1.3888	2.1793	1.2979
Si	72.706	71.7232	72.0567	72.802	73.1548	72.4842
K	0.5248	0.1386	0.6392	0.5595	0.6183	0.2768
Ca	8.7345	9.5832	8.0867	8.3838	8.6448	12.5016
Ba	0.0252	0.0364	0.6217	0.0015	0.8586	0.1658
Fe	0.2178	0.0214	0	0.0123	0.0134	0.0789

Tabelle 8: Die durch den K-Means erzeugten Mittelwerte der Cluster für die Datensätze *Iris*, *Ecoli* und *Glass*

Im Gegensatz dazu sind in Tabelle 8 die durch den K-Means erzeugten Mittelpunkte der Cluster gegeben. Die Ergebnisse anderer Algorithmen, welche die gefundenen Cluster durch Mittelpunkte darstellen, sehen annähernd gleich aus. Das Folgende gilt somit auch für andere Algorithmen mit mittelpunktbasierter Clusterdarstellung. Ein einfaches Vorgehen zum Einordnen einer Instanz ist in Tabelle 8 für einen menschlichen Anwender kaum möglich. Dies liegt daran, dass jedes Cluster durch die Mittelwerte aller im Datensatz vorkommenden Attribute definiert wird. Des Weiteren ist es kaum möglich, die Grenzen zweier Cluster ohne zusätzlichen Rechenaufwand zu bestimmen.
Ein deutliches Beispiel bietet ein exemplarisches Objekt des Datensatzes *Iris* mit dem Attributwert 1 bei den Attributen sepallength, sepalwidth, petallength und petalwidth. Bei der Regelmenge in Tabelle 7 ist sofort zu erkennen, dass die erste Regel nicht erfüllt ist, da die petallength kleiner als 3.85 ist, jedoch die zweite Regel erfüllt ist, da petallength kleiner als 3.15 ist. Somit wird das exemplarische Objekt dem Cluster 1 zugewiesen. Bei einer Mittelpunktdarstellung wie in Tabelle 8 werden die Distanzen vom zu überprüfenden Objekt zu jedem Clustermittelpunkt berechnet. Diese Berechnungen sind in den Gleichungen (20), (21) und (22) zu finden. Das Objekt wird dem Cluster mit der geringsten Entfernung zugewiesen. In diesem Beispiel wäre dies Cluster 1.

$$d_{Cluster3} = \sqrt{(1-6.8462)^2 + (1-3.0821)^2 + (1-5.7026)^2 + (1-2.0795)^2} \approx 7.8609.. \quad (20)$$

$$d_{Cluster1} = \sqrt{(1-5.006)^2 + (1-3.418)^2 + (1-1.464)^2 + (1-0.244)^2} \approx 4.7625 \quad (21)$$

$$d_{Cluster2} = \sqrt{(1-5.8885)^2 + (1-2.7377)^2 + (1-4.3967)^2 + (1-1.418)^2} \approx 6.2153 \quad (22)$$

Wie in diesem Beispiel zu erkennen ist, muss zu jedem gefundenem Cluster die Distanz berechnet werden. Dies ist für gewöhnlich für einen menschlichen Anwender nicht ohne viel Rechenaufwand möglich. Je mehr Attribute ein Datensatz beinhaltet, desto komplizierter wird diese Berechnung. Ein analoges Beispiel für die Datensätze *Ecoli* und *Glass* wäre somit noch komplizierter, da sowohl mehr Cluster gefunden wurden als auch mehr Attribute existieren. Das einfache gezeigt Beispiel mit dem Datensatz *Iris* verdeutlicht bereits, wie schwer eine Zuordnung einer Instanz zu einem Cluster fällt, falls nur die Mittelpunkte der Cluster gegeben sind.

Klasse	K-Means			COBWEB		EM					FRBC			SBCA		
	Cluster			Cluster		Cluster					Cluster			Cluster		
	1	2	3	1	2	1	2	3	4	5	1	2	3	1	2	3
1	50	0	0	50	0	28	22	0	0	0	50	0	0	50	0	0
2	0	47	3	3	47	0	0	23	27	0	0	50	0	1	41	8
3	0	14	36	0	50	0	0	0	15	35	0	17	33	0	9	41

Tabelle 9: Clustering des Datensatzes *Iris* mit verschiedenen Algorithmen

Klasse	K-Means			COBWEB	EM			FRBC			SBCA
	Cluster			Cluster	Cluster			Cluster			Cluster
	1	2	3	124 gesamt	1	2	3	1	2	3	1
1	150	0	0	hier nicht darstellbar	148	0	2	150	0	0	150
2	17	18	0		1	34	0	12	23	0	35
3	7	0	23		4	0	26	6	0	24	30

Tabelle 10: Clustering des Datensatzes *Thyroid* mit verschiedenen Algorithmen

Klasse	K-Means			COBWEB	EM				FRBC			SBCA
	Cluster			Cluster	Cluster				Cluster			Cluster
	1	2	3	156 gesamt	1	2	3	4	1	2	3	39 gesamt
1	59	0	0	hier nicht darstellbar	50	0	9	0	59	0	0	hier nicht darstellbar
2	2	62	7		0	45	22	4	11	56	4	
3	0	0	48		0	0	0	48	0	0	48	

Tabelle 11: Clustering des Datensatzes *Wine* mit verschiedenen Algorithmen

Klasse	K-Means		COBWEB	EM					FRBC		SBCA			
	Cluster		Cluster	Cluster					Cluster		Cluster			
	1	2	1	1	2	3	4	5	6	1	2	1	2	3	4
1	93	33	126	63	48	14	1	0	0	114	111	118	8	0	0
2	68	157	225	0	21	128	20	36	20	27	99	214	1	8	2

Tabelle 12: Clustering des Datensatzes *Ionosphere* mit verschiedenen Algorithmen

Klasse	K-Means		COBWEB	EM								FRBC		SBCA						
	Cluster		Cluster	Cluster								Cluster		Cluster						
	1	2	1	1	2	3	4	5	6	7	8	1	2	1	2	3	4	5	6	7
1	60	37	97	39	21	16	13	7	1	0	0	49	48	94	3	0	0	0	0	0
2	55	56	111	25	4	17	8	16	23	12	6	55	56	100	0	3	2	2	2	2

Tabelle 13: Clustering des Datensatzes *Sonar* mit verschiedenen Algorithmen

Klasse	K-Means											COBWEB		EM
	Cluster											Cluster		Cluster
	1	2	3	4	5	6	7	8	9	10	11	1	2	27 gesamt
1	49	21	20	0	0	0	0	0	0	0	0	89	1	hier nicht darstellbar
2	30	23	26	0	1	0	0	0	0	0	10	90	0	
3	5	14	13	32	26	0	0	0	0	0	0	90	0	
4	0	6	0	52	26	0	0	0	0	0	6	87	3	
5	0	0	0	6	3	39	3	22	12	0	5	34	56	
6	0	0	0	36	28	12	0	0	12	0	2	64	26	
7	0	0	0	0	0	27	11	27	21	4	0	22	68	
8	0	0	0	0	0	6	45	28	10	1	0	7	83	
9	0	0	0	0	6	1	16	15	15	18	19	37	53	
10	8	0	16	0	0	0	24	6	3	24	9	33	57	
11	0	0	0	13	28	5	0	0	10	6	28	76	14	

Klasse	FRBC								SBCA
	Cluster								Cluster
	1	2	3	4	5	6	7	8	281 gesamt
1	53	27	0	0	0	0	0	10	hier nicht darstellbar
2	45	12	24	0	0	0	0	9	
3	35	0	31	17	0	7	0	0	
4	0	0	3	56	20	11	0	0	
5	5	0	14	32	18	21	0	0	
6	0	0	2	28	35	10	3	12	
7	0	0	0	5	54	0	25	6	
8	3	3	3	1	25	13	40	2	
9	11	12	0	0	9	6	46	6	
10	4	4	4	37	12	15	14	0	
11	43	34	0	0	0	0	0	13	

Tabelle 14: Clustering des Datensatzes *Vowel* mit verschiedenen Algorithmen

Klasse	K-Means								COBWEB	EM				
	Cluster								Cluster	Cluster				
	1	2	3	4	5	6	7	8	1	1	2	3	4	5
1	76	64	0	2	0	1	0	0	143	141	0	2	0	0
2	6	0	39	1	26	0	1	4	77	8	40	1	27	1
3	2	1	0	44	1	2	0	2	52	7	0	44	1	0
4	1	0	1	0	25	1	1	7	35	0	1	1	32	1
5	0	0	0	2	0	17	1	0	20	0	0	19	0	1
6	0	0	0	0	0	0	5	0	5	0	0	0	0	5
7	0	0	0	0	0	0	2	0	2	0	0	0	0	2
8	0	0	0	1	1	0	0	0	2	0	0	1	1	0

Klasse	FRBC			SBCA				
	Cluster			Cluster				
	1	2	3	1	2	3	4	5
1	20	21	102	132	0	0	11	0
2	58	11	8	1	66	8	2	0
3	1	0	1	26	1	21	3	1
4	1	0	1	0	33	1	1	0
5	30	1	4	11	0	7	0	2
6	2	1	17	0	0	5	0	0
7	1	0	4	0	1	1	0	0
8	9	5	38	0	1	1	0	

Tabelle 15: Clustering des Datensatzes *Ecoli* mit verschiedenen Algorithmen

Klasse	K-Means						COBWEB									EM					
	Cluster						Cluster									Cluster					
	1	2	3	4	5	6	1	2	3	4	5	6	7	8	9	1	2	3	4	5	6
1	17	38	15	0	0	0	70	0	0	0	0	0	0	0	0	20	46	0	4	0	0
2	2	41	21	11	1	0	64	4	2	1	1	1	0	1	2	2	51	11	8	4	0
3	2	12	3	0	0	0	17	0	0	0	0	0	0	0	0	5	11	0	1	0	0
4	0	0	1	7	2	3	5	0	0	0	0	0	1	6	1	0	0	9	0	1	3
5	0	1	0	1	4	3	5	0	0	0	0	1	0	2	1	4	0	3	0	1	1
6	1	0	0	0	5	23	5	0	0	0	0	0	0	0	24	0	0	0	1	7	21

Klasse	FRBC			SBCA							
	Cluster			Cluster							
	1	2	3	1	2	3	4	5	6	7	8
1	45	25	0	66	2	1	1	0	0	0	0
2	40	25	11	53	15	4	3	1	0	0	0
3	12	5	0	17	0	0	0	0	0	0	0
4	2	3	8	0	11	0	0	2	0	0	0
5	5	0	4	0	6	0	0	1	2	0	0
6	4	0	25	1	22	0	0	0	0	2	4

Tabelle 16: Clustering des Datensatzes *Glass* mit verschiedenen Algorithmen

Klasse	K-Means				COBWEB	EM									
	Cluster				Cluster	Cluster									
	1	2	3	4		1	2	3	4	5	6	7	8	9	10
1	107	59	15	31		44	31	30	24	10	0	21	34	18	0
2	105	62	16	34	hier nicht	37	36	34	21	12	0	29	34	14	0
3	49	68	101	0	darstellbar	0	1	29	10	86	28	27	1	34	2
4	1	78	66	54		0	0	0	0	0	0	1	92	100	6

Klasse	FRBC			SBCA
	Cluster			Custer
	1	2	3	104 gesamt
1	114	3	82	
2	63	115	39	hier nicht
3	72	57	89	darstellbar
4	60	117	53	

Tabelle 17: Clustering des Datensatzes *Vehicle* mit verschiedenen Algorithmen

5.3.2 Schlussfolgerung

Die Ergebnisse, die der SBCA auf einigen Datensätzen in Hinsicht auf NMI und Purity erzielt, sind ähnlich wie die von klassischen Algorithmen wie dem K-Means oder dem EM-Algorithmus. Auch wenn die klassischen Algorithmen meistens etwas bessere Werte bei *Purity* und NMI erzielen, so ist der Vorteil des SBCA die Interpretierbarkeit der Ergebnisse. Eine weitere Stärke ist die selbständige Suche nach der Anzahl der Cluster. Anders als beim K-Means wird kein zusätzlicher Parameter benötigt, der dem SBCA die Anzahl der Cluster vorgibt. Der verwendete EM-Algorithmus findet die Anzahl der Cluster mittels Cross-validation heraus. Dies ist bei dem SBCA ebenfalls nicht nötig.
Auch im Vergleich mit der State-of-the-Art Methode FRBC für interpretierbares Clustering erzielen die Ergebnisse des SBCA ähnlich gute Wertungen. Die Anzahl der Datensätze, auf denen der SBCA höhere und niedrigere Wertungen als der FRBC erzielt, ist etwa gleichgroß.
Ein Nachteil des SBCA ist jedoch, dass Datensätze existieren, auf denen eine viel zu hohe Anzahl an Clustern gefunden wird. Auch wenn hierdurch alle Instanzen dieser Cluster meist der gleichen Klasse angehören und somit eine hohe *Purity* erreicht wird, sind diese Ergebnisse nur schwer zu interpretieren.

6 Related Work

Dieses Kapitel handelt von Verfahren, die eine ähnliche Zielsetzung haben wie der Separate-and-Conquer based Clustering Algorithm. Neben einer sinnvollen Gruppierung der Daten sollen die dadurch gefundenen Cluster für einen menschlichen Anwender interpretierbar sein. Die nachfolgenden Verfahren entstanden in den Jahren von 2005 bis heute und verfolgen dabei unterschiedliche Ansätze. In den Jahren davor erschienen keine relevanten Verfahren, bei denen der Fokus auf Interpretierbarkeit der Ergebnisse lag.
Basak und Krishnapuram in [4] oder Fraiman, Ghattas und Svarc in [10] setzen bei ihren Lösungsideen auf das Erzeugen von Entscheidungsbäumen. Alternativen bieten Mansoori in [23] oder Williams, Soares und Gilbert in [33]. Diese bedienen sich an Methoden des überwachten Lernens, indem sie zuerst die Daten manipulieren, um anschließend einen regelbasierten Klassifikator verwenden zu können.
Zusätzlich zu den unterschiedlichen Vorgehensweisen, wird in diesem Abschnitt auch auf die grundlegenden Unterschiede zu dem in Kapitel 4 entworfenen SBCA eingegangen.

6.1 Clustering Rule-based Algorithm

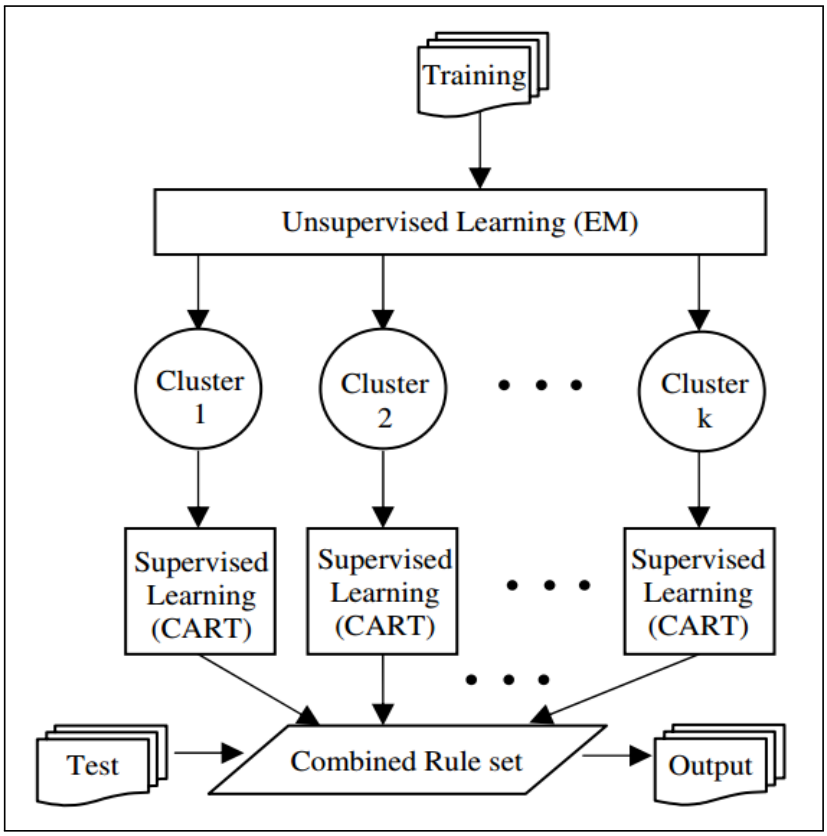


Abbildung 10: CRA Flussdiagramm
Quelle: [33], Abbildung 1

Der Clustering Rule-based Algorithmus (CRA) [33] ist ein Verfahren, welches Cluster in einem Datensatz findet und diese anschließend mit Regeln beschreibt. Hierbei versucht der CRA nicht direkt durch Regeln die einzelnen Cluster zu finden, sondern kombiniert zwei unterschiedliche Problemstellungen. Diese sind das Finden von Clustern auf unüberwachten Datenmengen und das Lernen von Klassifikationsregeln auf einer überwachten Datenmenge. Ersteres wird durch einen üblichen Clustering-Algorithmus gelöst. Anschließend werden den unüberwachten Daten, mithilfe der Ergebnisse des verwendeten Clustering-Algorithmus, Klassenwerte zugewiesen. Diese dadurch entstehenden überwachten Daten werden nachfolgend von einem regelbasierten Klassifikator verwendet, um Regeln zu erzeugen.

Der CRA verwendet zuerst einen Expectation-Maximization-Algorithmus (EM-Algorithmus), um die gegebenen Trainingsdaten in Cluster zu unterteilen. Hierbei wird jede Instanz genau einem Cluster zugeteilt. Nachdem die einzelnen Datenpunkte in unterschiedliche Cluster unterteilt wurden, wird der Classification and Regression Trees-Algorithmus (CART) [6] zum Lernen von Regeln verwendet. Dieser erzeugt binäre Klassifikations- und Regressionsbäume, welche der Klassifikation von Datenobjekten dienen. Der Clustering Rule-based Algorithmus verwendet jedes durch den EM-Algorithmus gefundene Cluster einzeln, um einen separaten CART zu trainieren.
Klassifikations- und Regressionsbäume sind Entscheidungsbäume, welche zur Darstellung von Entscheidungsregeln verwendet werden. Ein Entscheidungsbaum besteht aus beliebig vielen inneren Knoten, aus mindestens zwei Blattknoten und aus genau einem Wurzelknoten. Diese sind absteigend von dem Wurzelknoten bis zu den Blättern durch Zweige verbunden. Hierbei stellt jeder komplette Pfad des Baumes eine Regel dar und die Blätter liefern die Antwort auf das Entscheidungsproblem. Die einzelnen Knoten entsprechen den Bedingungen der Regeln.
Um die Attribute der einzelnen Knoten auszuwählen, bestimmt der CART-Algorithmus den Informationsgehalt der vorhandenen Attribute. Die Knotenattribute werden so gewählt, dass der Informationsgehalt maximiert wird.
Jeder CART stellt eine Regelmenge dar. Diese Regelmengen werden anschließend zu einer einzigen kombinierten Regelmenge zusammengefügt. Die Regeln dieser Menge stellen die gefundenen Cluster dar.

Das Ziel und die Ausgaben des CRA und des SBCA sind sehr ähnlich. Beide Algorithmen suchen Cluster in einer Datenmenge und geben diese in Form einer Regelmenge aus. Die Vorgehensweise des CRA unterscheidet sich jedoch grundlegend von der des SBCA: Während der SBCA durch Verfeinerung einzelner Regeln die Cluster findet und abgedeckte Daten aus der Datenmenge entfernt, löst der CRA zuerst ein unüberwachtes und anschließend ein überwachtes Problem in der gesamten Datenmenge.

6.2 Fuzzy Rule-Based Clustering Algorithm

Ein weiterer Algorithmus, welcher Regeln zur Beschreibung interpretierbarer Cluster verwendet, ist der von Eghbal G. Mansoori entwickelte Fuzzy Rule-Based Clustering Algorithm (FRBC) [23]. Wie auch bei dem CRA werden bei dem FRBC die unüberwachten Eingabedaten so weit gehend bearbeitet, bis das ursprüngliche Clustering-Problem als überwachtes Klassifikationsproblem gelöst werden kann. Dies geschieht durch das Hinzufügen von Hilfsdatenpunkten. Anschließend wird ein Klassifikator zur Lösung des Problems verwendet.
Um die einzelnen Cluster in den Daten zu finden, betrachtet FRBC alle unbezeichneten Eingabedaten als Primärdaten. Diese werden für die nachfolgenden Schritte als Klasse 1 bezeichnet. Als nächstes werden gleichmäßig verteilte Datenpunkte als Hilfsdaten generiert. Diese werden zufällig erzeugt und mit der Klassenbezeichnung 2 zu den Primärdaten hinzugefügt. Die Anzahl der generierten Hilfsdaten ist Abhängig von der Anzahl der Primärdaten und deren Verteilung.
Das durch die vorhergehenden Schritte entstandene Zwei-Klassen-Problem bildet die Basis für das Finden der Cluster unter Verwendung eines Klassifikators. FRBC benutzt den Regellerner Steady-State Genetic Algorithm for Extracting Fuzzy Classification Rules From Data (SGERD) [24], um dieses Zwei-Klassen-Problem zu lösen und Klassifikationsregeln zu erhalten. SGERD ist ein Algorithmus für das generieren von Fuzzy-Klassifizierungsregeln. Zur Auswahl der Regeln benutzt SGERD mehrere Regel- und Datenabhängige Parameter. Allgemein generiert SGERD für ein n-dimensionales Problem mit M Klassen eine vordefinierte Anzahl Q an Regeln für jede Klasse. Insgesamt liefert SGERD somit $M \times Q$ Regeln. Die erzeugten Regeln sind nach einer internen Bewertung absteigend sortiert. In FRBC wird nur die Erste, und somit beste Regel, der Klasse 1 verwendet. Jede auf diese Weise gelernte Regel repräsentiert ein gefundenes Cluster.
Die von einer solchen Regel abgedeckten Beispiele sind die initialen Elemente der Cluster und werden von den Primärdaten entfernt. Ähnlich wie bei dem Separate-and-Conquer basierten SBCA werden auch bei dem Fuzzy Rule-Based Clustering Algorithmus die abgedeckten Instanzen für das Finden weiterer

Cluster nicht mehr berücksichtigt.
Diese Schritte zum Regellernen werden so lange wiederholt, bis in den verbleibenden Primärdaten keine aussagekräftigen Cluster vorhanden sind. Diese Situation bildet das Abbruchkriterium und wird durch eine Regel-Effektivität-Messung erkannt. Regeln, die ein aussagekräftiges Cluster beschreiben, besitzen eine hohe Effektivität. Eine starke Abnahme der Regel-Effektivität einer neuen Regel im Vergleich zu den zuvor gelernten Regeln, ist die Abbruchbedingung des FRBC.
Um die Grenzen der einzelnen Cluster zu bestimmen, werden die Primärdaten mithilfe der erzeugten Regeln klassifiziert. Die Abbruchbedingung kann dazu führen, dass einige Instanzen von keiner der Regeln abgedeckt werden. Um diese Daten einzuschließen, werden die Zentren der gefundenen Cluster bestimmt. Anschließend wird jede nicht abgedeckte Instanz dem Cluster, das die geringste Entfernung vom Clusterzentrum zur Instanz besitzt, zugewiesen.

Die Vorgehensweise des FRBC ist der des SBCA sehr ähnlich. Sowohl der SBCA als auch der FRBC finden die einzelnen Cluster nacheinander und entfernen dabei abgedeckte Instanzen aus der Datenmenge. Der Unterschied liegt in der Art des Regellernens. Durch das Hinzufügen von Hilfsdaten ist der FRBC in der Lage, einen Regellerner für Klassifizierungsaufgaben zu benutzen. Dies entspricht dem Lösen eines überwachten Problems. Der SBCA hingegen behandelt zu keinem Zeitpunkt ein überwachtes Problem, sondern lernt die Regeln auf den unbearbeiteten Eingabedaten.

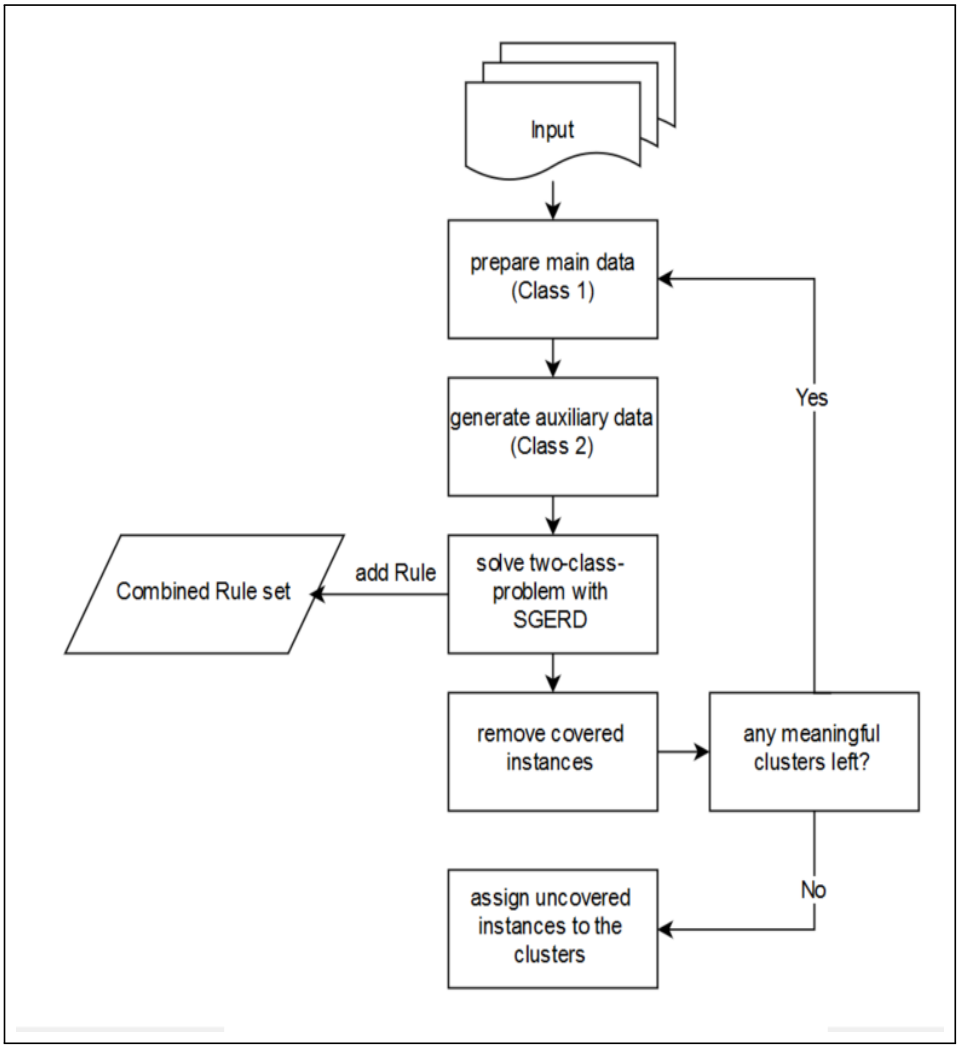


Abbildung 11: FRBC Flussdiagramm

6.3 Interpretable Clustering of Numerical and Categorical Objects

Der von Claudia Plant und Christian Böhm veröffentlichte Algorithmus Interpretable Clustering of Numerical and Categorical Objects (INCONCO) [29] wurde entworfen, um interpretierbare Cluster zu finden. Hierzu wird die erweiterte Cholesky Zerlegung für eine zielgerichtete Suche nach interpretierbaren Clustern verwendet. Der Algorithmus arbeitet auf numerischen und kategorischen Daten und ist ein partitionierendes Top-Down Clusterverfahren.
INCONCO beginnt mit einem einzigen Cluster, welches alle Instanzen der Datenmenge beinhaltet. Anschließend wird dieses Cluster durch k-INCONCO mit k = 2 in zwei Clustern aufgeteilt.
Der iterative Algorithmus k-INCONCO ist Grundbaustein von INCONCO. k-INCONCO beginnt mit der Initialisierung der k Cluster mittels zufälligen Daten aus der Datenmenge. Anschließend folgen zwei weitere, sich wiederholende Schritte.
Im ersten Schritt werden die Instanzen den einzelnen Clustern zugewiesen. Im nächsten Schritt wird das daraus folgende Modell berechnet. Bei der Modell-Berechnung werden die einzelnen Attribute zu sogenannten Super-Attributen zusammengefasst. Ein Super-Attribut ist ein Attribut, welches aus einem oder mehreren ursprünglichen Attributen zusammengesetzt ist. Das Bilden dieser Super-Attribute basiert auf der Minimum Description Length (MDL). Eine ausführliche Beschreibung der Initialisierung, der Modell-Berechnung und der Erzeugung von Super-Attributen mithilfe der MDL ist in [29] zu finden.
k-INCONCO wird wiederholt mit dem Parameter k = 2 ausgeführt. Iterativ werden hierdurch alle folgenden Cluster solange aufgeteilt, bis die Abbruchbedingung $MDL(C_L) + MDL(C_R) \geq MDL(C)$ den gesamtem Algorithmus beendet. Diese ist erfüllt, wenn die MDL(C) eines Clusters C kleiner ist, als die Summe aus MDL(C_L) und MDL(C_R). Hierbei sind C_L und C_R die Aufteilungen des Clusters C.
Ein Nachteil des Algorithmus ist der Zufallsfaktor bei der Initialisierung. Da bei der Initialisierung des k-INCONCO zufällige Daten aus der Datenmenge verwendet werden, muss der Algorithmus mehrere Male ausgeführt werden, um optimale Ergebnisse zu erzielen. Dabei müssen sich die Initialisierungen bei jeder Ausführung unterscheiden. Anschließend können die besten Ergebnisse im Bezug auf die MDL verwendet werden.

Im Gegensatz zu CRA und FRBC wird bei INCONCO kein überwachtes Klassifikationsproblem gelöst, um Regeln zu generieren. Der größte Unterschied zu SBCA liegt an der Vorgehensweise. Während bei SBCA die Cluster nacheinander gefunden und abgedeckte Instanzen entfernt werden, behandelt INCONCO alle Cluster simultan und spaltet lediglich vorhandene Cluster in zwei neue Cluster auf.
Ein weiterer Unterschied ist der vorhandene Zufallsfaktor bei INCONCO. Dieser liefert bei mehreren Durchläufen mit den gleichen Eingabedaten unterschiedliche Ergebnisse. SBCA hingegen erzeugt bei jedem Durchlauf die gleichen Regeln.

6.4 Interpretable Hierarchical Clustering by Constructing an Unsupervised Decision Tree

Basak und Krishnapuram erhalten in [4] interpretierbare Ergebnisse einer Clusteranalyse durch Erzeugen eines unüberwachten Entscheidungsbaumes. Der generierte Entscheidungsbaum lässt sich in Form von Regeln interpretieren. Jeder Blattknoten repräsentiert ein eindeutiges Cluster. Die einzelnen Wege von der Wurzel bis hin zu den Blättern stellen die verschiedenen Regeln dar, die zu den Clustern führen. Die Generierung des Entscheidungsbaumes erfolgt durch schrittweises Clustering. Dabei werden die Daten bei jedem Entscheidungsknoten geteilt bis der Algorithmus terminiert. Jeder dieser Knoten besitzt ein Knotenattribut. Diese müssen vom Algorithmus gewählt werden.
In ihrer Arbeit stellen Basak und Krishnapuram vier verschiedene Maße für die Wahl der Knotenattribute vor. Diese basieren alle auf der Inhomogenität der Datenmenge und versuchen diese zu maximieren.
Nachdem ein entscheidendes Attribut selektiert wurde, müssen die Daten abhängig von diesem geteilt werden. Hierzu werden zwei Alternativen geliefert. Während die erste Möglichkeit durch binäre Teilungen versucht, die Änderung des Informationsgehaltes zu maximieren, beruht die andere auf einer Valley Detection. Hierbei werden bei kategorischen Daten alle im Datensatz vorkommende Attributwerte des

Knotenattributes als Split-Punkt verwendet. Bei numerischen Attributen wird zuerst das Histogramm der Daten entlang der Attribut-Dimension berechnet. Anschließend werden die dort auftretenden Valley-Punkte evaluiert. Die besten k-1 Valley-Punkte werden als Split-Punkte verwendet. Bei dieser Methode können Entscheidungsbäume mit mehreren Verzweigungen entstehen.
Damit der Algorithmus terminiert, wurde eine untere Grenze für Anzahl von Datenpunkte an jedem Knoten festgelegt. Es ist außerdem möglich, diese Bedingung mit der Tiefe des Entscheidungsbaumes oder der Minimum Description Length zu kombinieren, um präzisere Abbruchbedingungen zu erhalten.

Da dieser Ansatz darauf beruht, einen Entscheidungsbaum zu generieren, unterscheidet er sich sowohl in der Vorgehensweise als auch in den Resultaten deutlich von den bereits vorgestellten Algorithmen und dem SBCA. Während bei dem SBCA die Regeln einzeln gelernt werden, werden in [4] die Regeln aus einem Entscheidungsbaum abgeleitet.

6.5 Interpretable Clustering using Unsupervised Binary Trees

In [10] wird die hierarchische Top-down Methode Clustering using unsupervised binary trees (CUBT) vorgestellt, welche ebenfalls auf Entscheidungsbäume zurückgreift. Bei ihr werden jedoch nur binäre Bäume durch einen Prozess mit drei Phasen erzeugt. Diese Phasen enthalten einen Vorwärtsschritt (Erzeugung eines maximalen Entscheidungsbaumes) und zwei Rückwärts-schritte (Pruning und Joining).
CUBT beginnt mit dem Vorwärtsschritt, bei dem zuerst die gesamte Datenmenge dem Wurzelknoten zugewiesen wird. Folgend wird diese Menge in zwei Untermengen unterteilt. Dabei wird die Aufteilung so gewählt, dass sich die Instanzen in der gleichen Menge möglichst ähnlich sind. Dies wird anhand einer Zielfunktion ermittelt, welche ein Maß für die Heterogenität darstellt.
Das Aufteilen einer Menge wird rekursiv auf jede Untermenge angewandt bis ein Abbruchkriterium erfüllt ist. Dieses ist abhängig von einem Parameter, den der Anwender setzen muss, und bezieht sich auf die Mindestgröße eines Knotens. Kleine Werte des Parameters führen dabei zu einem tiefen Baum mit vielen Blattknoten.
Am Ende des Vorwärtsschrittes ist der entstandene Baum der maximale Entscheidungsbaum. Jeder Blattknoten entspricht dabei einem Cluster und bekommt eine eindeutige Bezeichnung. Für den Fall, dass der Baum bereits genau so viele Blattknoten besitzt wie wirkliche Cluster existieren, ist es nicht notwendig, die folgenden Schritte durchzuführen. Andernfalls werden nun die beiden Rückwärtsschritte durchgeführt.
In der zweiten Phase, dem *Minimum Dissimilarity Pruning*, werden ähnliche Knoten in dem Vorgängerknoten zusammengefasst. Dabei werden immer zwei Knoten mit dem gleichen Vorgängerknoten untersucht. Haben die Datenmengen dieser beiden Knoten nicht den Mindestwert an geforderter Ungleichheit (Dissimilarity), so werden diese in dem darüberliegenden Knoten vereint.
Die dritte Phase, das Joining, ist der Pruning Phase sehr ähnlich. Der Unterschied liegt an den Knoten, die untersucht werden. Während die Pruning-Phase nur jeweils zwei benachbarte Geschwisterknoten betrachtet, werden in dieser Phase alle Knoten-Paare untersucht, die unterschiedliche Vorgänger besitzen. Die Bedingung für das Zusammenführen ist hierbei die gleiche wie bei dem *Minimum Dissimilarity Pruning*.
Nach Ablauf dieser drei Phasen liegt ein Entscheidungsbaum vor, der durch zwei Rückwärtsschritte gekürzt wurde. Auch hier bilden die einzelnen Wege zu den Blättern die Regeln der jeweiligen Cluster.

CUBT lernt einen Entscheidungsbaum in drei Schritten. In diesen wird zuerst ein Baum erzeugt und anschließend gekürzt. Diese Schritte unterscheiden den CUBT von anderen Algorithmen zur Erzeugung interpretierbarer Cluster, da diese Schritte weder bei dem Separate-and-Conquer based Clustering Algorithm, noch bei einem der anderen in diesem Kapitel vorgestellten Algorithmen in irgendeiner Form vorhanden sind.

6.6 Learning Predictive Clustering Rules

Predictive Clustering (PC) ist eine Kombination aus Predictive Modelling und Clustering. Beide gehören zu den häufigsten Aufgaben im Bereich des Datamining. Bei Predictive Clustering werden Elemente aus beiden Gebieten zu einer Erweiterung generalisiert. Dieses Vorgehen und das dazugehörige Framework Preductive Clustering Rules (PCRs) wird von Zenko, Dzeroski und Struyf in [35] beschrieben.
Wie bei der Clusteranalyse üblich, werden auch hier Beispiele gesucht, die sich möglichst ähnlich sind. Diese werden in Cluster zusammengefasst. Beispiele unterschiedlicher Cluster sollen sich hierbei möglichst unähnlich sein. Der Unterschied zum normalen Clustering liegt in den Attributen, die bei der Cluster-Suche berücksichtigt werden. Während die Clusteranalyse ein unüberwachtes Problem löst, besitzen die einzelnen Eingabedaten beim Predictive Clustering einen Zielwert (z.B Klasse). Dieser kann für das Finden von Clustern mitberücksichtigt werden. Ob und wie sehr der Zielwert für das Clustering verwendet wird, lässt sich in der verwendeten Distanzmetrik mithilfe eines Parameters einstellen. Setzt man diesen Parameter beispielsweise auf Null, wird wie beim Clustering üblich ein unüberwachtes Problem gelöst.
Nachdem die Cluster gefunden wurden, wird zusätzlich für jedes Cluster ein Predictive Modell erzeugt. Dieses liefert abhängig von den Attributen der Beispiele, die dem gleichen Cluster angehören, eine Vorhersage über den Zielwert.
Bei PC in [35] wird jedes gefundene Cluster durch eine Regel beschrieben. Dieses hat die Form *Wenn Cluster-Beschreibung DANN Cluster-Prototyp*. Die Cluster-Beschreibung hat hierbei die gleiche Form wie der Body bei den im SBCA verwendeten Regeln. Der Prototyp ist das Predictive Modell, welches die Zielwerte der Cluster vorhersagt. Damit eine Regel auch auf ungesehene Instanzen ohne Zielwerte angewandt werden kann, wird bei der Cluster-Beschreibung der Zielwert nicht verwendet.
Das System für das Lernen der PCRs basiert auf dem bekannten CN2 Algorithmus [7]. Dieser erzeugt iterativ neue Regeln. Instanzen, die von einer Regel abgedeckt werden, werden aus der Trainingsmenge entfernt. Dies wird wiederholt bis die Datenmenge leer ist oder keine neuen Regeln gefunden werden. Die Heuristik, die für das Lernen von PCRs verwendet wird, ist ein Maß für die Kompaktheit und misst die durchschnittliche Distanz eines Beispieles, das von einer Regel abgedeckt wird, zu dem Prototyp dieser Beispielmenge.

Die Eingabedaten für PC können auch Zielwerte enthalten, die für die Clusterfindung verwendet werden. Zusätzlich wird für jedes Cluster ein Predictive Modell erzeugt. Dieses liefert abhängig von den Attributen der Beispiele, die dem gleichen Cluster angehören, eine Vorhersage über den Zielwert. Die Verwendung der Zielwerte für die Clusterfindung bildet zusammen mit den für die Cluster erzeugten Predictive Modells den wesentlichen Unterschied zwischen PC und SBCA. Durch die Kombination von Predictive Modelling und Clustering bietet PC somit eine einzigartige Möglichkeit, interpretierbare Cluster zu erhalten.

7 Diskussion und Ausblick

In dieser Arbeit wurde der neuartige Algorithmus SBCA für interpretierbares Clustering entwickelt. Der SBCA findet automatisch potentiell vorhandene Cluster in einer Datenmenge. Er verwendet hierzu einen Separate-and-Conquer Ansatz, um die Strukturen der Daten zu erforschen. Die gefundenen Cluster werden in Form von Regeln beschrieben, die in einer geordneten Regelmenge zusammengefasst werden.
Der SBCA erzielt auf fast allen verwendeten Datensätze gute Ergebnisse, die mit anderen Algorithmen vergleichbar sind. Ausnahmen bilden die Datensätze *Vehicle*, *Wine* und *Vowel*, da die Anzahl der gefundenen Cluster unangemessen hoch ist.
Im Vergleich mit der State-of-the-Art Methode FRBC für interpretierbares Clustering erzielen die Ergebnisse des SBCA ähnlich gute Wertungen. Trotz der für die NMI und *Purity* suboptimalen Parameter, die in den Experimenten verwendet wurde, erzielt der SBCA auf den Datensätzen *Ecoli*, *Glass*, *Vowel*, *Vehicle* und *Sonar* eine etwas höhere NMI als der FRBC. Eine bessere *Purity* erzielt der SBCA auf allen Datensätzen außer *Iris* und *Thryoid*. Vergleicht man den SBCA mit dem K-Means und dem EM-Algorithmus, so schneidet der SBCA etwas schlechter ab. Auf den Datensätzen *Iris*, *Ecoli*, *Glass*, *Sonar* und *Vehicle* befinden sich die *Purity* und die NMI der Ergebnisse des SBCA, des K-Means und des EM-Algorithmus etwa im gleichen Bereich. Jedoch erzielt der SBCA in den meisten Fällen eine etwas geringere NMI und *Purity*. Auch wenn der SBCA auf manchen Datensätzen eine unangemessen hohe Anzahl an Cluster findet oder die NMI und *Purity* herkömmlicher Algorithmen etwas besser ausfallen, so bietet der SBCA andere Vorteile, die ihm zu einem bedeutungsvollen Algorithmus machen:
Der wohl wichtigste Vorteil ist die einfache Interpretierbarkeit der Ergebnisse, die in Tabelle 7 vorgestellt wurden. Während bei einer mittelpunktbasierten Darstellung für eine Zuordnung einer neuen Instanz die Distanzen zu den Mittelpunkten der Clustern berechnet werden muss, genügt es bei der regelbasierten Darstellung, die der SBCA verwendet, die Regeln der Reihe nach auszuführen. Dies macht den SBCA in der Datenanalyse zu einem nützlichen Algorithmus für interpretierbares Clustering. Da es das Hauptziel der Datenanalyse ist, verwendbares Wissen über Daten zu erhalten, sind schlecht interpretierbare Ergebnisse für einen menschlichen Anwender kaum nutzbar, da dieser kein neues Wissen aus den erhaltenen Ergebnissen gewinnen kann. Mit zunehmender Interpretierbarkeit nimmt das Verständnis über die Zusammensetzung der einzelnen Cluster zu. Es wird ersichtlich, weshalb eine bestimmte Menge von Instanzen ausgewählt wurde, ein Cluster zu bilden, und wie sie sich von Objekten anderer Cluster unterscheiden.
Im Gegensatz zu vielen anderen Clustering-Algorithmen findet der SBCA selbständig die Anzahl der Regeln und somit die Cluster ohne einen Parameter, der die Anzahl der vorhandenen Cluster angibt. Die zusätzlich gelieferte generische Form des SBCA bietet viele weitere Möglichkeiten, den Algorithmus in Hinblick auf Suchrichtung, Suchalgorithmus, Abbruchbedingungen und Heuristiken zu verändern und zu optimieren. Dies ermöglicht es auch, eine Entscheidungsliste zu erzeugen, bei der die Default-Regel aussagt, dass die verbleibenden Instanzen kein Cluster bilden. So können beispielsweise Ausreißer zusammengefasst werden. Eine solche Regelmenge ist in Tabelle 3 dargestellt.

Die Tatsache, dass der SBCA bei manchen Datensätzen schlechte Ergebnisse liefert, fordert ein Weiterentwickeln und Erforschen des Algorithmus. Neben einer ausgiebigen Evaluation auf einigen weiteren Datensätzen kann es für die Experimente sinnvoll sein, weitere Kriterien für das Beurteilen der Ergebnisse hinzuzunehmen. Dies ermöglicht präzisere Aussagen über die Qualität der Ergebnisse und gibt somit neue Anhaltspunkte für verbesserte Heuristiken und Abbruchbedingungen.
Bei der Erforschung weiterer Heuristiken und Abbruchbedingungen kann es nützlich sein, auch auf Distanzen und Merkmale bereits entfernter Instanzen zurückzugreifen. Somit ist es möglich, bei der Beurteilung die Eigenschaften und Strukturen des gesamten Datensatzes zu betrachten.
Zuletzt ist noch zu erwähnen, dass die Laufzeit auf manchen Datensätzen mehrere Stunden betrug. Der Grund hierfür liegt in der Bestimmung aller möglichen Split-Punkte und in der wiederholten Berechnung der einzelnen Distanzen zwischen den Datenpunkten. Daher ist es nötig, die bisher verwendeten

Bestandteile und die Implementation des SBCA auf Effizienz zu überprüfen und diese gegebenenfalls zu verbessern. Nach einer Optimierung sollte die Laufzeit des SBCA für jeden der hier verwendeten Datensätze weniger als eine Stunde betragen.

Literatur

[1] D. N. A. Asuncion. UCI machine learning repository, 2007.

[2] M. Anderberg. *Cluster analysis for applications*. Probability and mathematical statistics. Academic Press, 1973.

[3] O. Arbelaitz, I. Gurrutxaga, J. Muguerza, J. M. Pérez, and I. Perona. An extensive comparative study of cluster validity indices. *Pattern Recognition*, 46(1):243 – 256, 2013.

[4] J. Basak and R. Krishnapuram. Interpretable hierarchical clustering by constructing an unsupervised decision tree. *Knowledge and Data Engineering, IEEE Transactions on*, 17(1):121–132, Jan 2005.

[5] J. Bezdek and N. Pal. Some new indexes of cluster validity. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 28(3):301–315, Jun 1998.

[6] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth International Group, Belmont, CA, 1984.

[7] P. Clark and T. Niblett. The cn2 induction algorithm. *Mach. Learn.*, 3(4):261–283, Mar. 1989.

[8] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistcal Society*, 39(B):1–38, 1977.

[9] D. H. Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139–172, 1987.

[10] R. Fraiman, B. Ghattas, and M. Svarc. Interpretable clustering using unsupervised binary trees. *Adv. Data Analysis and Classification*, 7(2):125–145, 2013.

[11] J. Fürnkranz, D. Gamberger, and N. Lavrac. *Foundations of Rule Learning*. Cognitive Technologies. Springer, 2012.

[12] J. Fürnkranz. Separate-and-conquer rule learning. *Artif. Intell. Rev.*, 13(1):3–54, Feb. 1999.

[13] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: An update. *SIGKDD Explor. Newsl.*, 11(1):10–18, Nov. 2009.

[14] J. Handl, J. D. Knowles, and D. B. Kell. Computational cluster validation in post-genomic data analysis. *Bioinformatics*, 21(15):3201–3212, 2005.

[15] J. A. Hartigan. *Clustering Algorithms*. John Wiley & Sons, Inc., New York, NY, USA, 99th edition, 1975.

[16] A. K. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1988.

[17] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, Sept. 1999.

[18] F. Janssen and J. Fürnkranz. Separate-and-conquer regression. In M. Atzmüller, D. Benz, A. Hotho, and G. Stumme, editors, *Proceedings of the German Workshop on Lernen, Wissen, Adaptivität - LWA2010*, pages 81–89, 2010.

[19] F. Janssen and M. Zopf. The seco-framework for rule learning. In *Proceedings of the German Workshop on Lernen, Wissen, Adaptivität - LWA2012*, 2012.

[20] R. Kohavi and F. Provost. Glossary of terms. *Machine Learning*, 30(2-3):271–274, February 1998.

[21] R. Kruse, G. D. Riccia, and H.-J. Lenz, editors. *Learning, Networks and Statistics*. Springer, 1997.

[22] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[23] E. Mansoori. Frbc: A fuzzy rule-based clustering algorithm. *Fuzzy Systems, IEEE Transactions on*, 19(5):960–971, Oct 2011.

[24] E. Mansoori, M. Zolghadri, and S. Katebi. Sgerd: A steady-state genetic algorithm for extracting fuzzy classification rules from data. *Fuzzy Systems, IEEE Transactions on*, 16(4):1061–1071, Aug 2008.

[25] R. S. Michalsk. On the quasi-minimal solution of the covering problem. In *Proceedings of the 5th International Symposium on Information Processing (FCIP-69)*, volume A3, page 125–128, 1969.

[26] R. S. Michalski and R. E. Stepp. Automated construction of classifications: Conceptual clustering versus numerical taxonomy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-5(4):396–410, July 1983.

[27] K. Oehler and R. Gray. Combining image classification and image compression using vector quantization. In *Data Compression Conference, 1993. DCC '93.*, pages 2–11, 1993.

[28] G. Pagallo and D. Haussler. Boolean feature discovery in empirical learning. *Mach. Learn.*, 5(1):71–99, May 1990.

[29] C. Plant and C. Böhm. Inconco: interpretable clustering of numerical and categorical objects. In C. Apté, J. Ghosh, and P. Smyth, editors, *KDD*, pages 1127–1135. ACM, 2011.

[30] E. Rasmussen. Information retrieval. chapter Clustering Algorithms, pages 419–442. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1992.

[31] P. Sneath and R. Sokal. *Numerical Taxonomy. The Principles and Practice of Numerical Classification*. Freeman, 1973.

[32] T. Taxt, P. J. Flynn, and A. K. Jain. Segmentation of document images. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(12):1322–1329, 1989.

[33] P. Williams, C. Soares, and J. E. Gilbert. A clustering rule-based approach to predictive modeling. In H. C. Cunningham, P. Ruth, and N. A. Kraft, editors, *ACM Southeast Regional Conference*, page 45. ACM, 2010.

[34] P. Williams, C. Soares, and J. E. Gilbert. A clustering rule-based approach to predictive modeling. In *Proceedings of the 48th Annual Southeast Regional Conference*, ACM SE '10, pages 45:1–45:5, New York, NY, USA, 2010. ACM.

[35] B. Zenko. Learning predictive clustering rules. *Informatica (Slovenia)*, 32(1):95–96, 2008.