Universität Konstanz

Fachbereich für Informatik und Informationswissenschaft
Nycomed Stiftungs-Lehrstuhl für Angewandte Informatik
Bioinformatik und Information Mining

Dissertation

Lernen hierarchischer Fuzzy-Regelmodelle

zur Erlangung des akademischen Grades des
Doktors der Naturwissenschaften (Dr. rer. nat.)

vorgelegt von:

Thomas Gabriel

Tag der mündlichen Prüfung: 1. Juni 2010

Prüfer:
Prof. Dr. Michael R. Berthold
Prof. Dr. Marcel Waldvogel
Prof. Dr. Rudolf Kruse (Universität Magdeburg)

Thomas Gabriel
Lernen hierarchischer Fuzzy-Regelmodelle
Universität Konstanz
1. Juni 2010

Kurzfassung

In unserer modernen Gesellschaft und der ständig wachsenden Flut von Informationen kommen in vielen Bereichen immer häufiger Verfahren des Data Minings zum Einsatz. Oft sind die Daten unstrukturiert und nicht überschaubar, sodass es schwer ist, die interessanten und relevanten Informationen zu extrahieren. Data Mining Methoden helfen, Zusammenhänge in großen Datenmengen zu finden und für den Menschen verwendbar zu machen. Damit neues Wissen interpretiert werden kann, ist es wichtig, verständliche und am besten in einfache Regeln gefasste Modelle zu erzeugen.

Diese Forschungsarbeit beschäftigt sich mit Verfahren aus dem Bereich der intelligenten Datenanalyse, die speziell Muster in Form von Regeln automatisch aus Daten lernen. Die regelbasierten Systeme kommen gerade dann zum Einsatz, wenn Abhängigkeiten in Daten für den Menschen verständlich beschrieben werden sollen. Die Regeln kommen aufgrund ihrer einfachen Struktur dem menschlichen Handeln und Denken recht nah und können deswegen von einem Anwender direkt verstanden und interpretiert werden. Es wird ein Verfahren zum automatischen Lernen von Fuzzy-Regelbasen erweitert und untersucht, welchen Einfluss verschiedene Parameter auf die Generalisierungsleistung des gelernten Fuzzy-Regelmodells haben. Solche Modelle von Regeln sind zwar allgemein verständlich, leiden aber darunter, dass gewöhnlich viele Regeln erzeugt werden müssen, um alle Details der zugrunde liegenden Daten zu modellieren.

Den Fokus bilden in dieser Dissertation sogenannte hierarchische, regelbasierte Vorhersagemodelle. Solche Hierarchien von Modellen bestehen aus verschiedenen Schichten von einfachen Regelmodellen, die das Konzept des Ursprungs der Daten in jedem Niveau zu einem bestimmten Detailgrad beschreiben. In den oberen Schichten erklären nur wenige Regeln grob das Konzept der Daten. Dagegen konzentrieren sich Regeln weiter unten in der Hierarchie auf Details, aber auch auf Artefakte und Ausreißer. Eine Hierarchie von Regelmodellen wird durch einen lokalen Lerner in-

duziert und erklärt über die Ebenen der Hierarchie die Daten vollständig. Durch die Verwendung eines einfachen Lernverfahrens mit verständlicher Hypothesensprache bleibt die Hierarchie auch im Falle von komplexen Zusammenhängen in den Daten interpretierbar. Diese hierarchischen Modelle helfen bei der Exploration von großen Datenbeständen und können insbesondere auch für die Klassifikation unbekannter Daten verwendet werden.

Die abschließende Evaluierung des hierarchischen Ansatzes bezüglich Explorations- und Klassifikationsfähigkeit auf ausgewählten Benchmark-Datensätzen sowie auf praxisrelevanten Daten sollen unsere Annahmen bestätigen. Zum einen werden zum Teil signifikant kleinere Regelmodelle erzeugt, zum anderen verbessert sich die Klassifikationsleistung der Einzelmodelle mit gleichzeitig weniger Regeln. Die Experimente auf dem hierarchischen Lerner werden vergleichend zum klassischen Fuzzy-Regellerner durchgeführt, dessen Leistungsfähigkeit zuvor mit anderen anerkannten Datenanalyseverfahren unter Verwendung verschiedener Parameter evaluiert wird.

In dieser Arbeit wird ein Ansatz für die Erweiterung eines regelbasierten Lernverfahrens vorgestellt, der hierarchisch-strukturierte Regelmodelle erzeugt. Die schichtenartigen Modelle sollen auf der einen Seite verständlich und interpretierbar sein, auf der anderen Seite aber auch gute Klassifikationseigenschaften besitzen. Basierend auf einem lokalen Lernverfahren für das konstruktive Training von Fuzzy-Regeln wird gezeigt, wie sich interpretierbare Fuzzy-Regelhierarchien generieren lassen. Das erzeugte Fuzzy-Regelmodell generalisiert und bleibt damit auch im Falle komplexer Zusammenhänge verständlich; ist aber beliebig speziell, da alle Details der Daten über die verschiedenen Schichten der Hierarchie erklärt werden.

Eine Hierarchie von graduellen Modellen zerlegt ein ansonsten komplexes Modell in unterschiedliche Abstraktionsebenen, die jeweils ein einfaches und verständliches Regelmodell beinhalten. Das Gesamtmodell eröffnet damit Möglichkeiten für die interaktive Exploration von Modell und Daten auf den unterschiedlichen Niveaus in der Hierarchie.

Abstract

In our modern society and due to the ever-growing flood of information data mining methods are increasingly applied in many areas. However, often the data is unstructured and not manageable which makes it difficult to extract interesting and relevant information. Data mining methods help find coherences in large data sets and make them useful for humans. In order to interpret new knowledge, it is important to generate understandable and simple rules captured in easy models.

This research work deals with methods from the field of intelligent data analysis that automatically find patterns as rules in data. Rule-based systems are particularly applied when dependencies in data need to be modeled in a manner that is comprehensible to humans. Due to their simple structure, the rules closely reflect how humans act and think and can therefore be understood and interpreted directly by the user. In this research study, a rule-learning approach that automatically generates fuzzy rule bases has been extended and the influence of different parameters on the accuracy of the trained fuzzy model is examined. Even though these models are generally understandable, they suffer from the fact that usually a huge amount of rules has to be generated to explain all the details of the underlying data.

The focus of this thesis is on so-called hierarchical, rule-based models. This type of model hierarchy consists of different layers of simple rule models that describe the concept of the origin of the data in each layer according to a certain degree of detail. In the upper layers only a few rules exist that roughly approximate the concept, while rules further down in the hierarchy concentrate not only on details, but also on artifacts and outliers. A hierarchy of models is usually induced by a local learner, which completely explains the data throughout the hierarchy levels. By using a simple learning algorithm with an understandable hypothesis language, the hierarchy remains interpretable even in the event of complex concepts in the data. These hierarchical models help explore large amounts of data and may also be used

particularly for the classification of unknown data.

The final evaluation of the hierarchical approach with respect to the exploration and classification ability on selected benchmark data sets and on practical relevant data will confirm our assumptions. On the one hand, significantly smaller rule systems are generated, while on the other hand classification performance improves with fewer rules in each of the models of the hierarchy. The experiments on the hierarchical learner are conducted comparatively to the classical fuzzy rule learner, whose performance is previously evaluated by other recognized data analysis methods and using different parameters.

This work presents an approach for the extension of a rule-based learning method to generate hierarchically-structured rule models. Layered models are understandable and interpretable, and also have good classification properties. Based on a local learning method for the constructive training of fuzzy rules, the hierarchical fuzzy rule learner generates interpretable models. The trained fuzzy rule model generalizes and remains understandable even in the event of complex coherences, and at the time continues to be specific, as all the details of the data are explained by the different layers of the hierarchy.

A hierarchy of gradual models separates a complex system into different abstraction levels, each containing a simple and understandable rule model. The overall rule system opens insights for interactive exploration of the model and data at different levels within the hierarchy.

Danksagung

Mein besonderer Dank gilt meinem Doktorvater, Professor Michael Berthold, der mich nicht nur von Magdeburg nach San Francisco gelockt hat, um meine Diplomarbeit in reizvoller Umgebung zu bearbeiten, sondern im Anschluss daran mir die Möglichkeit gegeben hat, hier in Konstanz die Forschungsarbeit fortzusetzen. Hieraus entstand mit vielen erleuchtenden E-Mails und seinem Vorantreiben diese Dissertation, auf die ich stolz bin.

Des Weiteren möchte ich mich besonders auch bei Professor Rudolf Kruse bedanken, der mir während meiner Studienzeit Gelegenheit gegeben hat, mein Wissen durch mehrere interessante Auslandsaufenthalte praktisch anzuwenden und meine Diplomarbeit bei Michael Berthold anzufertigen. Auch möchte ich Ihnen und Professor Marcel Waldvogel dafür danken, dass sie als Prüfer meiner Arbeit zur Verfügung gestanden haben.

Vor allem möchte ich mich bei meiner Frau Yvonne und meinen Kindern, Yannek und Yaris, bedanken, die mich durch diesen Dissertationsmarathon begleitet und mit mir gelitten haben. Herzlich Dank für eure Unterstützung, den zugesprochenen Mut und die Zeit, die ihr mir gegeben habt. Mein Dank gilt auch meinen Eltern, Verwandten und Freunden, die mich durch ihr permanentes Nachfragen angehalten haben, das Ziel vor Augen nicht zu verlieren. Für die angenehme Arbeitsatmosphäre möchte ich ebenfalls beim gesamten Lehrstuhl bedanken.

Inhaltsverzeichnis

Kapitel 1

Motivation und Einleitung

Die heutigen Herausforderungen der sich schnell entwickelnden Informationsgesellschaft des 21. Jahrhunderts bestehen nicht nur darin, Daten zu speichern und zu verwalten, sondern auch die interessanten Informationen in den Daten zu finden und für den Menschen nutzbar zu machen. Menschen können wichtige Informationen leicht extrahieren, um die für den Zeitpunkt, Ort und Kontext wichtigen Details zu erhalten. Uninteressante Informationen werden ignoriert und gehen verloren. Neues Wissen kann durch bekannte Muster leichter geordnet werden, ohne sich mit irrelevanten Informationen aufzuhalten. Aber wie können diese Datenberge automatisch durch Maschinen verarbeitet und analysiert werden, um sinnvolle und gleichzeitig für den Menschen verständliche Zusammenhänge in den Daten zu finden?

Viele Unternehmen, Institutionen, Universitäten, Forschungseinrichtungen oder Behörden haben Rechnersysteme, in denen riesige Datenarchive von Zahlen, Bildern und Texten schlummern. Diese Datenfluten entstehen häufig automatisch. So werden zum Beispiel in einem einzigen Supermarkt viele Tausende Kauftransaktionen pro Tag gesammelt, wobei jede Aktion für sich wiederum aus einer Liste von Produkten besteht. In der Automobilindustrie werden bei der Herstellung eines Fahrzeuges alle Prozessschritte nachvollziehbar protokolliert. Beginnend bei den verarbeiteten Bauteilen, die sehr oft in anderen Werken gefertigt werden, über die Maschinen und deren aktuellen Parametern, bis hin zum fertigen Produkt mit all seinen produktspezifischen Eigenschaften. Auch in der Bioinformatik fallen große Datenbestände an, die zum Beispiel durch gleichzeitige Messung bestimmter Aktivitäten vieler Stoffe entstehen und später helfen sollen, das für eine bestimmte Krankheit passende Medikament zu entwickeln. Diese Beispiele zeigen, dass in vielen unterschiedlichen

Anwendungsbereichen immer größer werdende Datenmengen anfallen, die für den Menschen aber in ihrer ursprünglichen Form nutzlos sind. Erst durch die Verarbeitung und Analyse dieser Datenberge können nützliche Informationen gefunden werden. Maschinen können gewöhnlich nicht so einfach filtern und Muster erkennen. Hierfür bedarf es einer Logik, die die wichtigen von unwichtigen Daten unterscheidet und Zusammenhänge automatisch erkennt. Diese Ströme von einfließenden Daten werden oft aufwendig auf Datenmedien gespeichert und erst später für Analysezwecke verwendet. Dabei nutzt man leistungsfähige Rechner- und Softwaresysteme, die die Daten sammeln, verwalten und schließlich dem Anwender wieder geeignet zur Verfügung stellen.

Eine Möglichkeit der Wissensentdeckung in großen Datenbeständen ist die Anwendung von Verfahren, die sich dem Forschungsgebiet des Knowledge Discovery (KD) und Data Minings (DM) zuordnen lassen, siehe Fayyad u. a. (1996). Das Data Mining kommt erst zur Anwendung nachdem die Daten durch Selektion, Bereinigung und Transformation aufwendig aufbereitet wurden, siehe Abbildung 1.1. Diese Schritte nehmen einen Großteil der Zeit in Anspruch, bevor die eigentliche Suche nach Zusammenhängen in den Daten beginnen kann. Allgemein beschäftigen sich Anwendungen des Data Minings mit Modellierungs- und Entdeckungstechniken, um neue, unerwartete, valide, verständliche und verwertbare Informationen aus Datenbanken zu gewinnen. Sind diese Muster erst einmal in den Daten gefunden, kann mit der Interpretation dieser Zusammenhänge und deren Exploration begonnen werden. Hierbei ist es oft wichtig, dass der Anwender sein Modell versteht und Rückschlüsse auf das zugrunde liegende Konzept selbstständig durchführen kann. Die gewonnenen Hypothesen müssen entsprechend zielgerichtet evaluiert werden. Dabei wird neues Wissen oftmals erst sichtbar, wenn Zusammenhänge zwischen bekannten Fakten und unbekannten Phänomenen zusammen betrachtet werden.

Das Data Mining kann durch eine Liste von Aufgaben charakterisiert werden (Nakhaeizadeh, 1999). Diese umfassen die Segmentierung, die Konzeptbeschreibung, die Vorhersage, die Abweichungsanalyse und die Abhängigkeitsanalyse der Daten. Hierfür kommen neben Verfahren aus dem Bereich der Datenbanken (Data Warehouse) und der Statistik (Explorative Datenanalyse) auch Nicht-Standardansätze – Soft Computing, Künstliche Intelligenz und Maschinelles Lernen – zum Einsatz. Zu nennen sind hier bekannte Verfahren, wie Neuronale Netze, Regressionsanalysen, Diskriminanzanalysen, Zeitreihenanalysen, Entscheidungsbaumverfahren, Induktive Logik

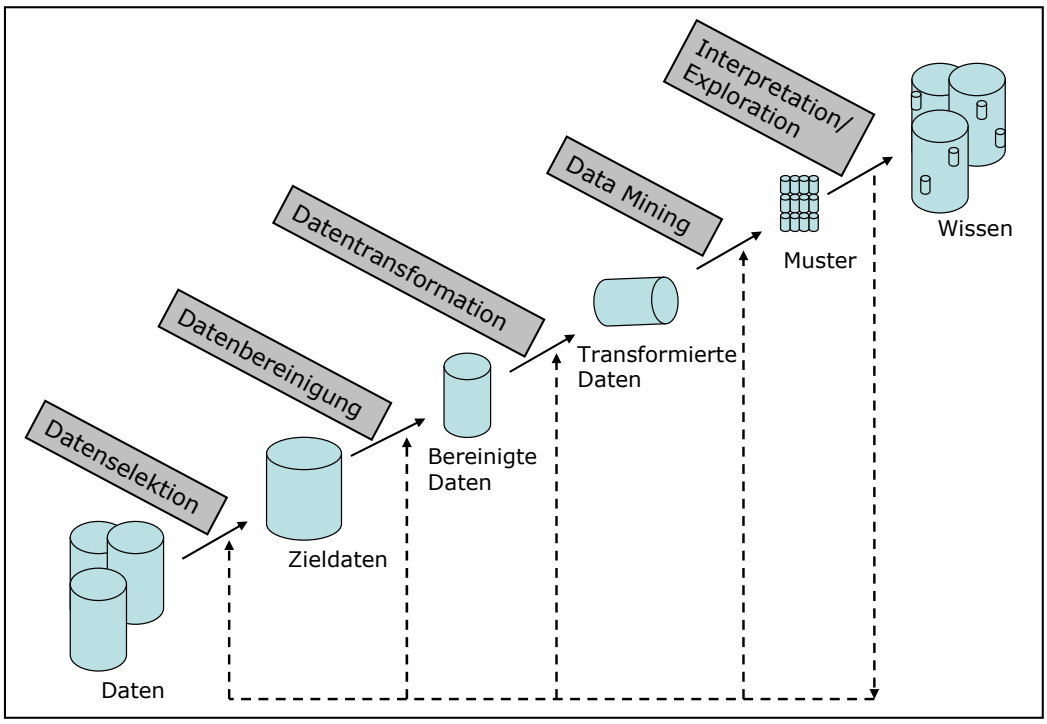


Abbildung 1.1: Zeigt die verschiedenen Aufgaben im Prozess der Wissensentdeckung von den Rohdaten bis zu neuem Wissen. Wichtig ist hier der Schritt des Data Minings, der Muster in den Daten findet und die spätere Interpretation und Exploration vorbereitet, um (neues) Wissen daraus zu extrahieren.

Programmierung, (Fuzzy)-Clusteranalysen und (Neuro)-Fuzzy-Systeme.

Auf dem Markt existieren einige erfolgreiche Softwareanwendungen zur intelligenten Datenanalyse und zur Daten-Exploration. Einen Überblick findet man in Gentsch (2001) und mayato (2008). Diese Verfahren vereinen die genannten Aufgaben in einem mächtigen Werkzeug. Dem Benutzer wird die Möglichkeit gegeben, mit einfachen, für eine bestimmte Aufgabe entworfenen Modulen, nützliche Informationen und Zusammenhänge in seinen gesammelten Daten selbstständig zu entdecken. Diese Tools helfen, den umfangreichen und komplexen Prozess von der Vorverarbeitung, über die Analyse, bis hin zur Auswertung und Exploration zu modulieren, und begleiten hierbei den Benutzer unterstützend, um verschiedene Data Mining Verfahren zu testen und zu evaluieren.

Das Data Mining kann als ein integrierter Prozess verstanden werden, der durch gezielte Anwendung von Entdeckungstechniken Zusammenhänge in Daten findet,

die ansonsten nicht sofort in den Daten erkennbar sind. Viele dieser Methoden, darunter Neuronale Netze oder Clustering-Verfahren, generieren Modelle, die von einem Anwender nicht oder nur kaum verstanden werden können. Andere Techniken, wie zum Beispiel klassische Regellernverfahren und Entscheidungsbaumlerner, generieren regelbasierte Vorhersagemodelle, die von einem Menschen direkt nachvollzogen und interpretiert werden können. Gerade diese Verfahren sind von besonderem Interesse, wenn es um die Wissensextraktion aus Daten geht.

Für einen Experten ist es sehr wichtig, sein generiertes Modell zu verstehen und sein gewonnenes Wissen anzuwenden. Dazu gehört auch, dass die erzeugten Hypothesen sich in den Daten widerspiegeln und er damit in der Lage ist, Rückschlüsse auf die Daten abzuleiten – gerade wenn Fehlmessungen und/oder Artefakte erkannt werden sollen. Hierbei kommen oftmals Visualisierungstechniken aus dem Gebiet der explorativen Datenanalyse zum Einsatz. Diese Sichten stellen das Vorhersagemodell in einer für den Benutzer verständlichen Form dar und ermöglichen dadurch die Interpretation und Exploration des Modells. Mit dem Einsatz von Visualisierungstechniken nimmt die Anforderung an eine klare Zielstellung ab. Der Hauptaufwand liegt nun beim Benutzer, der die Ergebnisse des Data Mining Prozesses interpretieren und evaluieren muss. Der Anwender ist durch die Kombination von geeigneten Modellen und Visualisierungsmethoden in der Lage, seine Hypothesen zielgerichtet zu evaluieren. Hierbei helfen ihm gerade verständliche Modelle, um sein gewonnenes Wissen auf seine Daten anwenden zu können.

1.1 Regelextraktion aus Daten

Ein Anwender hat gewöhnlich nur ein ungefähres Verständnis von seinen Daten und kann Hypothesen nur unvollständig formulieren und Abhängigkeiten erraten. Oft sind die Daten unstrukturiert und nicht überschaubar, sodass es sehr schwer ist, die interessanten und relevanten Informationen zu extrahieren. Ob diese Informationen seine Modelle stützen, weiß er selten. Deshalb ist es notwendig, verständliche und am besten in sprachliche Regeln gefasste Modelle zu erzeugen. Hier bieten sich gerade Fuzzy-Methoden an, um die „unscharfen“ (fuzzy) Zusammenhänge sprachlicher Ausdrücke zu modellieren. Fuzzy-Lösungen sind leicht zu verstehen, da sie dem menschlichen Denken und der Art der Formulierung sehr viel näher kommen und einfacher anzuwenden sind. Sie erlauben es, nicht präzise Informationen zu verar-

beiten (Kruse u. a., 1999). Fuzzy-Systeme werden nicht nur anhand ihrer Exaktheit der erzielten Ergebnisse, sondern in erster Linie auch nach der Einfachheit beurteilt. Deshalb werden sie dort verwendet, wo der Anwender durch Untersuchung einer gelernten Regelbasis Einsichten in die Zusammenhänge seiner Daten gewinnen will.

Die regelbasierten Verfahren erzeugen einen Satz von Regeln, der das zugrunde liegende Konzept der Entstehung der Daten approximiert. Sind die Daten und die Art ihrer Erzeugung bekannt, ist es manchmal möglich, die Regeln manuell für die Klassifikation aufzustellen. Dafür muss aber der Klassifikationsprozess vollständig von einem Experten verstanden worden sein. In vielen praktischen Anwendungen ist diese Voraussetzung nicht gegeben, sodass das Regelmodell automatisch anhand von Trainingsmustern erzeugt werden muss. Es wird versucht, ein Modell in Form von Regeln zu induzieren, das hinterher eine Interpretation erlaubt und damit Einblicke in die Art der gefundenen Klassifikation gestattet.

Ein Regelsystem besteht aus einer Regelbasis (Struktur) und Partitionen der Wertebereiche aller Variablen (Parameter). Diese Struktur und deren Parameter gilt es, durch die Analyse der verfügbaren Daten zu bestimmen. Hierbei muss oftmals ein Kompromiss zwischen Genauigkeit und Interpretierbarkeit des Modells gefunden werden. Gute Regelsysteme zeichnen sich durch eine möglichst geringe Anzahl von Regeln aus, die durch wenige Variablen (Dimensionen) eingeschränkt sind, aber dennoch gute Klassifikationsergebnisse liefern.

Regelsysteme können als Klassifikatoren verwendet werden, deren Aufgabenstellung darin besteht, bei der automatischen Klassifikation von Mustern anhand einer Menge von vorklassifizierten Trainingsdaten, neue, noch unbekannte Muster einer korrekten Klasse zuzuordnen. Diese Muster sollen möglichst korrekt klassifiziert werden (Fukunaga, 1990). Die Generalisierungsfähigkeit des Modells beschreibt die Klassifikationsleistung auf unbekannten Daten (Krogh und Vedelsby, 1995; Holden und Anthony, 1993). Die Testdaten werden nicht für die Erzeugung des Klassifikators verwendet, sondern ausschließlich für die Messung der Generalisierungsfähigkeit des Klassifikators.

Die Aufgabe der Klassifikatorerzeugung besteht darin, eine Hypothese zu finden, die dem zugrunde liegenden Konzept möglichst nahe kommt. Das heißt, dass die Hypothese in einem möglichst großen Bereich des Merkmalsraumes mit dem Konzept übereinstimmt. Je mehr Hypothese und Konzept überlappen, desto besser ist der erzeugte Klassifikator. Fehler entstehen oftmals durch den verwendeten Algorithmus,

der die Hypothese anhand der Trainingsdaten erzeugt und nur eine bestimmte, eingeschränkte Struktur erzeugen kann. In der Praxis werden Fehler oft durch die Daten verursacht, die durch Fehlmessungen, fehlerhafte Aufzeichnungen oder auch durch unvollständige Trainingsdaten entstehen.

Der Nachteil dieser Regellerner ist, dass sie das Konzept durch die Form der Regeln nur eingeschränkt beschreiben können und deswegen oftmals sehr viele, sehr detaillierte Regeln benötigt werden, um alle Details eines komplexen Systems zu modellieren. Dadurch wird die Interpretation des Modells erschwert, weil es für den Experten kaum möglich ist, die häufig umfangreichen Regelbasen zu überschauen.

Lernen von Fuzzy-Regelsystemen

Fuzzy-Systeme sind wegen ihrer besseren Interpretierbarkeit beim Anwender beliebt und kommen daher gewöhnlich in der (Fuzzy)-Regelungstechnik zum Einsatz. In der Fuzzy-Regelung werden Signalen, wie Regelgröße, Regelfehler oder Stellwert, symbolische Variablen anstatt exakter numerischer Werte zugewiesen (Tsoukalas und Uhrig, 1996). Oftmals ist kein formaler Reglerentwurf für einen Prozess praktikabel, sondern nur intuitives Expertenwissen über die manuelle Regelung des Prozesses vorhanden. In diesen Fällen werden Fuzzy-Regelungssysteme angewendet.

Fuzzy-Regeln sind im Bereich des Data Minings interessant und erfolgreich, weil sie verständliche und interpretierbare Modelle erzeugen. Dennoch leiden auch diese Ansätze darunter, viele Regeln zu erzeugen, wenn komplexe Konzepte approximiert werden sollen. Hierdurch wird die Interpretation des Gesamtmodells erschwert, weil viele Detailregeln erzeugt werden, die nur einen geringen Einfluss auf die Modellierung des Gesamtkonzeptes haben und daher für die Betrachtung im ersten Schritt häufig nicht relevant sind. Diese Regeln mit niedriger Relevanz werden erst dann interessant, wenn bestimmte Details des Modells exploriert werden sollen. Viele Regellerner erzeugen eine unstrukturierte Regelbasis, in der Regeln unterschiedlicher Größe und Relevanz zusammen in einem monolithischen Modell gehalten werden. Zwar können diese Regeln anhand lokaler Eigenschaften, wie Gewicht und Größe der Regeln, sortiert werden, dennoch sind diese Möglichkeiten oftmals nicht praktikabel, wenn nur bestimmte Bereiche im Merkmalsraum exploriert werden sollen.

Ein lokales Regellernverfahren wird in dieser Arbeit verwendet, um durch eine einfache Filterheuristik eine Hierarchie von Basismodellen zu generieren. Diese Hie-

rarchie von Regelmodellen vereint die Eigenschaften eines allgemeinen und eines detaillierten Systems zugleich. Das System beschreibt alle Details der Daten, bleibt aber aufgrund seiner schichtenartigen Formation verständlich, da jede Schicht in der Hierarchie nur wenige Regeln enthält. Hierbei wird in den oberen Schichten das Konzept grob beschrieben, weiter unten in der Hierarchie findet man speziellere Regeln, die alle Feinheiten der Daten beschreiben. Damit ermöglicht diese Hierarchie von Regelmodellen eine graduelle Betrachtung des zugrunde liegenden Konzeptes.

1.2 Hierarchische Fuzzy-Regelmodelle

Tagtäglich begegnet man Situationen, in denen zum Beispiel Abstraktionen von sprachlichen Ausdrücken verwendet werden, um komplexe Zusammenhänge mit einfachen Begriffen zu beschreiben. Wörter wie langsam, mittel, zügig, schnell, sehr schnell und rasen werden typischerweise verwendet, wenn man über Geschwindigkeiten redet. Beschreibt man dagegen eine zeitliche Domäne, verwendet man Wörter wie Morgen, Mittag, Abend, Nacht oder Mitternacht. Diese Konzepte helfen, Zusammenhänge zu beschreiben und zu strukturieren, um darauf Aktionen auszuführen. Abstraktionen sind wichtig, um unsere Welt zu verstehen und komplexe Beziehungen mit einfachen Mitteln darzustellen. Hierarchische Strukturen findet man in vielen natürlichen, künstlichen und abstrakten Systemen und diese spiegeln die Ordnung, Stabilität und Kontrolle des Systems wider. Um eine Hierarchie zu konzipieren, bedarf es unterschiedlicher Elemente, Relationen, Schichten und einem Prozess oder Algorithmus, der die Hierarchie generiert.

Die Konstruktion einer modularen Regelhierarchie – wie sie in dieser Arbeit betrachtet wird – geschieht auf Basis eines konstruktiven Trainingsalgorithmus zum Lernen von Fuzzy-Regeln. Die Schichten der Regelhierarchie werden von unten (spezifisch) nach oben (generell) erzeugt. Jedes Regelmodell wird auf den Eingabedaten trainiert, welche zuvor durch das darunter liegende Regelmodell gefiltert wurden. Hierdurch entsteht eine gekoppelte Hierarchie von Regelmodellen mit einer klaren Beziehung zwischen den Ebenen. Diese Beziehung ist wichtig, um später selektiv bestimmte Bereiche des Modells zu explorieren.

Die regelbasierten Modelle sind beim Anwender beliebt, weil die gelernten Modelle direkt interpretiert werden können. Hierzu gehören Entscheidungsbaumverfahren, statistische und direkte Regellerner sowie Fuzzy-Regellerner. Diese Modelle leiden

oftmals darunter, dass sie zu viele Regeln auf komplexen Datensätzen erzeugen. Es existieren erweiterte Ansätze, um diesem Problem zu begegnen. Einige versuchen durch eine unterschiedliche Granulierung des Eingaberaumes, die Anzahl der Regeln zu minimieren, erzeugen aber mitunter sehr viele Regeln, wenn komplexe Strukturen im Datenraum erklärt werden müssen. Ansätze, die Ausnahmeregeln generieren, wurden ebenfalls in der Vergangenheit angewendet. Diese Verfahren erzeugen keine flachen Regeln im klassischen Sinn, sondern verschachtelte Regeln, wobei die Konklusion einer Regel die Ausnahme enthält. Alle diese Verfahren haben Schwierigkeiten, wenn komplexe Konzepte durch einfache, verständliche Modelle beschrieben werden sollen, da sich diese Ansätze häufig auf Artefakte und Ausnahmen in den Daten konzentrieren und dadurch umfangreiche Regelbasen erzeugen. Diese Modelle verändern zum Teil die Art der Hypothesensprache, weil nicht mehr einfache Regeln für die Modellformulierung verwendet werden, sondern Regeln in Regeln geschachtelt werden oder sogar Regeln auf Regeln verweisen. Damit sind diese Regelmodelle schwer zu interpretieren und für den Anwender häufig nicht anwendbar.

Der in dieser Arbeit verfolgte hierarchische Ansatz erzeugt Fuzzy-Regelmodelle unterschiedlicher Detailgrade durch Extraktion von Ausreißer-Regelmodellen, wie diese erstmals in Berthold (2000) vorgeschlagen wurden. Hierbei werden im ersten Schritt Regeln mit niedriger Relevanz herausgefiltert und in einem separaten Ausreißermodell erklärt. Mit den verbliebenen Daten wird im zweiten Schritt ein generelleres Modell trainiert, welches größere Bereiche des Eingaberaumes zusammenfasst. Dieses Vorgehen kann so lange wiederholt werden, bis keine Regeln mehr mit niedriger Relevanz extrahiert werden können. Das resultierende System ist eine Hierarchie von Regelmodellen.

Hieraus leiten sich zwei natürliche Varianten von Modellhierarchien ab. Eine Hierarchie von Detailmodellen, die auf den unteren Schichten nur Detailregeln enthalten und alle Feinheiten, aber auch Ausreißer, in den Daten beschreiben. Auf der obersten Schicht werden die Daten sehr allgemein durch nur wenige, große Regeln erklärt, die nach dem Filterprozess im Modell verbleiben. Die zweite Variante ist eine Hierarchie von robusten (generellen) Regelmodellen, die auf Basis der Detailmodelle entstehen. Diese Modellhierarchien basieren auf dem Regelmodell jeder Schicht, das nach Entfernen der Detailregeln erhalten bleibt. In beiden Fällen entsteht eine Hierarchie von Regelmodellen, die das Konzept hinter den Daten durch die gleiche, einfache Hypothesensprache wie der verwendete lokale Algorithmus beschreiben.

1.3 Ziele der Arbeit

Diese Arbeit befasst sich mit Ansätzen zum Lernen von Regelmodellen. Diese Methoden leiden häufig darunter, dass sie sehr viele Detailregeln generieren, die eine Interpretation der Regelbasis erschweren, wenn nicht unmöglich machen. Es wird beispielhaft ein Verfahren genauer betrachtet, welches interpretierbare Fuzzy-Regelbasen erzeugt. Um die Leistungsfähigkeit dieses Verfahrens beurteilen zu können, wird dieses unter dem Einfluss verschiedener Parameter auf bekannten Benchmark-Datensätzen evaluiert. Es wird gezeigt, dass solche Verfahren vergleichbare Ergebnisse zu anderen Standard-Methoden liefern, aber dennoch praktisch oft unbrauchbar sind.

Unsere weiteren Untersuchungen sollen zeigen, dass sich lokale Lernverfahren allgemein erweitern lassen, um hierauf hierarchisch-organisierte Modelle zu erzeugen. Es soll untersucht werden, welche Eigenschaften diese Modelle in Bezug auf Komplexität, Exploration- und Klassifikationsfähigkeit besitzen. Die Annahme, dass solche Modelle eine niedrigere Komplexität, aber dennoch gute Klassifikationsergebnisse liefern, wird im zweiten Teil der Arbeit basierend auf einem lokalen Regellernverfahren diskutiert. Um die Ergebnisse direkt mit dem flachen, nicht-hierarchischen Regelsystemen vergleichen zu können, wird die generierte Regelhierarchie auf denselben Datensätzen erzeugt. Die Experimente sollen zum einen zeigen, dass hierarchische Regelsystem weniger Regeln enthalten können, und zum anderen ähnliche Klassifikationsergebnisse liefern.

Die nachfolgende Liste von Publikationen weist die Forschungsarbeiten aus, die im Zusammenhang mit der Konstruktion, Evaluierung und Visualisierung von hierarchischen Fuzzy-Regelsystemen während meiner Promotionszeit an der Universität Konstanz entstanden sind:

- Thomas R. Gabriel and Michael R. Berthold, *Constructing Hierarchical Rule Systems*, Advances in Intelligent Data Analysis, Proc. 5th International Symposium on Intelligent Data Analysis (IDA), Lecture Notes in Computer Science (LNCS 2810), Springer Verlag, pp. 76–87, 2003.

- Thomas R. Gabriel and Michael R. Berthold, *Formation of Hierarchical Fuzzy Rule Systems*, Proc. 22nd Conference North American Fuzzy Information Processing Society (NAFIPS), pp. 87–92, 2003.

- Thomas R. Gabriel and Michael R. Berthold, *Influence of fuzzy norms and other*

heuristics on "Mixed Fuzzy Rule Formation", International Journal of Approximate Reasoning (IJAR), Elsevier, pp. 35:195–202, 2004.

- Thomas R. Gabriel and Michael R. Berthold, *Missing Values in Fuzzy Rule Induction*, IEEE Conference on Systems, Man and Cybernetics (IEEE SMC), IEEE Press, pp. 2:1473-1476, 2005.

- Thomas R. Gabriel, A. Simona Pintilie, and Michael R. Berthold, *Exploring Hierarchical Rule Systems in Parallel Coordinates*, Proc. 6th International Symposium on Intelligent Data Analysis (IDA), Lecture Notes in Computer Science (LNCS 3646), Springer Verlag, pp. 97–108, 2005.

- Thomas R. Gabriel, Kilian Thiel, and Michael R. Berthold, *Rule Visualization based on Multi-Dimensional Scaling*, IEEE International Conference on Fuzzy Systems, pp. 66-71, 2006.

- Thomas R. Gabriel, Kilian Thiel, and Michael R. Berthold, *Multi-Dimensional Scaling applied to Hierarchical Rule Systems*, Proceedings of the IFSA World Congress and the 6th Conference of EUSFLAT, pp. 944–949, 2009.

Die Grundlage hierarchischer Regelsysteme legte die Veröffentlichung *Constructing Hierarchical Rule Systems (2003)*, die die initiale Idee des Lernens von Regelhierarchien beschreibt und diese anhand der Erweiterung eines klassischen Fuzzy-Regellerners erläutert. In der zweiten und dritten Publikation werden Einflussmöglichkeiten auf die Modellbildung des Fuzzy-Regellerners beschrieben. Hierbei wird speziell auf verschiedene Fuzzy-Normen und Heuristiken eingegangen, sowie Ansätze aufgezeigt, die Daten mit fehlenden Werten in den Lernprozess integrieren. Der anschließende Artikel *Exploring Hierarchical Rule Systems in Parallel Coordinates (2005)* zeigt, wie hierarchische Regelsysteme mithilfe von Parallelen Koordinaten visualisiert werden können und sich dadurch explorative Möglichkeiten der interaktiven Datenanalyse ergeben. Zwei weitere Papers, die sich mit der Visualisierung von (hierarchischen) Regelsystemen beschäftigen, wurden 2006 und 2009 veröffentlicht. Die hierin vorgestellten Techniken visualisieren die gelernten Regeln zusammen mit den zugrunde liegenden Daten in einer zusammengefassten Darstellung und erlauben die interaktive Analyse der Regeln zwischen den Ebenen der Hierarchie.

1.4 Aufbau der Arbeit

Nach einer Einführung in die regelbasierten Lernverfahren in Kapitel 2 wird speziell auf die fuzzy-theoretischen Grundlagen, Fuzzy-Mengen und Fuzzy-Operatoren, eingegangen. Diese werden im weiteren Verlauf aufgegriffen, um verschiedene Ansätze des Erzeugens von Fuzzy-Regelsystemen zu diskutieren, und um anschließend ein spezielles Trainingsverfahren zum Lernen von Fuzzy-Regeln zu beschreiben, siehe Abschnitt 2.4. Hierfür werden verschiedene Fuzzy-Normen, Konfliktlösungsstrategien und die Behandlung von fehlenden Werten diskutiert. Im nächsten Kapitel 3 wird die Klassifikationsleistung des vorgestellten Fuzzy-Regellerners durch verschiedene Experimente auf bekannten Benchmark-Datensätzen evaluiert. Ebenfalls werden die Fuzzy-Normen (siehe Kapitel 2.3) und Heuristiken zur Konfliktlösung (siehe Kapitel 2.4) experimentell verglichen. In Kapitel 4 wird dann zu Ansätzen übergegangen, die, wie zum Beispiel Entscheidungsbaum- oder hierarchische Clustering-Verfahren, Modelle hierarchischer Struktur lernen. Es wird eine Formalisierung für Regelsysteme aufgestellt, die im nachfolgenden Kapitel 5 aufgegriffen wird, um eine hierarchische Erweiterung des Fuzzy-Regellerners vorzustellen. Diese basiert auf einem Filteransatz, der es erlaubt, automatisch Regelhierarchien auf Basis eines lokalen Lernverfahrens zu erzeugen. Das hierarchische Verfahren wird in Kapitel 6 auf Benchmark-Datensätzen des StatLog-Projekts (Michie u. a., 1994) und einem bekannten Datensatz aus der Bioinformatik validiert, um die Leistungsfähigkeit mit dem klassischen, lokalen Verfahren und anderen Datenanalyseverfahren vergleichen zu können. In Kapitel 7 werden die Ergebnisse dieser Arbeit zusammengefasst und abschließend diskutiert.

Kapitel 2

Fuzzy-Regelmodelle

Im Zusammenhang mit Verfahren des Data Minings ist es von großem Interesse, Regeln automatisch aus Daten extrahieren zu können. Hierfür ist es wichtig, einfache Verfahren zur Erzeugung eines verständlichen Regelmodells zu haben. Im ersten Teil dieses Kapitels werden klassische Regellernverfahren beschrieben. In den anschließenden Abschnitten werden die Grundlagen der Fuzzy-Datenanalyse formalisiert und es wird ein Überblick gegeben, um später Fuzzy-Regelmodelle aus Daten lernen zu können. Im Anschluss wird vertiefend auf ein Fuzzy-Regellernverfahren eingegangen und es werden dessen Anpassungs- und Erweiterungsmöglichkeiten diskutiert.

2.1 Klassische Regelsysteme

Das Lernen von Regeln kann als das Finden einer geeigneten Menge von Regeln verstanden werden, die von einem Lerner durch eine Menge von Trainingsdaten induziert wird. Durch einen gegebenen Algorithmus wird das Regelsystem so lange erweitert und angepasst, bis das Modell eine möglichst gute Approximation auf den Daten darstellt. Das erzeugte Regelsystem besteht aus einer Struktur und Partitionen der Wertebereiche aller Parameter, die durch die Analyse der verfügbaren Trainingsdaten bestimmt werden sollen. Gute Regelsysteme zeichnen sich durch möglichst wenige Regeln aus, die auf wenigen Variablen eingeschränkt sind. Hierbei muss oftmals ein Kompromiss zwischen Genauigkeit und Verständlichkeit dieser Systeme gefunden werden: Auf der einen Seite soll eine gute Interpretierbarkeit des Modells gewährleistet werden, auf der anderen Seite sollen Vorhersagemodelle gute Klassifikationsergebnisse auf unbekannten Daten liefern.

Eine Regelbasis $\mathcal{R} = \{R_1, \ldots, R_{r_1}, \ldots, R_j, \ldots, R_{r_c}\}$ mit $1 \leq r_j \leq M$ Regeln für die Klasse $\{y_1, \ldots, y_j, \ldots, y_c\} \in C$ wird durch einen Lernalgorithmus auf einer gegebenen Trainingsmenge $T = (\vec{x}, y) = \{(\vec{x_1}, y_1), (\vec{x_2}, y_2) \ldots, (\vec{x_m}, y_m)\}$ mit $1 \leq i \leq m, \forall \vec{x_i} \in \mathbb{R}^n$ und $(x, y) \in T$ induziert. Jede Regel $R = (A, y) \in \mathcal{R}$ ist hierbei eine Abbildung von einem Eigenschaftsraum $E = E_1 \times E_2 \times \ldots \times E_d$ der Dimension d mit $E^t = (E_1^t, \ldots, E_N^t)$ auf ein Klasse $y_j \in C$. Ein Regelsystem $\mathcal{R}$ kann als eine Menge von Regeln R_j definiert werden:

$$
\begin{array}{lllllllll}
R_1: & \texttt{IF} & x_1 \in E_{1,1} & \times & \cdots & \times & x_n \in E_{n,1} & \texttt{THEN} & y_1 \\
 & & \vdots & & & & \vdots & & \vdots \\
R_{r_1}: & \texttt{IF} & x_1 \in E_{1,r_1} & \times & \cdots & \times & x_n \in E_{n,r_1} & \texttt{THEN} & y_1 \\
 & & \vdots & & & & \vdots & & \vdots \\
R_j: & \texttt{IF} & x_1 \in E_{1,j} & \times & \cdots & \times & x_n \in E_{n,j} & \texttt{THEN} & y_j \\
 & & \vdots & & & & \vdots & & \vdots \\
R_{r_c}: & \texttt{IF} & x_1 \in E_{1,r_c} & \times & \cdots & \times & x_n \in E_{n,r_c} & \texttt{THEN} & y_c.
\end{array}
$$

Damit ergibt sich eine formale Definition für Regeln basierend auf einer einfachen Hypothesensprache, mit deren die Regelstruktur durch ein geeignetes Regellernverfahren induziert werden kann.

Die erweiterten, strukturorientierten Verfahren versuchen, komplexe Konzepte durch mehrschichtige Modelle oder Ausreißerregeln zu beschreiben. Diese Ansätze modellieren Ausnahmen oder Bereiche niedriger Relevanz speziell durch eine variable Granulierung der Dimensionen und versuchen dadurch, die Regelbasis klein und explorierbar zu belassen. Strukturorientierte Ansätze lassen sich als Spezialfälle von hyperquaderorientierten Verfahren verstehen, die nicht nach Clustern im Datenraum suchen, sondern Hyperquader auswählen, die in einer Gitterstruktur angeordnet sind. Durch die Definition initialer Fuzzy-Mengen für jede Variable wird der Datenraum durch sich überlappende Hyperquader überdeckt. Diese Verfahren generieren eine Menge von Prototypen, die Bereiche im Eingaberaum zusammenfassen, in denen Muster gleicher Klasse gefunden werden. Diese Regeln werden unabhängig voneinander in einem global-partitionierten Modell generiert.

Die prototypbasierten Lernverfahren erzeugen typischerweise achsenparallele Hyperrechtecke im Eingaberaum; zum Beispiel Regellernverfahren von Salzberg (1991) und Wettschereck u. a. (1995). Diese Verfahren erlauben die Konstruktion neuer Instanzen während des Trainings, die oft als Repräsentant oder eben Prototyp einer ganzen Gruppe von Beobachtungen dienen. Diese Gruppe von abgedeckten Instan-

zen kann durch eine allgemeine Beschreibung in Form eines Hyperrechtecks ersetzt werden und damit auch als instanzbasiertes Lernverfahren angesehen werden.

2.2 Lernen von Fuzzy-Modellen

Es existieren verschiedene Möglichkeiten, Fuzzy-Modelle automatisch zu lernen, wobei man diese in drei Klassen unterteilen kann. Dazu zählen die clusterbasierten und hyperquaderorientierten Ansätze, die Regeln (Struktur) und die Fuzzy-Mengen (Parameter) des Fuzzy-Systems gleichzeitig erlernen. Bei den clusterbasierten Ansätzen handelt es sich um unüberwachte Lernverfahren, da es keine vorgegebene Zielgröße gibt. Man spricht bei diesen Verfahren auch von Fuzzy-Clusteranalyse (Bezdek u. a., 1998; Höppner u. a., 1999). Dagegen sind die hyperquaderorientierten Methoden überwachte Lernverfahren, bei denen versucht wird, den Eingaberaum durch (überlappende) mehrdimensionale Quader (Hyperquader) abzudecken, um so die Klassenzuordnung durch Fuzzy-Grafen oder Fuzzy-Regeln zu beschreiben. Diese Systeme benötigen eine initiale Fuzzy-Partitionierung des Wertebereiches, um die Regelbasis zu erzeugen (Nauck und Kruse, 1998). In beiden Verfahren werden die Fuzzy-Mengen aus der Projektion der Cluster bzw. Hyperquader auf die Wertebereiche der einzelnen Dimensionen der Eingabe gewonnen. Das Hauptproblem dieser Ansätze ist, dass jede erzeugte Fuzzy-Regel eigene spezifische Fuzzy-Mengen verwendet und deshalb die Regelbasis unter Umständen nur schwer zu interpretieren ist. Clusterbasierte Verfahren kranken außerdem daran, dass durch die Projektion auf die einzelnen Variablen Informationen über die eventuell nicht achsenparallele Form der Cluster verloren gehen. Ein weiterer Nachteil ist die Bestimmung der Regelbasisgröße, die oftmals nur durch mehrere aufwendige Tests bestimmt werden kann, um anschließend die Ergebnisse bewerten und vergleichen zu können.

Ein Ansatz, um unscharfe Regeln aus Beispieldaten abzuleiten, wurde von Wang und Mendel (1992) vorgeschlagen. Um diesen Algorithmus anwenden zu können, müssen alle Variablen durch Fuzzy-Mengen partitioniert werden. Dazu verwendet man üblicherweise gleichmäßig verteilte, sich überlappende dreieck- oder trapezförmige Fuzzy-Zugehörigkeitsfunktionen. Auf diese Weise wird der Daten- oder Merkmalsraum durch sich überlagernde mehrdimensionale Fuzzy-Mengen beschrieben, die sich zu Hyperquadern formieren. Die Regeln werden durch Auswahl der Hyperquader generiert, die die Trainingsdaten enthalten. Um zwischen verschiedenen

Ausgabewerten für identische Eingabewerte zu mitteln, werden diese Regeln gewichtet. Das Wand&Mendel-Verfahren kann auch für die Funktionsapproximation verwendet werden. Dadurch können reelwertige, stetige Funktionen durch eine Menge von Fuzzy-Regeln beliebig genau approximiert werden. Strukturorientierte Ansätze vermeiden diese Nachteile, weil sie nicht nach (hyperellipsoid- oder hyperquaderförmigen) Clustern im Datenraum suchen. Durch die vorgegebene (Anfangs)-Fuzzy-Partitionierung der Wertebereiche wird über den Datenraum ein mehrdimensionales Fuzzy-Gitter gelegt. Aus diesem Gitter wird eine Regelbasis bestimmt, indem die besetzten Gitterzellen ausgewählt und durch Fuzzy-Regeln beschrieben werden. Nachdem so die Regelbasis festgelegt ist, werden üblicherweise die Fuzzy-Mengen trainiert, um die Leistung des Fuzzy-Systems zu verbessern. Die Schwierigkeit bei der Konstruktion solcher Regler besteht neben der Bestimmung der entsprechenden Regeln hauptsächlich in der Beschreibung der Zugehörigkeitsfunktionen.

Ein weiterer Vertreter der prototypbasierten Verfahren ist eine Variante des Wang&-Mendel-Algorithmus und wurde von Higgins und Goodman (1993) entwickelt. Dieser Algorithmus erzeugt Fuzzy-Partitionen durch Verfeinerung der bestehenden Partitionierung der Eingabe während des Trainings. Der Algorithmus beginnt mit einer Fuzzy-Menge je Variable, sodass der Datenraum durch einen einzelnen Hyperquader abgedeckt wird. Anschließend werden neue Zugehörigkeitsfunktionen an Punkten maximalen Fehlers erzeugt, in denen die Fuzzy-Partitionen aller Variablen weiter verfeinert werden. Die alten Fuzzy-Regeln werden verworfen, und auf Grundlage der verfeinerten Partitionen wird eine neue Regelbasis erzeugt. Diese Prozedur wird so lange wiederholt, bis eine maximale Anzahl von Fuzzy-Mengen erzeugt wurde, oder der Fehler eine gegebene Schranke unterschreitet. Der Higgins&Goodman-Algorithmus wurde entworfen, um eine Schwäche des Wang&Mendel-Algorithmus auszugleichen, der Schwierigkeiten bei der Approximation extremer Funktionswerte hat. Allerdings tendiert dieser Algorithmus dazu, Ausreißer in der Trainingsmenge zu modellieren, da er sich auf Bereiche mit großen Approximationsfehlern konzentriert.

Eine erweiterte Version wird in dem System NEFCLASS (NEuro-Fuzzy-CLASSification) von Nauck u. a. (1997) verwendet. Dieses Verfahren nutzt ein Gütemaß zur Bewertung der gefundenen Fuzzy-Regeln. Auf diese Weise kann die Größe der Regelbasis automatisch bestimmt werden, indem Regeln dem System nach absteigender Güte hinzugefügt werden, bis alle Trainingsbeispiele abgedeckt sind. Mithilfe des Gütemaßes wird außerdem die beste Folgerung für jede Regel bestimmt. Weiterhin

kann die Zahl der Fuzzy-Regeln begrenzt werden, indem nur die besten Regeln in die Regelbasis aufgenommen werden. Auch ist es möglich, die Zahl der Regeln und die Zahl der je Regel benutzten Variablen durch Stutzen der Regelbasis („Pruning“) zu verringern. NEFCLASS verwendet ein einfaches, backpropagationartiges Verfahren, das durch Lernverfahren für Neuronale Netze inspiriert ist, um die Fuzzy-Mengen zu optimieren. Daher auch die Bezeichnung „Neuro-Fuzzy“ für diesen und verwandte Ansätze (Kruse u. a., 1999). Der Algorithmus führt jedoch nicht, wie das normale Backpropagation-Verfahren, einen Gradientenabstieg durch, da der Erfüllungsgrad einer Regel über das Minimum bestimmt wird. Außerdem werden die Fuzzy-Mengen oft durch nicht überall differenzierbare Funktionen beschrieben. Stattdessen wird eine einfache Heuristik benutzt, durch die die Fuzzy-Mengen verschoben und in ihren Formen verändert werden. Nachteil dieses Verfahrens ist, dass sich keine individuellen Zugehörigkeitsfunktionen einstellen lassen. Vielmehr basiert das Verfahren auf einer globalen Granulierung des Eingaberaumes. Das führt dazu, dass insbesondere in hochdimensionalen Eingaberäumen sehr viele Regeln erzeugt werden.

2.3 Fuzzy-Theorie und Fuzzy-Regelsysteme

In diesem Abschnitt wird auf die von Zadeh (1965) entwickelte Theorie der unscharfen Mengen (engl. *fuzzy set theory*) eingegangen, die im Gegensatz zur klassischen Logik auch partielle Zugehörigkeiten von Objekten zu einer Menge erlaubt. Im ersten Abschnitt werden die Grundlagen der Fuzzy-Theorie eingeführt, bevor im zweiten Teil Operatoren auf Fuzzy-Mengen diskutiert werden. Dieses Wissen wird anschließend auf Fuzzy-Regelsysteme übertragen.

Fuzzy-Mengentheorie

Im Gegensatz zur klassischen Logik werden in der Fuzzy-Logik auch partielle Zugehörigkeiten erlaubt. Dies impliziert den Begriff der *Zugehörigkeit* $\mu_A(x)$ eines Elements x zur Menge A und gestattet die Formalisierung auch vager bzw. umgangssprachlicher Zusammenhänge. Sie modelliert so die „Penumbra“ (Kruse u. a., 1994) („Halbschatten“) des Begriffs, der die zu definierende Eigenschaft bezeichnet und die unscharfen Grenzen (der Anwendbarkeit) eines sprachlichen Ausdrucks widerspiegelt. Die Zugehörigkeitsgrade können auch als *Ähnlichkeit*, *Präferenz* oder *Unsicherheit* gedeutet

werden (Dubois u. a., 1996). Diese können ausdrücken, wie ähnlich ein Objekt zu einem anderen Beispielobjekt (*Prototyp*) ist. Sie können Unsicherheiten oder Präferenzen über die wahre Situation darstellen, wenn diese durch unscharfe Begriffe beschrieben werden. In der klassischen Logik können Elemente nur Werte von 0 oder 1 ($x \in \{0, 1\}$) annehmen. Dagegen sind in der Fuzzy-Logik auch Werte zwischen 0 und 1 ($x \in [0, 1]$) möglich.

In den meisten Anwendungen der Fuzzy-Theorie wird jedem Attribut eine bestimmte Anzahl von *linguistischen Termen* zugeordnet. Zum Beispiel kann das Merkmal (oder die *linguistische Variable*) *Temperatur* in die Terme *kalt*, *warm* und *heiß* aufgeteilt werden. In der klassischen Mengenlehre lässt sich diese Menge *kalt* über alle Temperaturen t durch die folgende Notation beschreiben:

$$kalt = \{t : t \leq 5\} \; ; \; t \in \mathbb{R},$$

analog für *warm* ($10 \leq t \leq 20$) und *heiß* ($25 \leq t$). Im Gegensatz zu Lufttemperaturen kann Wasser gewöhnlich erst bei Temperaturen über 40 °C als *heiß* bezeichnet werden. Damit ist diese Definitionen der Variablen Temperatur stark vom Kontext der Fragestellung abhängig. Weiterhin kann die charakteristische Funktion c_{kalt} definiert werden:

$$c_{kalt}(t) = \begin{cases} 1 & : \quad t \leq 5 \\ 0 & : \quad t > 5, \end{cases}$$

wobei alle Elemente $t \in \mathbb{R}$. Diese Funktion liefert 1 für alle Elemente t kleiner gleich 5, d. h., wenn diese zur Menge von kalten Temperaturen gehören, ansonsten 0. Die Funktionen für warme und heiße Temperaturen lassen sich analog definieren. Diese charakteristische Funktion kann auch als *Zugehörigkeitsfunktion* über die Menge von kalten Temperaturen angesehen werden.

Wird zum Beispiel eine Temperatur von knapp über 5 °C gemessen, liefert die klassische Definition für die Menge der kalten Temperaturen in diesem Fall 0. Das stimmt aber nicht unbedingt mit der Intuition für Temperaturen von beispielsweise 5.1 °C überein. Dieser scharfe Übergang von „kalt“ zu „nicht kalt“ entspricht somit nicht dem menschlichen Empfinden (Zimmermann, 1995). Die Idee besteht nun darin, eine unscharfe Funktion zu definieren, die genau diese Grenzen in einer Art beschreibt, dass Elemente zu einem Grad zwischen 0 und 1 zu einer Menge gehören. Für dieses

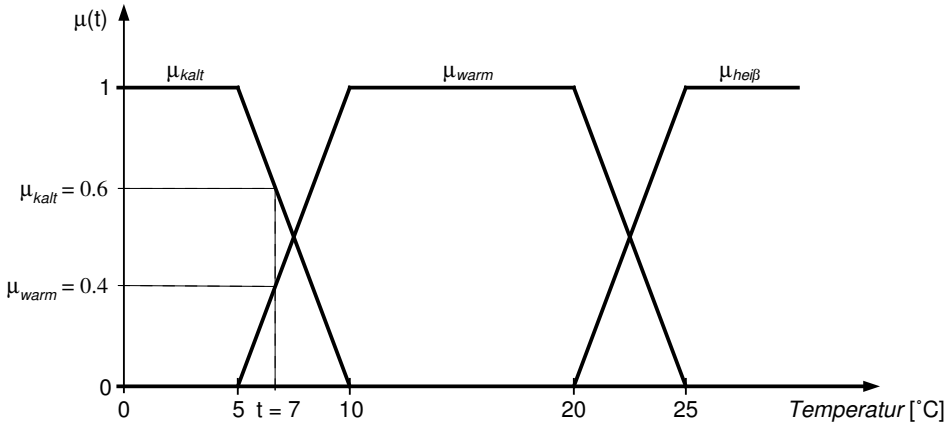


Abbildung 2.1: Die hier gezeigte Aufteilung des Attributes *Temperatur* in die drei linguistischen Terme *kalt*, *warm* und *heiß* geschieht mittels individueller Fuzzy-Zugehörigkeitsfunktionen. Das Beispiel zeigt für eine Temperatur von 7 °C einen Zugehörigkeitsgrad von 0.4 zur Menge μ_{kalt}, von 0.6 zu μ_{warm} und 0.0 zu $\mu_{heiß}$. Die Summe der Zugehörigkeitsgrade addieren sich für jedes Element t zu 1.

Beispiel kann man jetzt eine Fuzzy-Zugehörigkeitsfunktion definieren:

$$\mu_{kalt}(x) = \begin{cases} 1 & : \quad t \leq 5, \\ 1 - \frac{t-5}{5} & : \quad 5 < t \leq 10, \\ 0 & : \quad 10 < t. \end{cases}$$

Diese Funktion beschreibt Temperaturen zwischen 5 °C und 10 °C mit einem linear fallenden Zugehörigkeitsgrad. D. h., je näher der Grad der Zugehörigkeit an 5 °C, desto mehr gehört das Element zur Menge *kalt*; je mehr es sich an 10 °C annähert desto weniger. In diesem Beispiel können die drei Zugehörigkeitsfunktionen μ_{kalt}, μ_{warm} und $\mu_{heiß}$, wie in Abbildung 2.1 dargestellt werden, wobei die Aufteilung der Variablen *Temperatur* vom persönlichen Temperaturempfinden jedes Einzelnen abhängig ist.

Temperaturen von weniger als 5 °C können gewiss als *kalt* bezeichnet werden. Die Zugehörigkeitsfunktion fällt für Temperaturen bis 10 °C linear ab. Temperaturen zwischen 10 °C und 20 °C sind *warm* und über 25 °C *heiß*. Die Zugehörigkeitsfunktion für die Variable *warm* steigt in diesem Fall linear von 5 °C bis 10 °C an und fällt linear von 20 °C auf 25 °C, wobei für heiße Temperaturen die Funktion in diesem Bereich linear ansteigt. In diesem Beispiel addieren sich die Zugehörigkeitsfunktionen aller linguistischen Variablen für jeden Wert des Attributes *Temperatur* zu 1. Das kann durch

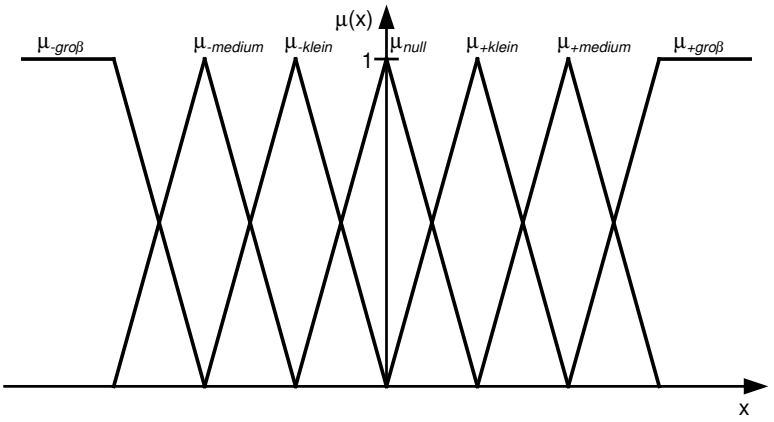


Abbildung 2.2: Beispiel einer Partitionierung durch sieben Fuzzy-Zugehörigkeitsfunktionen. Die fünf inneren Funktionen werden durch Dreieck- und die äußeren durch nach links- bzw. nach rechts-offene Trapezfunktionen beschrieben.

gleichmäßiges Überlappen der Funktionen erreicht werden, wodurch die eigentliche Unschärfe in den Daten ausgedrückt wird. Dies wird in der Fuzzy-Logik allerdings nicht unbedingt verlangt; in der Praxis aber oft so eingesetzt. Die Wahl der Zugehörigkeitsfunktion kann beliebig sein, solange sie sich im Intervall $[0, 1]$ bewegt und auf dem ganzen Wertebereich des Attributs definiert ist.

Wendet man eine Temperatur von 7 °C auf die linguistischen Terme an, erhält man als Ergebnis die drei Funktionswerte für $\mu_{kalt}(7) = 0.6$, $\mu_{warm}(7) = 0.4$ und $\mu_{heiß}(7) = 0.0$, wobei die Summe 1 ist. Eine Temperatur kann also durch die drei linguistischen Terme *kalt*, *warm* und *heiß* beschrieben werden, wobei die eigentliche Unschärfe durch die Überlappung der Zugehörigkeitsfunktionen entsteht. Diese Aufteilung wird auch als *Fuzzy-Granulierung* oder *Fuzzy-Partitionierung* bezeichnet; im Gegensatz zur klassischen Mengenlehre, die nur eine scharfe Aufteilung erlaubt. Es sei bemerkt, dass es sich hier nicht um Wahrscheinlichkeiten handelt, vielmehr hat man es mit einem Grad der Zugehörigkeit eines Elements zu einer Menge zu tun. Im Unterschied zur Wahrscheinlichkeitstheorie wird dadurch keine Wahrscheinlichkeit eines Ereignisses, sondern eine Art Unsicherheit mit der ein Element zu der Menge gehört, ausgedrückt.

Eine oft verwendete Fuzzy-Partitionierung zeigt Abbildung 2.2. Hier werden sieben Zugehörigkeitsfunktionen verwendet, um die Domäne x zu beschreiben. Die Funktionen lassen sich kennzeichnen durch $\mu_{-groß}$ (negativ groß), $\mu_{-medium}$ (negativ

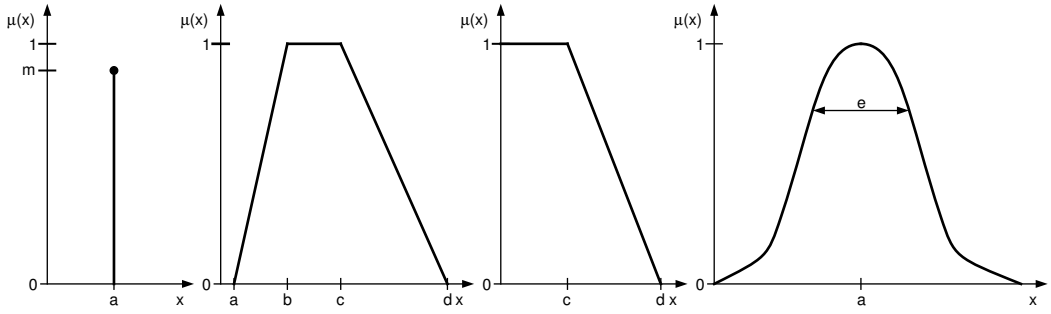


Abbildung 2.3: Oft verwendete Fuzzy-Zugehörigkeitsfunktionen – Singleton, Trapezfunktion, links-offene Trapezfunktion und Gauß-Funktion.

medium), μ_{-klein} (negativ klein), μ_{null} (Null) und analog für die positive Seite. Die Elemente der Partitionierung in unscharfe Cluster werden auch als *Fuzzy-Granules* bezeichnet.

Allgemein lässt sich eine Fuzzy-Menge A (engl. *fuzzy set*) als totale Abbildung $\mu_A : \mathcal{U} \rightarrow [0,1]$ vom Universum $\mathcal{U}$ (oder Grundmenge) mit den Elementen u in das abgeschlossene Intervall $[0,1]$ definieren:

$$A = \{(u, \mu_A(u)) \mid u \in \mathcal{U}, \mu_A(u) \in [0,1]\}$$

dabei gibt $\mu_A(u)$ den Zugehörigkeitsgrad eines Elements $u \in \mathcal{U}$ zur Menge A an. Diese Funktion nennt man Zugehörigkeitsfunktion (*membership function*) oder charakteristische Funktion (Nauck u. a., 1994).

Ausgehend von dieser Definition können weitere Formalisierungen auf Fuzzy-Mengen getroffen werden. Es gilt die *Gleichheit* $A = B$ zwischen zwei Fuzzy-Mengen A und B, wenn $\mu_A(u) = \mu_B(u)$ für alle $u \in \mathcal{U}$. Eine Fuzzy-Menge wird als *leere Menge* ($A = \emptyset$) bezeichnet, wenn $\mu_A(u) = 0$ für alle $u \in \mathcal{U}$ gilt. Man nennt A *Teilmenge* von B oder $A \subseteq B$ genau dann, wenn $\mu_A(u) \leq \mu_B(u)$ für alle $u \in \mathcal{U}$ ist.

Als Zugehörigkeitsfunktion sollte eine möglichst stetige Funktion gewählt werden, die in das geschlossene Intervall $[0,1]$ abbilden muss. Fuzzy-Mengen wie Singleton-, Dreieck-, Trapez- (rechts-offen, links-offen) und Gauß-Funktionen, die durch wenige Parameter beschrieben werden können, eignen sich besonders gut für die Speicherung in Computern und für die Durchführung von Berechnungen.

Mögliche Fuzzy-Zugehörigkeitsfunktionen sind in Abbildung 2.3 gezeigt (Nauck u. a., 1997). Die erste Darstellung stellt eine Singleton-Funktion dar, die für genau

einen Wert x einen Zugehörigkeitsgrad von m mit $0 < m \leq 1$ liefert. Die im nächsten Bild abgebildete Trapezfunktion lässt sich durch die vier Parameter $\langle a, b, c, d \rangle$ beschreiben, wobei b und c die Grenzen des Kernbereiches (oder *Core*) und a und d die Grenzen des Einflussbereiches (oder *Support*) spezifizieren. Diese Art von Funktion wird häufig verwendet, da sie sich leicht durch Expertenwissen modellieren lässt und umgangssprachlich interpretiert werden kann. Eine spezielle Trapezfunktion ist die Dreieck-Funktion, wobei $b = c$ ist. Die dritte Abbildung repräsentiert eine nach links-offene Trapezfunktion mit $a = b = -\infty$ oder $c = d = +\infty$ für die rechts-offene. Die letzte Darstellung zeigt die Gauß-Funktion mit den zwei Parametern a und e. Sie wird oft verwendet, da sie sich durch mathematische Eigenschaften, wie Stetigkeit und Differenzierbarkeit, auszeichnet. Diese Eigenschaften werden oft benötigt, um Regeln automatisch während eines Trainingsprozesses anzupassen, wie man es aus dem Gradientenabstiegsverfahren kennt (Poggio und Girosi, 1989). Fuzzy-Zugehörigkeitsfunktionen werden durch die folgenden Eigenschaften charakterisiert:

- *Support:* $s_A := \{x \mid \mu_A(x) > 0\}$, alle x des Wertebereiches von A mit $\mu_A(x) > 0$ (Zugehörigkeitsgrad größer Null)
- *Core:* $c_A := \{x \mid \mu_A(x) = 1\}$, alle x des Wertebereiches von A mit $\mu_A(x) = 1$ (maximaler Zugehörigkeitsgrad)
- *α-cut:* $A_\alpha := \{x \mid \mu_A(x) \geq \alpha\}$, alle x des Wertebereiches von A mit $\mu_A(x) \geq \alpha$ (Schnitt in Höhe α)
- *height:* $h_A := \max_x \{\mu_A(x)\}$, maximaler Funktionswert für $\mu_A(x)$.

Im Folgenden wird der *Support* auch als Einflussbereich (Zugehörigkeitsgrad größer Null) und der *Core* als Kernbereich (Zugehörigkeitsgrad gleich Eins) einer Fuzzy-Zugehörigkeitsfunktion $\mu_A(x)$ bezeichnet. Der *α-cut* wird häufig für die Inferenz auf Fuzzy-Mengen und die maximale Höhe *height* zur Defuzzifizierung benötigt.

Operationen auf Fuzzy-Mengen

In diesem Abschnitt werden Operationen – Komplement, Vereinigung und Durchschnitt – auf Fuzzy-Mengen definiert. Diese Operationen lassen sich aus der klassischen Mengenlehre ableiten, sind aber für Fuzzy-Mengen nicht eindeutig bestimmt und bilden keine Boolesche Algebra.

Fuzzy-Komplement

Das Komplement ist eine Abbildung, bei der Zugehörigkeitsgrade auf Zugehörigkeitsgrade abgebildet werden:

$$\bar{A} = \{(u, (1 - \mu_A(u))) \mid u \in \mathcal{U}\},$$

allgemein ist dieses eine Abbildung $c : [0,1] \mapsto [0,1]$. Je mehr ein Element zur Menge A gehört, desto weniger gehört es zur Menge $\bar{A}$ und umgekehrt, siehe Abbildung 2.4. Die zweite Bedingung ist das Monotoniekriterium, das besagt, dass c eine monoton fallende Funktion ist, wenn für alle $a, b \in [0,1]$ mit $a < b$ folgt, dass $c(a) \geq c(b)$. Alle Funktionen, die die genannten Bedingungen erfüllen, bilden die Klasse der Komplementabbildungen für Fuzzy-Mengen.

In den meisten Anwendungen werden an diese Funktionen noch zwei weitere Bedingungen gestellt. Erstens, dass sie stetig sind und zweitens, dass sie involutiv sind, d. h. für alle $a \in [0,1]$ gilt $c\,(c(a)) = a$. Ein Beispiel einer Komplementklasse stellt die nach Takagi und Sugeno (1985) bekannte Sugeno-Klasse dar, die durch:

$$c_\lambda(a) = \frac{1-a}{1+\lambda a} \quad \text{mit} \quad \lambda \in \,]-1, \infty[,$$

definiert ist. Hierbei ergibt sich gerade für $\lambda = 0$ das Zadeh'sche Fuzzy-Komplement.

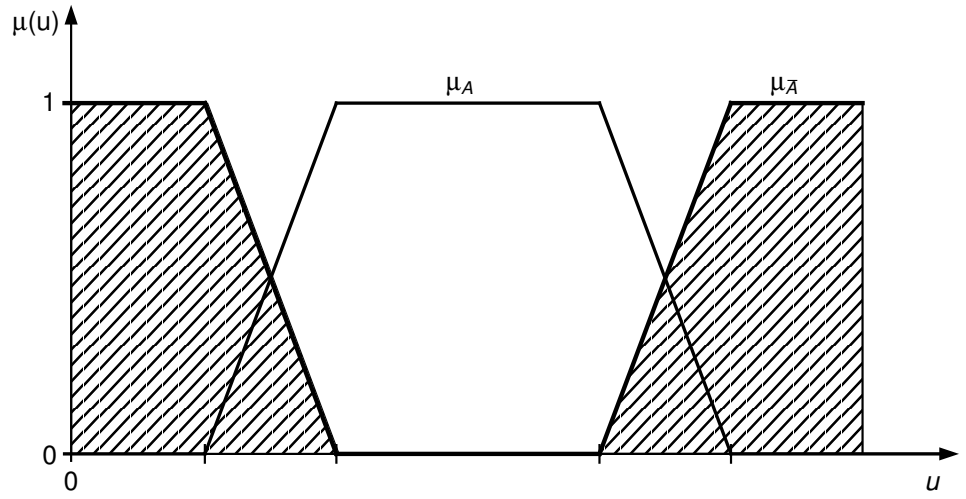


Abbildung 2.4: Komplementäre Fuzzy-Menge $\mu_{\bar{A}}(u)$ von $\mu_A(u)$, wobei $\mu_{\bar{A}}(u) = 1 - \mu_A(u)$ gilt.

Eine weitere Klasse von Komplementen stellt die nach Yager u. a. (1987) benannte Yager-Klasse dar, die durch:

$$c_\omega(a) = (1 - a^\omega)^{\frac{1}{\omega}} \quad \text{mit} \quad \omega \in \ ]0, \infty[$$

charakterisiert ist und das Zadeh'sche Fuzzy-Komplement für $\omega = 1$ liefert.

Fuzzy-Durchschnitt

Nach Zadeh ist der Durchschnitt oder die Konjunktion auf Fuzzy-Mengen als Abbildung $\top : [0,1] \times [0,1] \mapsto [0,1]$ definiert. Diese Abbildung nennt man auch $\top$-Norm (T-Norm), wobei $\top$ eine monoton nicht-fallende Funktion in beiden Argumenten ist und es gilt $\top(a,b) = \top(b,a) \leq \top(c,d) = \top(d,c)$ falls $a \leq b$ und $c \leq d$ mit $a, b, c, d \in [0,1]$. Die von Zadeh definierte Operation sieht wie folgt aus:

$$A \cap B = \{u, \min\{\mu_A(u), \mu_B(u)\} \mid u \in \mathcal{U}\}.$$

Alle T-Normen müssen den folgenden Bedingungen genügen:

Nullelement	$\top(a,0) = \top(0,a) = 0,$
Einheitselement	$\top(a,1) = \top(1,a) = a,$
Monotonie	$a \leq b \Rightarrow \top(a,c) \leq \top(b,c),$
Kommutativität	$\top(a,b) = \top(b,a),$
Assoziativität	$\top(a, \top(b,c)) = \top(\top(a,b), c),$

für alle $a, b, c \in [0,1]$. Damit gelten die Gesetze der Identität, Monotonie, Kommutativität und Assoziativität der klassischen Mengen auch für Fuzzy-Mengen unter der Durchschnittsbildung. Zusätzlich zu diesen Bedingungen ist es manchmal nützlich zu fordern, dass die Abbildung der $\top$-Norm stetig ist. Verlangt man zusätzlich, dass die Funktion idempotent ist, d. h., für alle $a \in [0,1]$ gilt $\top(a,a) = a$, erhält man die obere Grenzfunktion der $\top$-Norm. Klir und Yuan (1996) und Klir und Folger (1987) bezeichnen sie auch als optimistischen Durchschnittsoperator. Die Abbildung 2.5 zeigt die Minimumbildung der beiden Zugehörigkeitsfunktion $\mu_A(u)$ und $\mu_B(u)$. Die resultierende Fuzzy-Zugehörigkeitsfunktion kann dann geschrieben werden als $\mu_{A \cap B}(x) = \mu_A \wedge \mu_B$.

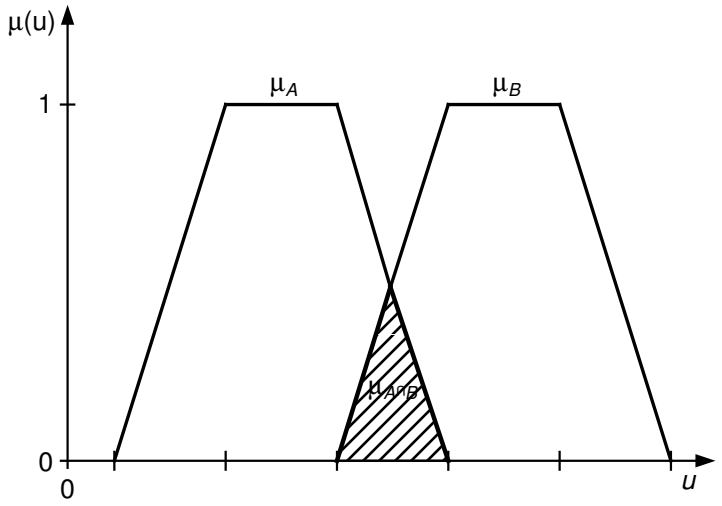


Abbildung 2.5: Fuzzy-Durchschnitt $\mu_{A\cap B}(u)$ von $\mu_A(u)$ und $\mu_B(u)$, wobei $\mu_{A\cap B}(u) = \min\{\mu_A(x), \mu_B(x)\} = 1 - \max\{1-\mu_A(x), 1-\mu_B(x)\}$ gilt.

Weitere Bespiele für parametrisierte $\top$-Normen sind die von Schweizer und Sklar (1960) vorgestellte Familie:

$$\top_p(a,b) = \max\{0, a^{-p} + b^{-p} - 1\}^{-\frac{1}{p}} \quad \text{mit} \quad p \in \,]-\infty, \infty[\,\backslash\{0\}.$$

Diese Funktion ist für $p = 0$ nicht definiert und liefert für $p = -1$ die von Zadeh definierte Minimum-Norm. Von Yager (1980) wurde die Familie der Yager-Normen veröffentlicht:

$$\top_\omega(a,b) = 1 - \min\left\{1, ((1-a)^\omega + (1-b)^\omega)^{\frac{1}{\omega}}\right\} \quad \text{mit} \quad \omega \in \,]0, \infty[.$$

Diese Funktion ist nur für positive ω definiert und ergibt für $\omega = 1$ die von Łukasiewicz (1970) vorgeschlagene Łukasiewicz-Norm:

$$\top(a,b) = \max\{0, a + b - 1\}.$$

Eine weitere wichtige Norm ist die Produkt-Norm, da diese sich durch wichtige Eigenschaften wie Stetigkeit und Differenzierbarkeit auszeichnet:

$$\top(a,b) = a \cdot b.$$

Fuzzy-Vereinigung

Wie auch für den Durchschnittsoperator, ist die Vereinigung oder Disjunktion auf Fuzzy-Mengen als Abbildung $\perp : [0,1] \times [0,1] \mapsto [0,1]$ definiert. Diese Abbildung nennt man auch $\perp$-Norm (TCo- oder S-Norm), wobei die $\perp$-Norm eine monoton nicht-fallende Funktion in beiden Argumenten ist und es gilt $\perp(a,b) = \perp(b,a) \leq \perp(c,d) = \perp(d,c)$ falls $a \leq b$ und $c \leq d$ mit $a,b,c,d \in [0,1]$. Die von Zadeh definierte Operation sieht wie folgt aus:

$$A \cup B = \{u, \max\{\mu_A(u), \mu_B(u)\} \mid u \in \mathcal{U}\}.$$

Alle Normen dieser Art müssen den folgenden Bedingungen genügen:

Nullelement	$\perp(a,1) = \perp(1,a) = 1,$
Einheitselement	$\perp(a,0) = \perp(0,a) = a,$
Monotonie	$a \leq b \Rightarrow \perp(a,c) \leq \perp(b,c),$
Kommutativität	$\perp(a,b) = \perp(b,a),$
Assoziativität	$\perp(a, \perp(b,c)) = \perp(\perp(a,b), c)$

für alle $a,b,c \in [0,1]$. Damit gelten die Gesetze der Identität, Monotonie, Kommutativität und Assoziativität der klassischen Mengen auch für Fuzzy-Mengen unter der Vereinigungsbildung. Zusätzlich zu diesen Bedingungen ist es manchmal nützlich zu fordern, dass die Abbildung der $\perp$-Norm stetig ist. Verlangt man zusätzlich, dass die Funktion idempotent ist, d. h., für alle $a \in [0,1]$ gilt $\perp(a,a) = a$, erhält man die untere Grenzfunktion der $\perp$-Norm. Klir und Yuan (1996) und Klir und Folger (1987) bezeichnen sie deshalb auch als pessimistischen Vereinigungsoperator. Abbildung 2.6 zeigt die Maximumbildung der beiden Zugehörigkeitsfunktionen $\mu_A(u)$ und $\mu_B(u)$. Die resultierende Fuzzy-Zugehörigkeitsfunktion kann dann geschrieben werden als $\mu_{A \cup B}(x) = \mu_A \vee \mu_B$.

Das bedeutet, dass dieser Operator gerade den Grenzoperator der $\perp$-Norm darstellt. Weitere Bespiele für parametrisierte $\perp$-Normen sind die von Schweizer und Sklar (1960) vorgestellte Familie:

$$\perp_p(a,b) = 1 - \max\{0, (1-a)^{-p} + (1-b)^{-p} - 1\}^{-\frac{1}{p}} \quad \text{mit} \quad p \in \,]-\infty, \infty[\setminus\{0\}.$$

Diese Funktion ist für $p = 0$ nicht definiert und liefert für $p = -1$ die von Zadeh defi-

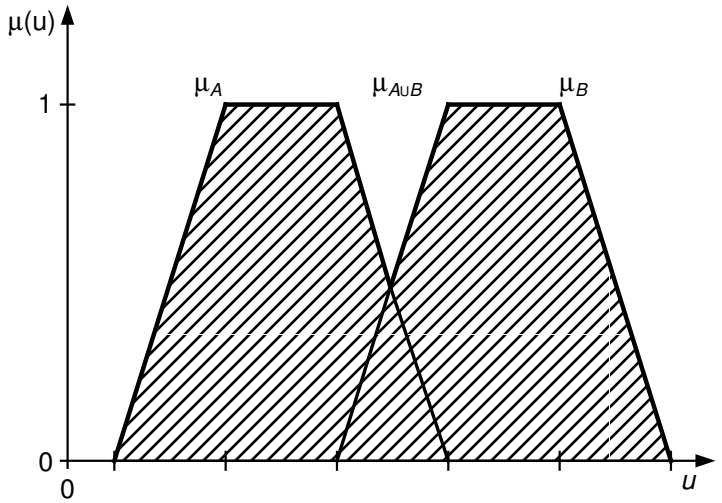


Abbildung 2.6: Fuzzy-Vereinigung $\mu_{A\cup B}(u)$ von $\mu_A(u)$ und $\mu_B(u)$, wobei $\mu_{A\cup B}(u) = \max\{\mu_A(x), \mu_B(x)\} = 1 - \min\{1-\mu_A(x), 1-\mu_B(x)\}$ gilt.

nierte Maximum-Norm. Yager (1980) hat die Familie der Yager-Normen veröffentlicht:

$$\perp_\omega (a,b) = \min\left\{1, (a^\omega + b^\omega)^{\frac{1}{\omega}}\right\} \quad \text{mit} \quad \omega \in \ ]0,\infty[.$$

Diese Funktion ist nur für positive ω definiert und ergibt für $\omega = 1$ die von Łukasiewicz (1970) vorgestellte Łukasiewicz-Norm:

$$\perp (a,b) = \min\{1, a+b\}.$$

Eine weitere wichtige Norm ist die Produkt-Norm, da diese sich durch Eigenschaften wie Stetigkeit und Differenzierbarkeit auszeichnet:

$$\perp (a,b) = a + b - a \cdot b.$$

Die weiteren Untersuchungen basieren auf den fünf Normen: Minimum/Maximum-Norm (oder kurz *Min/Max*), Produkt-Norm (oder kurz *Prod*), Yager-Norm mit $p = 2^{-1}$ (oder kurz $Yager_{1/2}$), Łukasiewicz-Norm (oder kurz *Luka*) und Yager-Norm mit $p = 2$ (oder kurz $Yager_2$), die in Kapitel 3.3 ausführlich evaluiert werden.

Fuzzy-Regeln und Fuzzy-Regelsysteme

In Expertensystemen wird Wissen zumeist in Form von Regeln gespeichert. Diese Regeln bestehen aus Vorbedingungen, die entweder wahr oder falsch sein können, und Folgerungen, die abhängig von den entsprechenden Vorbedingungen ausgeführt werden oder nicht. Experten formulieren ihre Regeln aber oft in einer Art, in der nicht genau festgelegt ist, wann eine Vorbedingung erfüllt ist, wie zum Beispiel „Wenn die Temperatur *kalt* ist, dann ist das Unfallrisiko *groß*“. Menschen sind in der Lage, diese Regel zu verstehen und können danach handeln. Für Computer stellen jedoch die unscharfen Begriffe *kalt* oder *groß* ein Problem dar. Ab wann soll ein Expertensystem eine Temperatur als kalt interpretieren? Doch zugleich zeigt sich, dass viele komplexe Probleme durch die Verwendung von unscharfen Regeln einfacher beschrieben werden können. Das Ziel ist es, dem Computer das Umgehen mit unscharfen Begriffen zu ermöglichen. Dafür können Fuzzy-Regeln verwendet werden, um das unscharfe Wissen zwischen den Variablen abzubilden:

$$\texttt{WENN}\ \textit{Temperatur}\ \texttt{IST}\ \textit{kalt}\ \texttt{DANN}\ \textit{Unfallrisiko}\ \texttt{IST}\ \textit{groß}.$$

Dieses Beispiel illustriert, wie man mithilfe der linguistischen Variablen *kalt* für die *Temperatur* und *groß* für das *Unfallrisiko* einen unscharfen oder vagen Zusammenhang ausdrücken kann, ohne direkt eine genaue Temperatur zu nennen oder das Unfallrisiko genauer zu beschreiben. Deshalb sind Fuzzy-Regeln immer dort interessant, wo präzise Formulierungen nicht erforderlich sind. Hierdurch kann eine höhere Interpretierbarkeit erreicht werden. Damit ist der Ausdruck „*Temperatur* `IST` *kalt*“ eben nicht nur wahr oder falsch, sondern kann auch zwischen wahr und falsch liegen, was sich durch die Zugehörigkeiten von *Temperatur* zu der Menge μ_{kalt} ausdrücken lässt. Allgemein kann eine Fuzzy-Regel R wie folgt definiert werden:

$$R : \texttt{IF}\ x_1\ \texttt{IS}\ A_1\ \texttt{AND}\ \cdots\ \texttt{AND}\ x_n\ \texttt{IS}\ A_n\ \texttt{THEN}\ y\ \texttt{IS}\ B,$$

wobei A_i die Vorbedingungen der linguistischen Variablen des Eingabevektor $\vec{x}$ sind und B die Folgerung der Ausgabe y ist. Diese Arten von Fuzzy-Regeln nennt man *Mamdani-Regeln* (Mamdani und Assilian, 1975). Eine Möglichkeit diese Art von Regeln zu gewinnen, ist die Erklärung durch einen Experten, der sein Wissen in umgangssprachlichen Formulierungen ausdrückt. Dieses Wissen wird dann durch linguistische Variablen formuliert, die durch Fuzzy-Zugehörigkeitsfunktionen beschrieben werden.

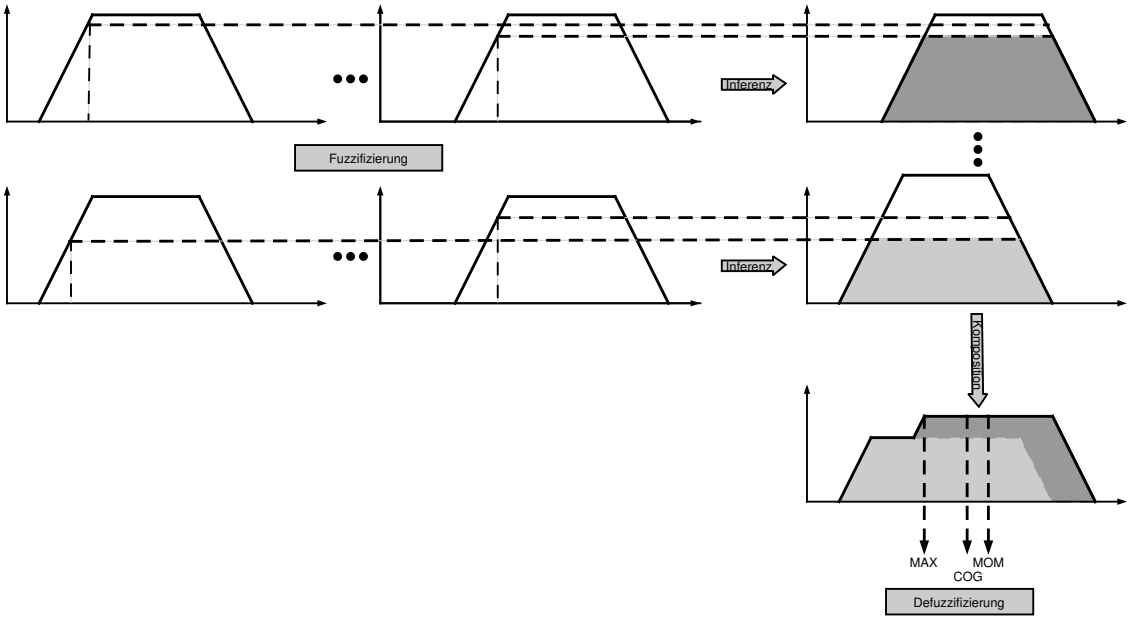


Abbildung 2.7: Beispiel eines Fuzzy-Regelsystems nach Mamdani mit zwei Regeln und zwei Merkmalen. Hierbei werden die scharfen Eingaben auf Fuzzy-Mengen abgebildet (Fuzzifizierung). Im nächsten Schritt werden diese Mengen mithilfe der $\top$-Norm verknüpft (Inferenz). Anschließend werden die unscharfen Ausgaben jeder Regel überlagert (Komposition), um im letzten Schritt mithilfe der $\bot$-Norm eine scharfe Ausgabe zu generieren (Defuzzifizierung).

Eine vollständige Regelbasis sieht dann wie folgt aus:

$$\begin{array}{lllllll} R_1: & \texttt{IF} & x_1\ \texttt{IS}\ A_{1,1} & \texttt{AND}\cdots\texttt{AND} & x_n\ \texttt{IS}\ A_{n,1} & \texttt{THEN} & y_1\ \texttt{IS}\ B_1 \\ & & \vdots & & \vdots & & \vdots \\ R_m: & \texttt{IF} & x_1\ \texttt{IS}\ A_{1,l_1} & \texttt{AND}\cdots\texttt{AND} & x_n\ \texttt{IS}\ A_{n,l_n} & \texttt{THEN} & y_m\ \texttt{IS}\ B_{l_m} \end{array}$$

mit m ($1 \leq j \leq m$) für die Anzahl der Regeln und mit l_i ($1 \leq i \leq n$) der Anzahl der linguistischen Variablen. Möchte man nun in diesem Regelsystem schließen (siehe Abbildung 2.7), so müssen im ersten Schritt die Fuzzy-Zugehörigkeitsgrade $\mu_{A_{1,l_i}}$ bis $\mu_{A_{n,l_i}}$ für alle Regeln R_j des Eingabevektors $\vec{x}$ berechnet werden.

Durch die Abbildung scharfer Eingabewerte auf die Fuzzy-Mengen erhält man die einzelnen Zugehörigkeitsgrade der Vorbedingungen (*Fuzzifizierung*), die auf den linguistischen Wert der Folgerung der Regel abgebildet werden. Diesen Schritt nennt man *Inferenz*. Diese erfolgt, indem man eine $\top$-Norm (siehe Abschnitt 2.3 Fuzzy-

Durchschnitt) anwendet, die den Grad der Zugehörigkeit für die Folgerung liefert. Grafisch kann es als „Abschneiden“ der Zugehörigkeitsfunktion in Höhe des Grades der Vorbedingung gesehen werden. Da in einem Regelsystem meist mehr als nur eine Regel vorhanden ist, müssen die Zugehörigkeitsgrade der Folgerungen wiederum verknüpft werden, was durch die $\perp$-Norm (siehe Abschnitt 2.3 Fuzzy-Vereinigung) geschieht. Man nennt dies auch *Komposition*. Im letzten Schritt muss – falls notwendig – eine konkrete Ausgangsgröße aus der überlagerten Ergebnisfunktion ermittelt werden. Diesen Vorgang, bei dem eine Fuzzy-Menge in eine scharfe Ausgabe abgebildet wird, nennt man *Defuzzifizierung*. Eine mögliche Defuzzifizierung ist die Schwerpunktmethode (COG), die den Schwerpunkt der Fläche unter der Zugehörigkeitsfunktion berechnet. Der Nachteil dieser Methode ist, dass dieses Verfahren aus Sicht der Fuzzy-Mengentheorie kaum zu rechtfertigen und sehr aufwendig zu berechnen ist (Kruse u. a., 1993). Zwei weitere häufig verwendete Ansätze sind die Maximum-Kriterium Methode (MAX), die einen beliebigen Wert der maximalen Zugehörigkeitsgrade auswählt; während die Mittelwerte-der-Maxima Methode (MOM), die Werte aller maximalen Zugehörigkeitsgrade mittelt.

Eine Modifikation des Mamdani-Reglers stellt der *Takagi-Sugeno*-Regler dar (Takagi und Sugeno, 1985) dar, der im Anweisungsteil einer Regel eine scharfe Ausgabe berechnet. Der Experte hat hier die Aufgabe, das gewünschte Regelverhalten des Anweisungsteils in expliziter mathematischer Form anzugeben. Die Fuzzy-Regeln werden in der Form:

$$R_j \ : \ \texttt{IF}\ x_1\ \texttt{IS}\ A_{1,l_i}\ \texttt{AND}\cdots\texttt{AND}\ x_n\ \texttt{IS}\ A_{n,l_n}\ \texttt{THEN}\ y_j = f_j(\vec{x})$$

geschrieben, wobei die Eingabe $\vec{x}$ als numerischer Wert vorliegen muss und die Ausgabe nicht „fuzzy“ ist. Als Ausgabefunktion kann jede beliebige Funktion gewählt werden. Häufig verwendet man eine lineare Funktion, die als $f_j(\vec{x}) = C_{0,j} + C_{1,j} \cdot x_1 + \ldots + C_{n,j} \cdot x_n$ geschrieben werden kann. Jede Regel liefert dann als Ergebnis des Bedingungsteiles den Erfüllungsgrad:

$$\omega_j = \min\left\{\mu_{A_{1,l_i}}(x_1), \ldots, \mu_{A_{n,l_n}}(x_n)\right\}.$$

Die Auswertung aller Regel R_j mit $1 \leq j \leq r$ geschieht ähnlich zum Mamdani-Regler. Der Regler übernimmt jetzt die Aufgabe, die Grenzen zwischen den lokalen (linearen)

Interpolationen zu approximieren. Das Gesamtergebnis ergibt sich dann aus:

$$y = \frac{\sum_{j=1}^{m} \omega_j \, y_j}{\sum_{j=1}^{m} \omega_j}.$$

Ein Nachteil dieses Verfahrens ist, dass die Bestimmung der Funktion y_j für jede Regel j gegebenenfalls sehr aufwendig sein kann. Ein weiterer Nachteil ist die schwierige Interpretierbarkeit der Ausgabe des Regelsystems, da jede Folgerung scharfe Funktionswerte liefert, die durch die Funktion y verknüpft werden. Diese Art von Regler werden oft für Anwendungen im Bereich der technischen Steuerung (Fuzzy-Control) verwendet, da auf die Defuzzifizierung verzichtet werden kann.

Klassifikation in Fuzzy-Regelsystemen

Die gelernten Fuzzy-Prototypen können als unabhängige Fuzzy-Regeln gesehen werden. Diese basieren auf einer lokalen Granulierung des Eingaberaumes, sodass sich das folgende System von unabhängigen Regeln ergibt:

$$\begin{array}{lllllll}
R_1^1: & \texttt{IF} & x_1 \,\texttt{IS}\, \mu_{1,1}^1 & \texttt{AND} \cdots \texttt{AND} & x_n \,\texttt{IS}\, \mu_{n,1}^1 & \texttt{THEN} & \texttt{class}\, 1 \\
 & & \vdots & & \vdots & & \vdots \\
R_{r_1}^1: & \texttt{IF} & x_1 \,\texttt{IS}\, \mu_{1,r_1}^1 & \texttt{AND} \cdots \texttt{AND} & x_n \,\texttt{IS}\, \mu_{n,r_1}^1 & \texttt{THEN} & \texttt{class}\, 1 \\
 & & \vdots & & \vdots & & \vdots \\
R_j^k: & \texttt{IF} & x_1 \,\texttt{IS}\, \mu_{1,j}^k & \texttt{AND} \cdots \texttt{AND} & x_n \,\texttt{IS}\, \mu_{n,j}^k & \texttt{THEN} & \texttt{class}\, j \\
 & & \vdots & & \vdots & & \vdots \\
R_{r_c}^c: & \texttt{IF} & x_1 \,\texttt{IS}\, \mu_{1,r_c}^c & \texttt{AND} \cdots \texttt{AND} & x_n \,\texttt{IS}\, \mu_{n,r_c}^c & \texttt{THEN} & \texttt{class}\, c
\end{array}$$

hierbei ist R_j^k die Regel j mit der Klasse k. Die Regelbasis enthält Regeln für c Klassen, in der r_k die Anzahl der Regeln für eine Klasse k angibt ($1 \leq j \leq r_k$ und $1 \leq k \leq c$).

Die Fuzzy-Mengen $\mu_{i,j}^k : \mathbb{R} \mapsto [0,1]$ sind in jeder Dimension i ($1 \leq i \leq n$) definiert. Der zusammengefasste Zugehörigkeitsgrad einer Regel mit einem Muster $\vec{x} = (x_1, \ldots, x_n)$ berechnet sich aus dem Minimum-Operator als Fuzzy-`AND`:

$$\mu_j^k(\vec{x}) = \min_{i=1,\cdots,n} \left\{ \mu_{i,j}^k(x_i) \right\}.$$

Der resultierende Grad der Aktivierung über alle Regeln der Klasse k ergibt sich durch den Maximum-Operator als Fuzzy-OR:

$$\mu^k(\vec{x}) = \max_{j=1,\cdots,r_k} \left\{ \mu_j^k(\vec{x}) \right\}.$$

Aus diesen Fuzzy-Zugehörigkeitsgraden lässt sich die beste („best“) Klasse k_{best} für ein Eingabemuster $\vec{x}$ vorhersagen:

$$k_{\text{best}}(\vec{x}) = \arg\max_k \left\{ \mu^k(\vec{x}) \mid 1 \leq k \leq c \right\}.$$

Damit ergibt sich die finale Klassenausgabe für ein konkretes Eingabemuster. Durch die Verwendung von trapezförmigen Fuzzy-Zugehörigkeitsfunktionen für den Trainingsalgorithmus werden nicht alle Dimensionen einer Fuzzy-Regel eingeschränkt. Damit bleiben diese Regeln auch in hochdimensionalen Räumen interpretierbar. Im nächsten Abschnitt werden Verfahren diskutiert, die Fuzzy-Regelsysteme für die Klassifikation von unbekannten Daten automatisch aus Trainingsdaten lernen.

2.4 Lernen von Fuzzy-Regelmodellen

Dieser Abschnitt stellt ein Verfahren zur Erzeugung von Fuzzy-Regeln vor, welches auf Basis eines Probabilistischen Neuronalen Netzes (PNN) ein Fuzzy-Regelmodell konstruiert (Berthold, 2003). Das Netz basiert auf rechteckigen Fuzzy-Regeln als Übertragungsfunktionen. Hierbei werden die Neuronen der versteckten Schicht des klassischen PNN durch einfache Regeln ersetzt, die jeweils in einem lokalen, rechteckigen Bereich des Merkmalsraums mit einer von Null verschiedenen Ausgabe reagieren.

Als Trainingsalgorithmus wird der Dynamic Decay Adjustment Algorithmus (kurz DDA-Algorithmus), siehe Huber und Berthold (1995); Berthold (1997), verwendet. Dieser arbeitet basierend auf einem Probabilistischen Neuronalen Netz (PNN, Specht (1990)) unter Verwendung von rechteckigen Fuzzy-Zugehörigkeitsfunktionen. Der Algorithmus zeichnet sich durch einfache Bedienbarkeit und gute Interpretierbarkeit der Fuzzy-Regelbasis aus.

Die Verwendung von Fuzzy-Regeln hält die Anzahl der Regeln gering, da sie durch ihre Unschärfe einen größeren Merkmalsbereich abdecken können. Deshalb sind Fuzzy-Regeln unempfindlicher gegenüber verrauschten Daten. Ein weiterer Vorteil der Verwendung einer Regelstruktur ist die bessere Interpretierbarkeit des Netzes.

Klassische PNNs approximieren eine Wahrscheinlichkeitsverteilung mithilfe der Normalverteilung und können deshalb nicht direkt oder nur schwer interpretiert werden. Dieses Verfahren kann aufgrund der Verwendung von Fuzzy-Regeln auch als ein Fuzzy-Possibilistisches Neuronales Netz interpretiert werden, da durch die unscharfe Ausgabe der Zugehörigkeitsgrade eine Möglichkeitsverteilung über die Eingabe modelliert wird.

Trainingsalgorithmus für Fuzzy-Regeln

Der Dynamic Decay Adjustment Algorithmus ist optimal, um ein schnelles und konstruktives Training von Fuzzy-Regelbasen durchzuführen. Der Benutzer muss kein a-priori-Wissen einbringen, um die Struktur des Klassifikators festzulegen. Der Klassifikator wird vielmehr während des Trainings sukzessive aufgebaut und angepasst. Dieser Algorithmus konstruiert Netzwerke, die in ihrer Struktur ähnlich den Probabilistischen Neuronalen Netzen sind, wie sie Specht (1990) verwendet. Das Verfahren basiert auf der Restricted Coulomb Energy (RCE) Verfahren (Reilly u. a., 1982), das in Analogie zum physikalischen Verhalten geladener Teilchen, Potentialtöpfe erzeugt, in dessen Wirkbereich Probeladungen liegen. Diese Teilchen werden von den Potentialtöpfen gleicher Ladung (Klasse) nach dem *„Winner takes all“*-Prinzip angezogen.

Im Gegensatz zu den klassischen PNNs sind die vom DDA erzeugten Netze heteroskedastisch (variable Varianzen der Prototypen), da jeder Prototyp einen individuellen Radius besitzt. Dagegen nennt man Netze, bei denen jeder Prototyp den gleichen Radius besitzt, homoskedastischen (gleiche Varianzen der Prototypen). Des Weiteren werden nicht mehr alle Trainingsmuster als Neuronen gespeichert. Vielmehr geschieht die Generierung und Auswahl der Prototypen während des Trainings, indem alle Muster dem Netz iterativ präsentiert werden. Zu Beginn des Trainings sind in der versteckten Schicht keine Neuronen vorhanden. Es werden auch nur dann neue Neuronen hinzugefügt, wenn sie für die Abdeckung eines Musters benötigt werden. Dabei arbeitet dieser Algorithmus spezialisierend, d. h., neu eingefügte Neuronen überdecken zu Beginn große Bereiche des Merkmalsraums. Die Einflussbereiche existierender Neuronen werden im Laufe des Trainings nur verkleinert, niemals vergrößert. Das hat den Vorteil, dass die Terminierung des Algorithmus nachgewiesen werden kann, da Konflikte nicht zyklisch auftreten können (Berthold, 1997).

Abbildung 2.8 zeigt ein Probabilistisches Neuronales Netz wie es vom DDA er-

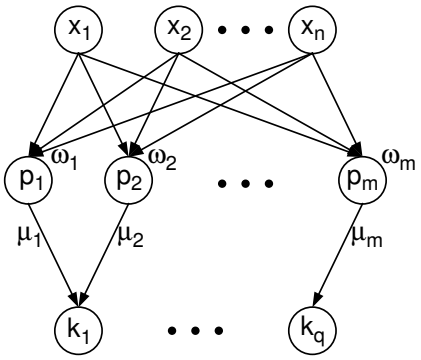


Abbildung 2.8: Dieses Netz berechnet in der verdeckten Schicht für ein Eingabemuster $\vec{x}$ aus der Eingabeschicht die Aktivierung (Fuzzy-Durchschnitt) mithilfe der Fuzzy-Zugehörigkeitsfunktion μ. Jedes Neuron in der verdeckten Schicht ist mit genau einem Neuron in der Ausgabeschicht verbunden. Dieses Neuron berechnet die Zugehörigkeit zur Klasse k (Fuzzy-Vereinigung).

zeugt wird. Diese Netze können als vorwärts gekoppelte Systeme mit einer verdeckten Schicht gesehen werden, die vollständig mit der Eingabeschicht verbunden sind. Dabei werden in der verdeckten Schicht keine Prototypen verwendet, wie beim klassischen PNN, sondern Fuzzy-Regeln als Übertragungsfunktionen, die die Aktivierung der Eingabe in eine Klassenausgabe übersetzen.

Die Regeln, wie sie im Folgenden verwendet werden, können im zweidimensionalen Raum, wie in Abbildung 2.9 zu sehen, verstanden werden. Diese Art von Funktionen lassen sich durch die vier Parameter $\langle a, b, c, d \rangle$ beschreiben. Den Kernbereich bildet hierbei das geschlossene Intervall $[b, c]$ mit einer Aktivierung von Eins (Kernbereich). Der Einflussbereich wird durch das offene Intervall (a, d) dargestellt, wobei die Aktivierung linear gegen Null an den Rändern a und d abfällt (Einflussbereich). Außerhalb der Grenzen a und d ist die Aktivierung Null.

Jeder Prototyp p besitzt eine Fuzzy-Zugehörigkeitsfunktion μ, die die Aktivierung (Fuzzy-Durchschnitt) zur Klasse k berechnet, und einen Zentrumsvektor $\vec{v}$, der durch das Eingabemuster $\vec{x}$ initialisiert wird. Jedes Neuron in der verdeckten Schicht ist mit genau einem Neuron in der Ausgabeschicht verbunden, in der anschließend die Ausgabe (Fuzzy-Vereinigung) berechnet wird.

Wie in Abbildung 2.10 dargestellt, wird in jedem Durchlauf genau ein Datenpunkt unabhängig von den anderen Eingabemustern behandelt. Die folgenden Fälle werden bei der Anpassung der Netzstruktur unterschieden:

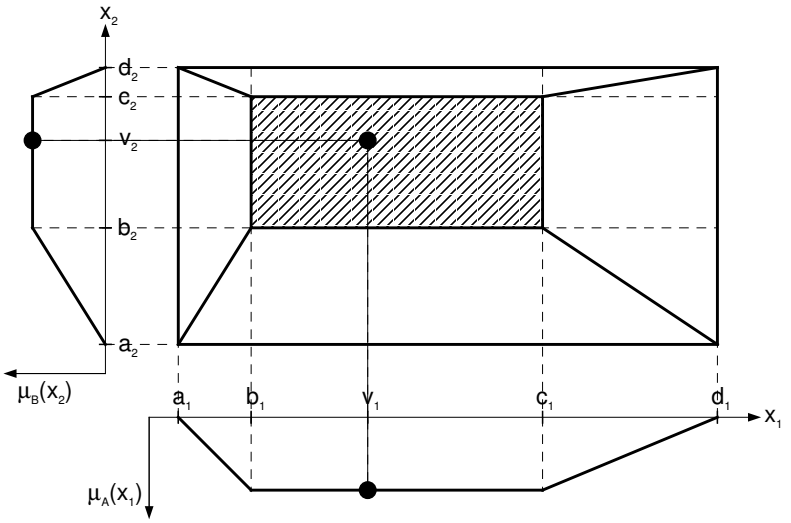


Abbildung 2.9: Zweidimensionale Fuzzy-Regeln mit den Attributen x_1 und x_2, und den jeweiligen Trapez-Zugehörigkeitsfunktionen $\mu_A(x_1)$ (mit $\langle a_1, b_1, c_1, d_1 \rangle$) und $\mu_B(x_2)$ (mit $\langle a_2, b_2, c_2, d_2 \rangle$). Der Vektor $\vec{v}$ stellt den Initialvektor der Regel dar.

cover Liegt das aktuelle Trainingsmuster innerhalb des Einflussbereiches eines existierenden Prototyps der gleichen Klasse, wird der Kernbereich entsprechend dieses Eingabemusters vergrößert, um das Eingabemuster abzudecken. Außerdem wird das zugehörige Gewicht um Eins inkrementiert und es muss eine zusätzliche Epoche nach Präsentation aller Eingabemuster gestartet werden. Für alle Prototypen p_j^k gilt damit:

$$\begin{aligned} \omega_j^k &= \omega_j^k + 1 \\ \vec{b}_j^k &= \left(\min\left\{b_{j,1}^k, x_{i,1}^k\right\}, \cdots, \min\left\{b_{j,n}^k, x_{i,n}^k\right\}\right)^T \\ \vec{c}_j^k &= \left(\max\left\{c_{j,1}^k, x_{i,1}^k\right\}, \cdots, \max\left\{c_{j,n}^k, x_{i,n}^k\right\}\right)^T \end{aligned}$$

commit Wird das aktuelle Muster durch keinen Prototypen der gleichen Klasse erklärt, muss ein neuer Prototyp eingefügt werden, der durch das Eingabemuster und dessen Klasse initialisiert wird. Zu Beginn wird der Einflussbereich in allen Dimensionen „unendlich“ gesetzt, da die Regel den gesamten Wertebereich abdeckt und noch nicht durch Konflikte (Regeln bzw. Muster anderer Klassen)

verkleinert wurde. Der neue Prototyp p_{m+1} wird initialisiert als:

$$\begin{aligned}
\vec{v}^{\star}_{m+1} &= \vec{x}^{\star}_{i} \\
\vec{a}^{\star}_{m+1} &= (-\infty, \cdots, -\infty)^T \\
\vec{b}^{k}_{m+1} &= \vec{x}^{\star}_{i} \\
\vec{c}^{\star}_{m+1} &= \vec{x}^{\star}_{i} \\
\vec{d}^{k}_{m+1} &= (+\infty, \cdots, +\infty)^T \\
\omega^{k}_{m+1} &= 1
\end{aligned}$$

shrink Das *shrink* passt alle Prototypen anderer Klassen bezüglich des aktuellen Trainingsmusters an. Dabei wird der Einflussbereich (und – wenn nötig – der Kernbereich) der Regel entsprechend des Musters verkleinert. Das kann dazu führen, dass in der nächsten Epoche ein neuer Prototyp eingefügt werden muss, da es Muster geben kann, die nun nicht mehr abgedeckt werden. Im nächsten Abschnitt werden hierzu Heuristiken zur Konfliktlösung vorgestellt.

Der Algorithmus verdeutlicht die Funktionsweise des DDA-RecBF-Trainingsverfahrens für das iterative Training rechteckiger prototypbasierter Netzwerke.

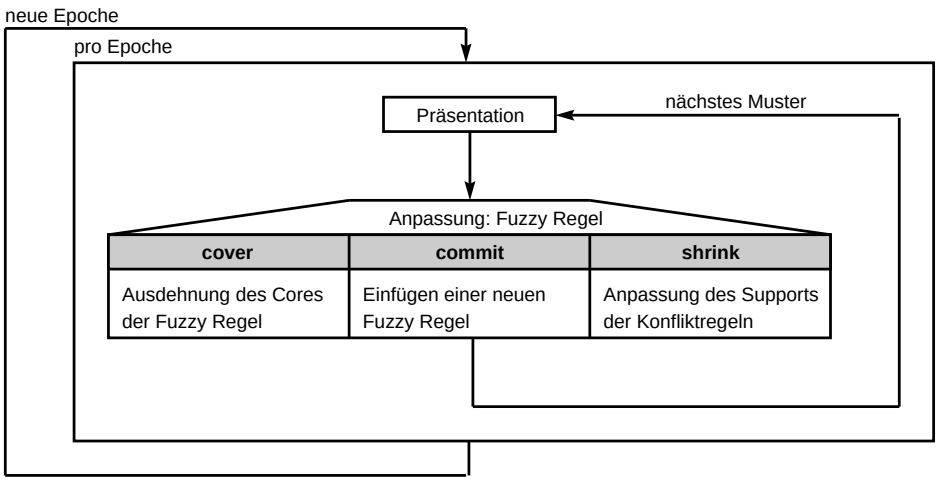


Abbildung 2.10: Das Diagramm zeigt die drei Fälle, die bei Anpassung einer Fuzzy-Regelbasis durch ein neues Muster unterschieden werden müssen.

ALGORITHM FRL
  REPEAT
    changed is 0 // 1, wenn ein *shrink* oder *commit* ausgeführt wurde
    FORALL Prototypen p_j^k **DO**
      $\omega_j^k = 0$ // setze Gewicht zurück
      $\vec{b}_j^k = \vec{c}_j^k = \vec{v}_j^k$ // setze Core-Bereich zurück
    ENDFOR
    FORALL Trainingsmuster $\vec{x}_i^k$ **DO**
      IF $\exists p_j^k$ covers $\vec{x}_i^k$ **THEN** // Trainingsmuster ist von Prototyp erklärt
        cover$(\mathrm{p}_j^k, \vec{x}_i^k)$
      ELSE
        commit$(\mathrm{p}_{m+1}^k, \vec{x}_i^k)$
        changed $=$ 1 // füge neuen Prototypen ein
      ENDIF
      FORALL Prototypen $\mathrm{p}_j^{\hat{k}}, \hat{k} \neq k$ **DO**
                  // verkleinere alle Prototypen anderer Klassen
        IF (***shrink***$(\mathrm{p}_j^{\hat{k}}, \vec{x}_i^k)$) **THEN** *changed* $=$ *true* **ENDIF**
      ENDFOR
    ENDFOR
  UNTIL (*changed is* 0)

Die äußere Schleife wird während des Trainings so lange durchlaufen bis entweder keine Prototypen mehr eingefügt oder keine mehr verkleinert werden müssen. Um sich eine mögliche Ausführung von *commit* oder *shrink* zu merken, dient die Variable *changed*. Ist diese Variable gesetzt (1), startet ein neuer Trainingsdurchlauf mit Präsentation aller Eingabemuster, da Prototypen angepasst wurden und sich dadurch das Netz in einem instabilen Zustand befinden kann. Zu Beginn jeder Epoche werden alle Gewichte sowie die Kernbereiche aller Regeln auf den Initialvektor zurückgesetzt. Im Gegensatz zum normalen PNN können Neuronen der verdeckten Schicht mehrere Trainingsmuster abdecken. Daher werden die Ausgangsgewichte ω^k verwendet, die die a-priori-Klassenwahrscheinlichkeit für jedes Neuron modellieren. Bei gleicher Aktivierung eines Prototypen überdeckt der besser, der ein höheres Gewicht ω^k besitzt.

Für jedes Trainingsmuster $\vec{x}$ wird geprüft, ob es einen Prototyp p^k der Klasse k gibt, der eine Aktivierung größer Null liefert. Ist dies erfüllt, ist das Muster abgedeckt. D. h.,

das zugehörige Ausgangsgewicht wird inkrementiert und die Grenzen des Kernbereiches notfalls vergrößert. Im anderen Fall muss ein neues Neuron für einen Prototypen p eingefügt werden. Dabei wird das Ausgangsgewicht auf Eins gesetzt (erklärt genau ein Trainingsmuster), und der Zentrumsvektor $\vec{v}$ wird mit dem Trainingsdatenpunkt $\vec{x}$ initialisiert, der gleichzeitig den Kernbereich der neuen Regel markiert. Weiterhin werden die Grenzen des Einflussbereichs so groß wie möglich gewählt, allerdings ohne in Konflikt mit angrenzenden Neuronen anderer Klassen zu treten. Das kann durch den anschließenden Aufruf der *shrink*-Prozedur verhindert werden. Diese Methode verkleinert – wenn notwendig – den neu eingefügten Prototypen bezüglich aller in Konflikt stehender Prototypen anderer Klassen. Dieses muss nicht notwendigerweise ausgeführt werden, da es ebenfalls am Ende jeder Präsentation eines Eingabemusters ausgeführt wird. Dieser Schritt ist erforderlich, da die *cover*-Methode möglicherweise Prototypen vergrößert hat, die nun zu Konflikten führen. Das *shrink* verkleinert – im Unterschied zum erst genannten *shrink* – alle Prototypen anderer Klassen, die mit dem aktuellen Eingabemuster im Konflikt stehen. Dieses Vorgehen erspart das Speichern aller Trainingsvektoren, da Konflikte mit Trainingsmustern, die nicht Referenzvektoren eines Prototypen sind, während der nächsten Trainingsepoche aufgelöst werden. Wird während einer Epoche ein neuer Prototyp eingefügt oder ein existierender angepasst, wird das Training um eine weitere Epoche fortgesetzt. Das bedeutet, dass alle Eingabemuster dem Netz von Neuem präsentiert werden. In Berthold (1997) wurde die Terminierung des Algorithmus bewiesen und gezeigt, dass maximal die Anzahl der Trainingsmuster an Epochen notwendig ist, um ein stabiles Netz zu konstruieren.

Ein neu eingefügter Prototyp muss nicht sofort durch alle Konfliktprototypen anderer Klassen angepasst werden. Es hat aber den Vorteil, dass sich die Anzahl der Epochen verringern kann, da der neue Prototyp sofort an das Netzwerk und somit an alle anderen Neuronen angepasst wird. Das führt aber wiederum zu einer Verkleinerung des Einflussbereichs dieser Regel. Nachteile dieser Methodik sind, dass unter Umständen mehr Regeln erzeugt werden als notwendig sind, da beim Hintereinander auftreten mehrerer Muster einer Klasse eine sofortige Anpassung des Prototyps nicht immer sinnvoll ist, und dass ein neu eingefügtes Neuron immer durch das erste gelernte Neuron einer anderen Klasse verkleinert wird. Das löst eine Art Symmetrieeffekt im Netz aus und neigt dazu, viele kleine Regeln zu erzeugen. Deshalb wird bei den weiteren Untersuchungen darauf verzichtet, Prototypen sofort zu adaptieren. Eine weitere Möglichkeit diesen Effekt zu vermeiden, kann durch Speichern einer sogenannten Konfliktliste erreicht werden, die beim Erreichen einer festgelegten Größe,

die Regel(n) mit dem geringsten „Verlust“ [1] anpasst, siehe Abschnitt 2.4. In Berthold (2003) wird gezeigt, dass diese Methodik zu einer Verringerung der Regelanzahl führen kann.

Heuristiken der Konfliktlösung

Die notwendige Modifikation des DDA, um diese Netze zu trainieren, beschränkt sich auf die Anpassung der Prozedur *shrink* zur Konfliktvermeidung. Die Überprüfung auf ausreichende Überdeckung und das Einfügen einer neuen Regel weist keine Unterschiede zum normalen DDA auf. Beim Einfügen eines neuen Prototypen wird die Größe der neuen Regel so gewählt, dass der gesamte Merkmalsraum abgedeckt wird. In diesem Fall ist also der Einflussbereich gleich dem Merkmalsraum. Der Kernbereich ist punktförmig, entspricht also dem neuen Zentrumsvektor. Im Falle rechteckiger Prototypen in einem n-dimensionalen Merkmalsraum gibt es n Möglichkeiten einen Konflikt zu lösen [2]. Um möglichst große Regeln zu erhalten, d. h. solche, die einen großen Bereich des Eingaberaumes abdecken, gilt es allgemein, die Dimension zu bestimmen, die den minimalen Volumenverlust verursacht. Die optimale Dimension i bezüglich des Volumenverlustes ergibt sich aus:

$$i_{\min} = \operatorname{argmin}_{i=1,\cdots,n} \{V_i\} .$$

Der Volumenverlust V_i des Prototypen p mit den Parametern $\langle a, b, c, d \rangle$ beim Abschneiden in der Dimension i kann berechnet werden durch:

$$V_i = d_i^*(\vec{x}, \mathrm{p}) \prod_{j=1, j\neq i}^{n} d_j^{\times}(\vec{x}, \mathrm{p}),$$

wobei d_i^* mit $1 \leq i \leq n$ den Abstand zwischen Eingabevektor $\vec{x}$ und den Grenzen (Kern- oder Einflussbereich) des Prototypen p in der Dimension i und $d_j^{\times}$ mit $1 \leq j \leq n$ die Ausdehnung der Regel in der Dimension j bezeichnet. Um die Volumenverluste direkt vergleichen zu können, ist es sinnvoll, den Verlust über das Gesamtvolumen

[1] Der Verlust kann zum Beispiel durch das Volumen des Hyperquader einer Regel ausgedrückt werden.

[2] Ein Konflikt kann alleine schon gelöst werden, wenn *eine* Dimension im Einfluss- oder – wenn nötig – zusätzlich im Kernbereich verkleinert wird.

zu normalisieren:

$$V_i^{norm} = d_i^*(\vec{x}, \mathrm{p}) \cdot \frac{\prod_{j=1, j\neq i}^{n} d_j^{\times}(\vec{x}, \mathrm{p})}{\prod_{j=1}^{n} d_j^{\times}(\vec{x}, \mathrm{p})} = \frac{d_i^*(\vec{x}, \mathrm{p})}{d_i^{\times}(\vec{x}, \mathrm{p})}.$$

Die Berechnung vereinfacht sich soweit, dass nur noch eine Dimension betrachtet werden muss. Das verringert den Rechenaufwand deutlich. Da eine optimale Bestimmung des günstigsten Rechtecks unter Berücksichtigung aller Konfliktpunkte zu komplex ist, wird eine Heuristik verwendet, die für jedes Eingabemuster die Größe des zu verkleinernden Rechtecks minimiert.

Nachfolgend werden verschiedene Möglichkeiten der Messung des Volumenverlustes betrachtet, um den Konflikt zu lösen. Dabei werden Volumenverluste im Kernbereich $[b, d]$ und Volumenverluste im Einflussbereich (a, b) und (c, d) getrennt behandelt. Abbildung 2.11 zeigt drei Konfliktlösungen im Einflussbereich der Regel an einem zweidimensionalen Beispiel. Diese Beispiele zeigen den Volumenverlust getrennt in zwei verschiedenen Dimensionen (Spalten).

Man erkennt den Unterschied in den Verhältnissen der Volumina. Das Beispiel (a)–(b) zeigt das Gesamtvolumen der Regel, (c)–(d) das Volumen zum Zentrumsvektor und (e)–(f) das Volumen zum Rand des Kernbereiches der Regel im Verhältnis zum Volumenverlust. Im Beispiel (a)–(b) wird der Schnitt vertikal durchgeführt, da der Volumenverlust im Verhältnis zum Gesamtvolumen am geringsten ist. Im Beispiel (c)–(d) und (e)–(f) hingegen wird der Schnitt horizontal gemacht. Im Falle der Konfliktlösung im Einflussbereich kann die Abstandsfunktion d_i^* und $d^{\times}$ für die Gewichtung wie folgt definiert werden:

$$d^*(\vec{x}, \mathrm{p}) = \begin{cases} x_i - a_i & : \quad x_i \leq v_i, \\ d_i - x_i & : \quad \text{sonst.} \end{cases}$$

regelbasiert Bei der regelbasierten Konfliktlösung wird der Volumenverlust über das Gesamtvolumen der Regel gewichtet. Damit berechnet sich die Distanz $d_i^{\times}$ aus:

$$d^{\times}(\vec{x}, \mathrm{p}) = d_i - a_i,$$

kernbasiert Bei der kernbasierten Konfliktlösung wird der Volumenverlust über das Volumen zum Zentrumsvektor gewichtet, damit berechnet sich die Distanz $d_i^{\times}$

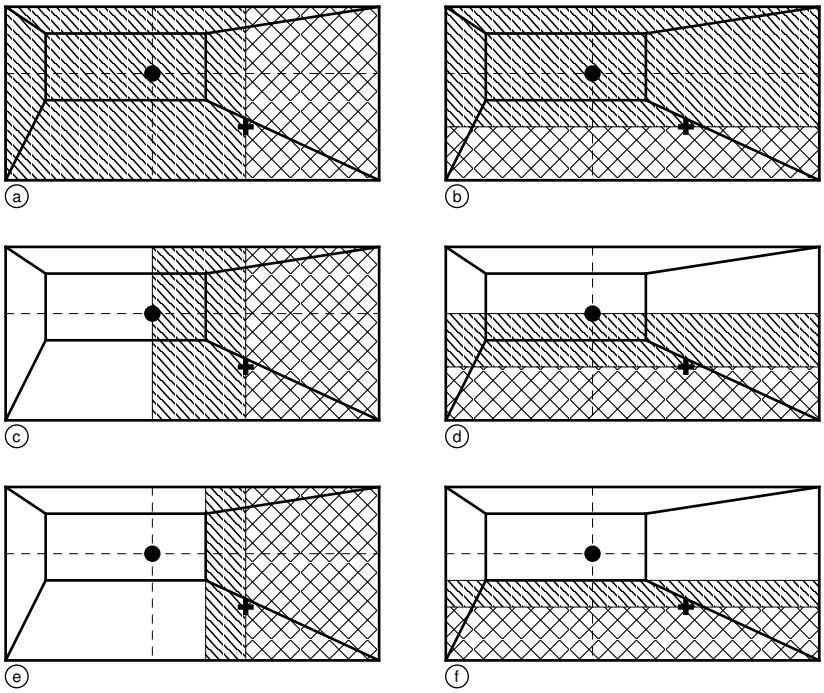


Abbildung 2.11: Zweidimensionales Beispiel der Konfliktlösung im Einflussbereich. (a)–(b) regelbasiert, (c)–(d) kernbasiert und (e)–(f) bereichbasiert, wobei (a),(c),(e) vertikal und (b),(d),(f) horizontal schneiden (Fläche des Verlustvolumen gerastert, Fläche des Wichtungsvolumen gestreift und gerastert).

aus:

$$d^{\times}(\vec{x}, \mathrm{p}) = \begin{cases} v_i - a_i & : \quad x_i \leq v_i, \\ d_i - v_i & : \quad \text{sonst}, \end{cases}$$

bereichbasiert Bei der bereichbasierten Konfliktlösung wird der Volumenverlust über das Volumen zu den Grenzen des Kernbereiches gewichtet, damit berechnet sich die Distanz $d_i^{\times}$ aus:

$$d_i^{\times}(\vec{x}, \mathrm{p}) = \begin{cases} b_i - a_i & : \quad x_i \leq v_i, \\ d_i - c_i & : \quad \text{sonst}. \end{cases}$$

Das nächste Beispiel zeigt in Abbildung 2.12 die gleiche Fuzzy-Menge, wobei der Konflikt jetzt im Kernbereich der Regel gelöst werden muss.

Hierbei wird die Heuristik angewandt, um im Fall von (a)–(b) horizontal und (c)–(d) vertikal aufgrund des geringeren Verhältnisses der Volumina zu schneiden. Im

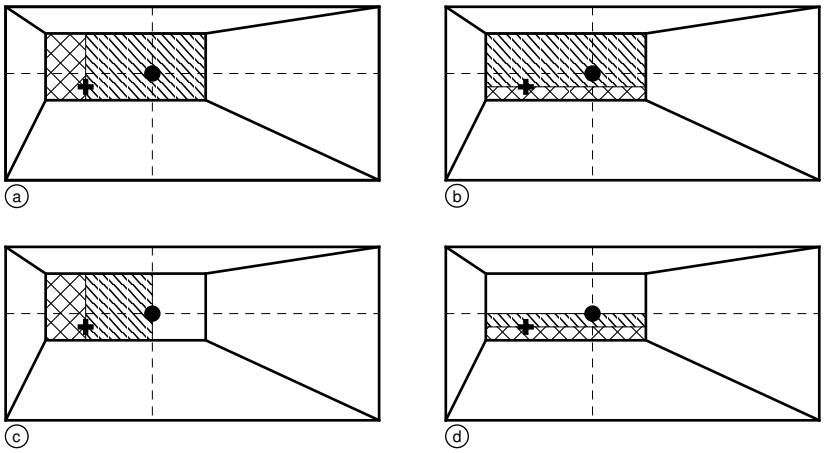


Abbildung 2.12: Zweidimensionales Beispiel der Konfliktlösung im Kernbereich. (a)–(b) bereich-/regelbasiert, (c)–(d) kernbasiert, wobei (a) und (c) vertikal und (b) und (d) horizontal schneiden (Fläche des Verlustvolumen gerastert, Fläche des Wichtungsvolumen gestreift und gerastert).

Falle der Konfliktlösung im Kernbereich lässt sich die Abstandsfunktion d_i^* und $d^\times$ für die Gewichtung wie folgt definieren:

$$d_i^*(\vec{x}, \mathrm{p}) = \begin{cases} x_i - b_i & : \quad x_i \leq v_i, \\ c_i - x_i & : \quad \text{sonst.} \end{cases}$$

bereich-/regelbasiert Bei der bereich-/regelbasierten Konfliktlösung wird der Volumenverlust über das Gesamtvolumen der Regel im Kernbereich gewichtet, d. h., die Distanz $d_i^\times$ berechnet sich aus:

$$d_i^\times(\vec{x}, \mathrm{p}) = d_i - c_i,$$

kernbasiert Bei der kernbasierten Konfliktlösung wird der Volumenverlust über die Distanz zum Zentrumsvektor gewichtet, d. h., die Distanz $d^\times$ berechnet sich aus:

$$d^\times(\vec{x}, \mathrm{p}) = \begin{cases} v_i - b_i & : \quad x_i \leq v_i, \\ c_i - v_i & : \quad \text{sonst.} \end{cases}$$

In Berthold (2003) wird gezeigt, dass durch Speichern einer Liste von Konflikten eine Verbesserung der Klassifikationsgüte erreicht werden kann. Diese Konfliktliste

enthält eine bestimmte Anzahl von Konfliktregeln. Wenn die Liste eine bestimmte Größe erreicht hat, wird die Regel mit dem optimalen Schnitt verkleinert. In Kapitel 3.2 werden die verschiedenen Konfliktlösungsstrategien experimentell validiert und diskutiert, um den Einfluss auf das Fuzzy-Regelmodell zu zeigen.

Fuzzy-Regeln mit fehlenden Werten

Unvollständige Informationen in den Trainingsdaten stellen oftmals ein Problem für den Lernalgorithmus dar, zum Beispiel weil Distanzen zwischen zwei Eingabevektoren nicht bestimmt werden können. Das Ignorieren dieser fehlenden Werte bzw. damit ganzer Trainingsmuster ist häufig nicht praktikabel und kann zu einem unbrauchbaren Modell führen. Es existieren Vorangehensweisen, die vor dem eigentlichen Training fehlende Werte ersetzen. Hier kommen oft statistische Kennzahlen zum Einsatz, die die Wahrscheinlichkeitsverteilung innerhalb einer bestimmten Klasse einbeziehen und so eine gute Näherung für den fehlenden Wert ermitteln. Basierend auf dem hier vorgestellten Fuzzy-Regellerner wurde eine Ersetzungsmethode vorgeschlagen, die fehlende Werte anhand des aktuellen Modells erraten (Berthold und Huber, 1997). Der sogenannte „Best Guess“-Wert wird bezüglich der besten Regel ermittelt. Dieser berechnet sich aus den Projektionen der validen Fuzzy-Zugehörigkeitsfunktionen der anderen Attribute.

Der von uns vorgestellte Ansatz zur Verbesserung des Fuzzy-Regellerners geht einen anderen Weg und bestimmt fehlende Werte erst, wenn diese Informationen basierend auf dem aktuellen Trainingsmuster vorhanden sind (Gabriel und Berthold, 2005). Bei der Initialisierung neuer Fuzzy-Regeln werden die Kernbereiche der Fuzzy-Zugehörigkeitsfunktionen immer mit dem aktuellen Eingabevektor ersetzt; die Support-Bereiche werden nicht eingeschränkt. Im Falle fehlender Werte in den Daten werden diese Dimensionen nicht sofort gesetzt, d. h. die Kern- und Einflussbereiche sind nicht definiert und die Ausgabe dieser Fuzzy-Dimension ist immer 1. Während des Trainingsprozesses wird immer die beste Fuzzy-Regel ermittelt, die das aktuelle Muster erklären kann. Enthält diese Fuzzy-Regel fehlende Werte, werden diese nun durch die aktuellen Werte des Trainingsvektors ersetzt; Dimensionen mit fehlenden Werten in Trainingsdaten bleiben unberührt.

Damit müssen nun, wie in Abbildung 2.13 zu sehen, die folgenden drei Operationen auf Fuzzy-Regeln angepasst werden:

cover Wenn ein neues Trainingsmuster fehlende Attributwerte enthält, werden diese Dimensionen ignoriert. Fehlende Dimensionen innerhalb der Fuzzy-Regeln werden mit Attributwerten initialisiert – soweit welche existieren.

commit Wenn eine neue Fuzzy-Regel kreiert wurde, werden die fehlenden Dimensionen als uneingeschränkt initialisiert, bis ein erneuter *cover* Schritt diese Dimensionen durch Werte aus den Trainingsdaten füllt. Alle anderen Dimensionen werden wie zuvor nicht eingeschränkt, d. h. der Einflussbereich deckt den kompletten Wertebereich ab.

shrink Wenn eine existierende Fuzzy-Regel anhand eines Trainingsmusters verkleinern werden muss, so werden die fehlenden Dimensionen ignoriert. Das ergibt sich daraus, dass noch kein initialer Kernvektor gesetzt wurde, an dem die Regel bezüglich des Datenpunktes verkleinert werden kann.

Während der Klassifikation von unbekannten Instanzen, werden die fehlenden Dimensionen als uneingeschränkt behandelt. Damit ergibt sich ein Fuzzy-Zugehörigkeitsgrad von 1. Auf diese Werte kann dann die Fuzzy-Norm angewendet werden, um die Ausgabe jeder Regel und anschließend die finale Ausgabe des Systems über alle Regeln zu berechnen.

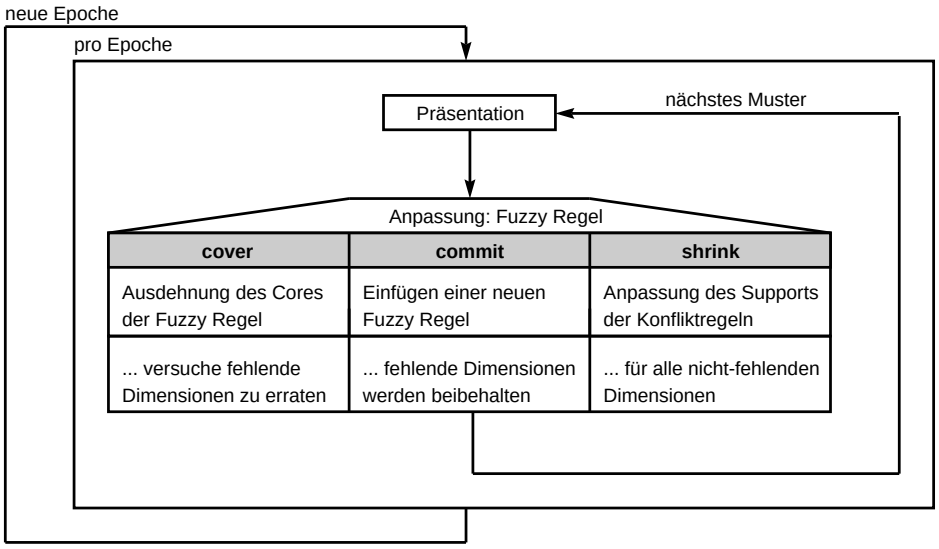


Abbildung 2.13: Das Diagramm zeigt die drei Fälle, die bei Anpassung einer Fuzzy-Regelbasis durch ein neues Muster mit fehlenden Werten beachtet werden müssen.

Beispiel: Erzeugung einer Fuzzy-Regelbasis

In diesem Abschnitt soll ein einfaches eindimensionales Beispiel mit zwei Klassen betrachtet werden. Die Datenpunkte werden nacheinander durch Fuzzy-Regeln abgedeckt, sodass am Ende des Lernprozesses alle Muster korrekt erklärt werden. In den Abbildungen 2.14 bis 2.17 wird die Arbeitsweise des Fuzzy-Regelalgorithmus demonstriert. Die gegebene Datenbasis besteht aus insgesamt sieben Eingabemustern, denen die zwei Klassen ● und + zugeordnet sind. Neue Trainingsmuster werden schwarz gezeichnet; schon existierende treten grau in den Hintergrund. Bei jedem Schritt kommt genau ein Datenbeispiel hinzu, wobei das System sich automatisch anpasst und wenn nötig eine neue Regel einfügt.

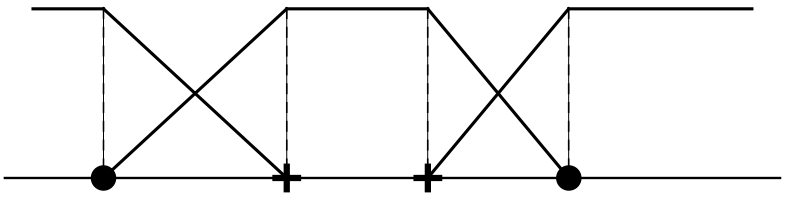


Abbildung 2.14: Anfangszustand des Systems mit vier Datenpunkten.

Abbildung 2.14 zeigt den Anfangszustand des Systems mit vier Datenpunkten, die von drei Fuzzy-Regeln mit trapezförmigen Zugehörigkeitsfunktionen überdeckt werden. Die zwei äußeren Punkte der Klasse ● werden durch eine links- und eine rechts-offene Trapez-Zugehörigkeitsfunktion beschrieben. Die beiden Muster der Klasse + in der Mitte werden ebenfalls durch eine Trapezfunktion abgedeckt, dessen Kernbereich durch die beiden Muster begrenzt wird. Die Einflussbereiche aller Regeln fallen linear bis zum jeweils benachbarten Datenpunkt einer anderen Klasse ab.

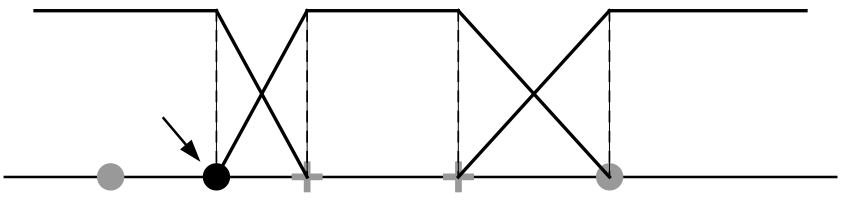


Abbildung 2.15: Regelsystem nach Einfügen eines Musters der Klasse ●.

Das Regelsystem nach Einfügen eines Musters der Klasse ● ist in Abbildung 2.15 zu sehen. Dieser neue Datenpunkt wird zwischen dem ersten und zweiten Muster eingefügt. Hierfür muss der Kernbereich der links-offenen Trapezfunktion nach rechts

erweitert werden, um das neue Muster zu überdecken. Weiterhin wird der Einflussbereich der mittleren Fuzzy-Regel bezüglich des neuen Datenpunktes verkleinert. Die rechts-offene Trapezfunktion wird nicht verändert. Am Ende erhält man ein System aus drei Fuzzy-Regeln, die fünf Muster erklären.

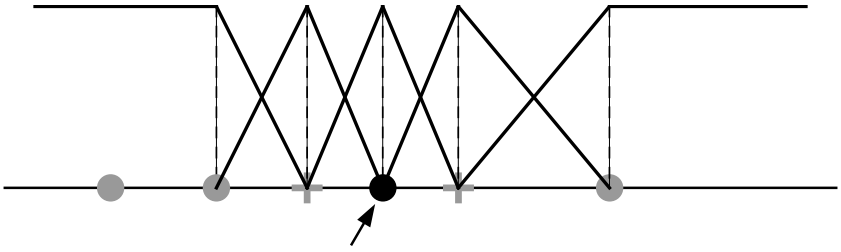


Abbildung 2.16: Regelsystem nach Einfügen eines Datenpunktes der Klasse ●.

Abbildung 2.16 zeigt das Regelsystem nach Erklären eines weiteren Datenpunktes der Klasse ●, wobei eine zusätzliche Dreieckfunktion zur Überdeckung eingefügt werden muss. Zusätzlich wird nach einer weiteren Iteration eine neue Dreieckfunktion der Klasse **+** hinzugefügt, um das nicht mehr erklärte Nachbarmuster zu überdecken. Die Grenzen der Einflussbereiche aller drei Regeln liegen dann auf den benachbarten Mustern anderer Klassen. Weiterhin besteht das System aus den links- und rechts-offenen Trapezfunktionen, die nicht verändert werden.

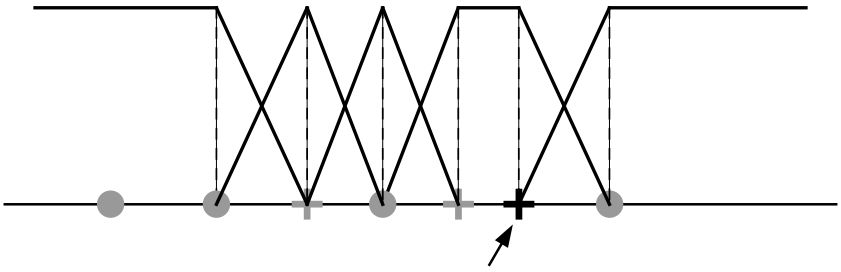


Abbildung 2.17: System mit fünf Fuzzy-Regeln nach Präsentation eines Musters **+**.

Abbildung 2.17 zeigt das System mit fünf Fuzzy-Regeln nach Präsentation eines weiteren Musters. In diesem Schritt wird ein Muster der Klasse **+** hinzugefügt, das zur Erweiterung des Kernbereiches der schon existierenden Nachbarregel führt. Das System enthält am Ende fünf Regeln, welche die sieben Trainingsmuster korrekt erklären. Das resultierende Regelsystem befindet sich in einem stabilen Zustand, in dem alle Datenpunkte perfekt gelernt sind.

Dieses einfache Beispiel verdeutlicht, dass durch den zugrunde liegenden Lerner viele Regeln notwendig sind, um die Daten ausreichend zu erklären, d. h., um ein

Regelmodell zu erzeugen, welches eine gute Approximation auf den Daten darstellt. Hierdurch wird das Modell komplex und kann häufig nicht interpretiert werden. Interessant wäre an dieser Stelle eine Ordnung auf den Regeln zu haben, damit Regeln, nach bestimmten Eigenschaften sortiert, ausgewertet werden können. Komplexere Regelsysteme könnten so zu einer Hierarchie von Regelmodellen erweitert werden, in der jede Schicht nur wenige Regeln enthält und verständlich bleibt.

2.5 Beschränkungen und Zusammenfassung

In diesem Kapitel wurden Fuzzy-Regelsysteme besprochen und speziell auf die Möglichkeit eingegangen, Fuzzy-Regeln automatisch aus Daten zu lernen. Es wurde ein konstruktives Trainingsverfahren durch rechteckige Fuzzy-Prototypen erweitert, um Fuzzy-Regelmodelle zu erzeugen. Dieser Algorithmus fügt Regeln während des Lernprozesses zu einer bestehenden Regelbasis hinzu und passt diese, wenn notwendig, an. Hierfür wurden verschiedene Heuristiken zur Beseitigung von Konflikten mit Regeln anderer Klassen vorgestellt, die in Kapitel 3.2 ausführlich evaluiert werden. Im Zusammenhang mit Fuzzy-Regelsystemen wurden verschiedene Fuzzy-Normen beschrieben, die ebenfalls in den Experimenten in Kapitel 3.3 auf mehreren Benchmark-Datensätzen evaluiert werden, um die Klassifikationsgüte mit anderen State-of-the-Art Verfahren zu vergleichen.

Die klassischen Regellernverfahren erzeugen häufig eine große Anzahl ungeordneter Regeln. Diese lassen sich auf Basis ihrer lokalen Eigenschaften, wie Größe, Gewicht oder Anzahl der eingeschränkten Merkmale, vergleichen und sortieren. Dennoch werden solche Modelle schnell sehr umfangreich und unübersichtlich, wenn solche Systeme visuelle exploriert werden sollen. Oftmals gehen wichtige Detailregeln in der Menge von Regeln unter, da die Beziehung zu stärkeren Regeln nicht explizit modelliert wird. Diese Regelsysteme wollen wir als klassische oder flache Regelmodelle bezeichnen, da nur eine einfache Ordnung zwischen diesen Regeln gegeben ist; im Gegensatz hierzu können Beziehungen zwischen in Schichten gruppierten Regeln besser beschrieben werden. Da oftmals eine Vielzahl von Regeln durch den Algorithmus erzeugt wird, ist es für die Exploration solcher Systeme wichtig, dass die Beziehung zwischen Regeln klar definiert ist und Regeln auf unterschiedlichen Niveaus betrachtet werden können. Einfache Regelsysteme können Beziehungen nur implizit abbilden und sind daher für große und komplexe Datensätze potenziell nicht geeignet.

In den Experimenten in Kapitel 6 wird gezeigt, dass das lokale Fuzzy-Regellernverfahren häufig zu viele Regeln generiert, um eine möglichst gute Approximation des zugrunde liegenden Konzeptes zu erzeugen. In unseren weiteren Untersuchungen werden Ansätze betrachtet, die Regelbasen erzeugen, die im Gegensatz zu den hier aufgezeigten Verfahren, Ausnahmen und Artefakte getrennt modellieren. Hierbei wird es im Speziellen um hierarchisch-strukturierte Fuzzy-Regelsysteme gehen. Diese bieten im Vergleich zu den flachen Regelsystemen einige Vorteile. Da sie aus nur wenigen Regeln in jeder Schicht bestehen, bleibt das Modell verständlich und kann auf verschiedenen Abstraktionsebenen exploriert werden.

Kapitel 3

Experimente: Fuzzy-Regelmodelle

In diesem Kapitel wird der vorgestellte hierarchische Fuzzy-Regellerner basierend auf acht frei verfügbaren Benchmark-Datensätzen eines ESPRIT-Projekts, dem „StatLog-Projekts“ (Michie u. a., 1994), experimentell evaluiert. Es soll gezeigt werden, dass dieses Verfahren vergleichbare Ergebnisse zu anderen Standardansätzen liefert und welchen Einfluss die Wahl der Parameter auf die Generalisierungsleistung des Klassifikators hat.

Entscheidend für die Ausführung der Experimente ist die Gegenüberstellung der Ergebnisse sowie die Anwendung einiger bekannter Verfahren. Das ist ein wichtiges Kriterium für die Auswahl der Datensätze. Zusätzlich wurde jedes der verwendeten Klassifikationsverfahren von genau einem Experten eingesetzt, sodass eine Bevorzugung bestimmter Verfahren und Strategien vermieden wird. Im Gegensatz zu den weitverbreiteten und oft verwendeten Datensätzen des FTP-Servers der UCI werden die im StatLog-Projekt durchgeführten Versuche koordiniert (Asuncion und Newman, 2007). In diesem Projekt werden zwanzig verschiedene Klassifikatoren verwendet; darunter mehrere statistische Methoden, einige Ansätze aus dem Bereich des maschinellen Lernens sowie verschiedene Arten von Neuronalen Netzen. Für diese Untersuchungen werden die in diesem Projekt verwendeten Methoden übernommen, um die Vergleichbarkeit der Ergebnisse zu gewährleisten und die Generalisierungsleistung der Klassifikatoren vergleichen zu können. Zwar sind für alle zwanzig verwendeten Datensätze Ergebnisse vorhanden, allerdings sind einige dieser Datensätze nicht öffentlich verfügbar. Die Untersuchungen werden auf acht ausgewählten Datensätzen durchgeführt, die ausschließlich auf numerischen Merkmalen basieren.

Die verwendeten Datensätze stammen aus verschiedenen Gebieten – Bilderken-

nung, Kreditwesen und Medizin – und enthalten zwischen 690 Eingabemustern für die Daten aus dem Australischen Kreditwesen und 58 000 in der Shuttle Landing Control Datenbank. Weiterhin unterscheiden sich die Probleme in der Dimensionalität des Merkmalsraumes, der von 8 Dimensionen für den Diabetes bis zu 60 Dimensionen des DNA Datensatzes reicht. Die Anzahl der Klassen variiert von einfachen Zweiklassenproblemen bis zu 26 Klassen des Letter Beispiels.

Im nächsten Abschnitt werden Eigenschaften und Besonderheiten der verwendeten Datensätze beschrieben, bevor sie anschließend im Einzelnen getestet und validiert werden. Hierfür werden unter Anwendung des Fuzzy-Regellerners fünf verschiedene Fuzzy-Normen und drei Konfliktlösungsstrategien evaluiert. Diese Parameter beinhalten die Minimum/Maximum-Norm, Produkt-Norm, Yager$_{1/2}$-Norm, Luka-Norm und Yager$_2$-Norm (siehe Kapitel 2.3) sowie die Konfliktlösungsstrategien regel-, kern- und bereichsbasiert (siehe Kapitel 2.4). Im letzten Teil werden die Ergebnisse abschließend zusammengefasst und bewertet.

3.1 Eigenschaften der Benchmark-Datensätze

Die Zusammenfassung in Tabelle 3.1 verschafft einen Überblick der verwendeten Datensätze. Aufgrund der geringen Anzahl von Eingabedaten der ersten vier Datensätze wird dort das Verfahren der Kreuz-Validierung [1] angewendet. Der Generalisierungsfehler berechnet sich aus dem arithmetischen Mittel über alle n Partitionen. Für alle anderen Benchmarks ist die Aufteilung in Test- und Trainingsdaten angegeben. Hier wird der Fehler auf den Testdaten als Maß für die Generalisierungsfähigkeit verwendet.

Pima Indians Diabetes Daten

Mithilfe von acht Merkmalen, wie zum Beispiel dem Körpergewicht, der Körperdichte und dem Alter eines Patienten, soll vorhergesagt werden, ob ein Patient positiv auf einen Diabetestest reagieren wird oder nicht; Klasse 1 (*negativ*) und Klasse 2 (*positiv*). Die insgesamt 768 Datensätze werden mit 12-facher Kreuz-Validierung für die

[1] In der Praxis wird das Verfahren der Kreuz-Validierung oft verwendet. Dabei werden die Daten in $M = m \cdot K^{-}1$ verschiedene Partitionen aufgeteilt, wobei K Muster zum Test und $m - K$ Muster für das Training verwendet werden. In Amari (1995) wird gezeigt, dass ein Verhältnis von $\sqrt{2p} - 1$ für große Datensätze optimal ist, wobei p die Anzahl einstellbarer Parameter des Klassifikator ist.

Tabelle 3.1: Verwendete Benchmark-Datensätze des StatLog-Projekts. Daten, bei denen das Verfahren der Kreuz-Validierung verwendet wird, befindet sich die Gesamtanzahl der Muster in *#Trainingsdaten* und die Anzahl der Rotationen in der Spalte *#Testdaten*. Für die anderen Datensätze ist die Aufteilung in Test- und Trainingsdaten angeben.

Datensatz	Attribute#	#Klassen	#Trainingsdaten	#Testdaten
Diabetes	8	2	768	12-fold
Aust. Cred.	14	2	690	10-fold
Vehicle	18	4	846	9-fold
Segment	11	7	2 310	10-fold
Shuttle	9	7	43 500	14 500
SatImage	36	6	4 435	2 000
DNA	60	3	2 000	1 186
Letter	16	26	16 000	4 000

folgenden Experimente verwendet. Die Zielvariable ist bezogen auf das Kriterium der Welt-Gesundheitsorganisation, d. h., eine Person ist zuckerkrank, wenn eine 2-stündige Plasma-Glucose-Konzentration einen Schwellwert von 200mg/dl übersteigt. Die untersuchte Population lebt in der Umgebung von Phoenix, Arizona, USA.

Australian Credit Approval

Dieser Datensatz aus dem Australischen Bankwesen soll helfen, die Kreditwürdigkeit von Kunden zu beurteilen; Klasse 1 (*kreditwürdig*) und Klasse 2 (*nicht kreditwürdig*). Aus datenschutzrechtlichen Gründen sind die Merkmale nicht bekannt gegeben, sodass eine Interpretation der Daten nicht möglich ist. Der 14-dimensionale Datensatz besteht aus 690 Beispielmustern.

Vehicle Silhouettes Daten

Dieser Datensatz beruht auf der Klassifikation von Bildern. Dafür wird eine Menge von Attributen aus dem jeweiligen Graubild extrahiert. Die Aufgabe besteht nun darin, anhand der 2-dimensionalen Silhouette eines Fahrzeuges auf die vier Fahrzeugtypen – *Opel*, *Saab*, *Bus* und *Van* – zu schließen. Die eigentliche Aufnahme liegt nur als Schwarz/Weiß-Bild vor, d. h., es sind keine Grauwerte im Original vorhanden. Aus diesen 180×180-Pixel Bildern werden jeweils 18 zumeist geometrische Attribute

generiert, zum Beispiel das Längen/Breiten-Verhältnis der Fahrzeuge. Aufgrund der geringen Größe von 846 Beispielen wird ebenfalls das Kreuz-Validierungsverfahren angewendet.

Image Segmentation Daten

Dieses Beispiel beschreibt die Erkennung von sieben gleich verteilten Landschaftsarten – *Mauer*, *Himmel*, *Beton*, *Fenster*, *Weg*, *Gras* und *Blattwerk*. Aus jeder 3×3-Bildregion werden 19 Attribute extrahiert, zum Beispiel der Kontrastwert in horizontaler und vertikaler Richtung. Es werden 2 310 Datensätze mit 10-facher Kreuz-Validierung für die folgenden Experimente verwendet.

Shuttle Landing Control Daten

Der mit Abstand größte Datensatz ist unterteilt in 43 000 Trainings- und 14 500 Testbeispiele und stammt im Original von der NASA. Er beschreibt die Positionen von Radiatoren im Space-Shuttle, die mit 9 Attributen beschrieben und in 7 Klassen unterteilt werden können. Es ist interessant zu sehen, dass fast 80% der Daten zu Klasse 1 und nur sechs Muster zu Klasse 6 gehören.

Landsat-Satellite Daten

Dieser Bilddatensatz wurde aus Satellitenaufnahmen gewonnen. Dabei wurden aus den Bildern 3×3-Bereiche extrahiert und jeweils vier verschiedene Spektralbereiche untersucht. Mit den 35 resultierenden Attributen werden 6 Klassen unterschieden, die unterschiedliche Bodenarten beschreiben. Die insgesamt 6.435 Datensätze werden in 4 435 Test- und 2 000 Trainingsdaten unterteilt.

DNA Sequence Daten

In diesem aus der Molekularbiologie stammenden 60-dimensionalen Datensatz geht es um die Erkennung von Verbindungsstellen in einer DNA Sequenz, bei denen „überflüssige“ DNA bei der Proteinerzeugung entfernt werden. Ein „Intron“-Teilstück begrenzt einen zu entfernenden Teil, das „Exon“ bleibt erhalten. Jeder der 3 186 Datensätze veranschaulicht ein Fenster mit 60 Nucleotiden, wobei jedes Nucleotid durch

vier symbolische Werte für die Basen – Adenin, Cytosin, Guanin und Thymin – beschrieben wird und die Zuordnung in drei Klassen – *Intron-Extron-Verbindung*, *Exon-Intron-Verbindung* oder *keines der beiden* – erfolgen kann. Die 60 symbolischen Attribute wurden in 240 binäre Merkmale überführt. 2 000 Datensätze wurden zufällig zum Training ausgewählt und die verbleibenden 1 186 Beispiele wurden zum Test verwendet.

Letter Recognition Daten

Dieser Datensatz besteht aus Bildern der 26 lateinischen Großbuchstaben. Es wurden zwanzig verschiedene Schriftarten verwendet. Jedes Zeichen wurde zufallsgesteuert gestört, um 20 000 unterschiedliche Muster zu gewinnen. Im Anschluss wurden für jedes Muster 16 numerische Attribute berechnet, die auf sogenannten Eckzählern und statistischen Momenten basieren. Die Merkmale wurden in einem weiteren Schritt skaliert und diskretisiert, um in das ganzzahlige Intervall $[0, 15]$ zu fallen. Die Daten wurden in 15 000 Trainings- und 5 000 Testdaten aufgeteilt.

3.2 Ergebnisse verschiedener Konfliktlösungsstrategien

In diesem Abschnitt werden die Ergebnisse des Fuzzy-Regellerners mit den drei in Kapitel 2.4 vorgestellten Konfliktlösungsstrategien beschrieben. Es soll ermittelt werden, ob diese Parameter einen Einfluss auf den Generalisierungsfehler haben, und ob es einen Zusammenhang zwischen den verwendeten Normen und Konfliktlösungsstrategien gibt. Die nachfolgenden Absätze zeigen die Ergebnisse bei der Aufteilung in die regel-, kern- und bereichsbasierte Heuristik. Die Zahl in Klammern repräsentiert die durchschnittliche Abweichung vom Mittelwert des Fehlers in dieser Gruppe.

Regelbasierte Konfliktlösung

Die Tabelle 3.2 zeigt den prozentualen Fehler auf den Testdaten der acht verwendeten Datensätze unter Verwendung der regelbasierten Konfliktlösung. Hierbei schneiden fast alle Datensätze mit $\text{Yager}_{1/2}$-Norm am schlechtesten ab (-2.67%), nur für die Shuttle und DNA Daten wird ein geringfügig besseres Ergebnis erreicht. Für alle anderen Normen lassen sich keine signifikanten Unterschiede feststellen, wobei zu erwähnen ist, dass die Luka-Norm durchschnittlich schlechtere Ergebnisse (-0.19%)

Tabelle 3.2: Generalisierungsfehler (in %) auf den acht Benchmark-Datensätzen unter Verwendung der regelbasierten Konfliktlösungsstrategie für die fünf Normen – Min/Max, Produkt, $\text{Yager}_{1/2}$, Luka, Yager_2.

Datensatz	Norm				
	Min/Max	Produkt	$\text{Yager}_{1/2}$	Luka	Yager_2
Diabetes	29.03	29.43	30.99	29.43	26.30
Aust. Cred.	17.10	17.10	17.10	16.81	16.81
Vehicle	38.18	37.71	46.10	39.72	36.88
Segment	7.79	7.92	13.81	9.61	8.01
Shuttle	0.08	0.08	0.06	0.07	0.09
SatImage	16.50	16.55	26.30	20.20	17.75
DNA	32.63	32.63	36.59	36.93	32.29
Letter	24.28	24.32	29.47	25.79	23.24

liefert als die Minimum/Maximum-Norm (0.99%), Produkt-Norm (0.94%) und Yager_2-Norm (0.93%), die sich in den Ergebnissen kaum unterscheiden.

Kernbasierte Konfliktlösung

Eine andere Konfliktlösungsstrategie, die hier zur Anwendung kommen soll, ist die kernbasierte (siehe Tabelle 3.3).

Tabelle 3.3: Generalisierungsfehler (in %) auf den acht Benchmark-Datensätzen unter Verwendung der kernbasierten Konfliktlösungsstrategie für die fünf Normen – Min/Max, Produkt, $\text{Yager}_{1/2}$, Luka, Yager_2.

Datensatz	Norm				
	Min/Max	Produkt	$\text{Yager}_{1/2}$	Luka	Yager_2
Diabetes	28.78	29.17	30.21	27.21	26.82
Aust. Cred.	17.82	17.68	17.97	17.25	17.25
Vehicle	32.51	32.15	45.51	36.67	29.91
Segment	4.55	4.59	10.61	4.85	4.16
Shuttle	0.06	0.06	0.08	0.07	0.07
SatImage	14.20	13.90	29.65	20.95	14.10
DNA	32.63	32.72	36.68	36.51	32.88
Letter	14.44	14.20	20.18	16.46	14.12

Es ist wieder zu sehen, dass der Fuzzy-Regellerner erhebliche Probleme unter Verwendung der $\text{Yager}_{1/2}$-Norm (−5.94%) hat; der Generalisierungsfehler ist signifikant schlechter als für die anderen verwendeten Normen. Weiterhin liegt die Luka-Norm (−0.19%) knapp unter dem Durchschnitt. Die Minimum/Maximum- (1.92%) und Produkt-Norm (1.98%) sind besser als der Durchschnitt. Bessere Generalisierungsfähigkeiten werden nur mit der Yager_2-Norm (2.20%) erzielt.

Bereichsbasierte Konfliktlösung

Im letzten Test wird die sogenannte bereichsbasierte Konfliktlösungsstrategie verwendet. Die Ergebnisse sind in Tabelle 3.4 aufgezeigt. Es lassen sich sehr ähnliche

Tabelle 3.4: Generalisierungsfehler (in %) auf den acht Benchmark-Datensätzen unter Verwendung der bereichsbasierten Konfliktlösungsstrategie für die fünf Normen – Min/Max, Produkt, $\text{Yager}_{1/2}$, Luka, Yager_2.

Datensatz	Norm				
	Min/Max	Produkt	$\text{Yager}_{1/2}$	Luka	Yager_2
Diabetes	32.29	31.90	28.91	29.56	27.21
Aust. Cred.	18.84	18.55	16.67	18.83	17.10
Vehicle	32.98	32.98	46.34	38.06	31.56
Segment	3.90	3.85	9.57	4.85	4.16
Shuttle	0.06	0.06	0.08	0.06	0.06
SatImage	13.75	13.80	24.55	19.50	14.75
DNA	32.72	32.97	36.59	36.93	31.96
Letter	14.36	14.28	20.18	15.66	14.92

Tendenzen wie in dem vorangegangenen Test der kernbasierten Strategie erkennen, wobei die $\text{Yager}_{1/2}$-Norm wiederum durchschnittlich die schlechtesten Ergebnisse liefert. Die Luka-Norm (−0.22%) findet man ebenfalls knapp unter dem Durchschnitt. Die besten Ergebnisse erreicht die Yager_2-Norm (2.02%), knapp gefolgt von der Minimum/Maximum-Norm (1.56%) und Produkt-Norm (1.64%).

Zusammenfassung der Konfliktlösungen

Abbildung 3.1 fasst die gewonnenen Ergebnisse grafisch zusammen. Es zeigt sich, dass die Wahl der Strategie nur einen geringen Einfluss auf die Generalisierungsfähigkeit

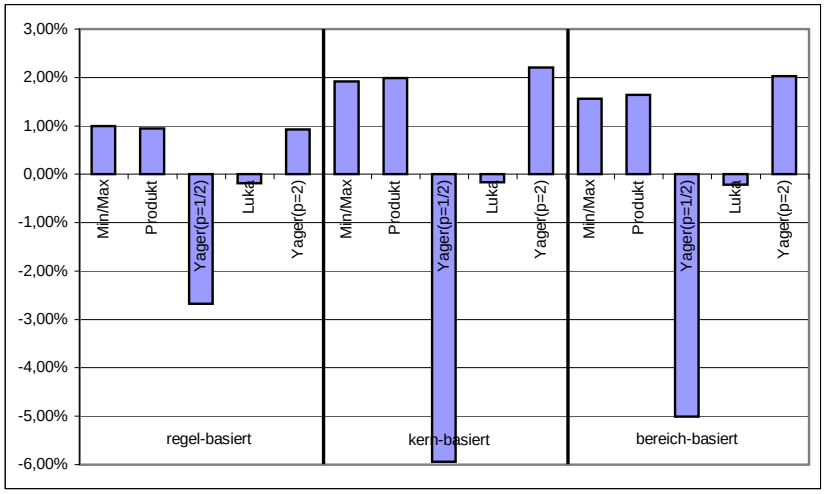


Abbildung 3.1: Prozentuale Abweichungen der Ergebnisse vom mittleren Fehler über alle Datensätze bei Aufteilung in die drei Konfliktlösungsstrategien – regel-, kern- und bereichsbasiert.

des Modells hat. In allen drei Kategorien ergibt sich nahezu das gleiche Bild in Bezug auf die Bewertung des Fehlers zum mittleren Fehler.

In allen Tests schneidet die $\text{Yager}_{1/2}$-Norm am schlechtesten ab; die Luka-Norm liegt ebenfalls unter dem Durchschnitt. Für die verbleibenden Normen lassen sich nur geringfügige Abweichungen in der Generalisierungsfähigkeit feststellen. Dennoch gibt große Unterschiede in den einzelnen Gruppen. Das Diagramm zeigt, dass der Fehler in der kern- und bereichsbasierten Konfliktlösungsstrategie größer ist als in der regelbasierten, bei der die Testergebnisse insgesamt schlechter ausfallen. Damit ist die regelbasierte Strategie auf diesen Datensätzen stabiler, liefert aber keine besseren Resultate. Alle anschließenden Untersuchungen der hierarchischen Erweiterung werden basierend auf die bereichsbasierte Strategie durchgeführt.

3.3 Ergebnisse verschiedener Fuzzy-Normen

In diesem Abschnitt wird der Fuzzy-Regellerner mit den fünf Normen, wie in Kapitel 2.3 vorgestellt, getestet. Es soll untersucht werden, ob diese einen Einfluss auf den Generalisierungsfehler haben, und ob es einen Zusammenhang zwischen den verwendeten Normen und Konfliktlösungsstrategien gibt. Die nachfolgenden Absätze zeigen die Ergebnisse bei der Aufteilung in die Minimum/Maximum-, Produkt-, $\text{Yager}_{1/2}$-, Luka- und Yager_2-Norm. Die Zahl in Klammern repräsentiert die durchschnittliche

Abweichung des Fehlers vom Mittelwert in dieser Gruppe.

Minimum/Maximum-Norm Diese Norm liefert für die meisten Datensätze bessere Ergebnisse unter Verwendung der kern- (2.46%) und bereichsbasierten (2.32%) Konfliktlösungsstrategie. Signifikant schlechtere Ergebnisse erzielt die regelbasierte Strategie (−4.79%).

Produkt-Norm Ähnlich wie bei der Minimum/Maximum-Norm fallen die Ergebnisse auch hier aus; am schlechtesten wiederum ist die regelbasierte Strategie (−4.96%). Durchschnittlich bessere Ergebnisse werden mit der kern- (2.55%) und der bereichsbasierten Konfliktlösungsstrategie (2.41%) erreicht.

Yager$_{1/2}$-Norm Unter Verwendung dieser Norm sind die Klassifikationsergebnisse durchweg schlechter im Vergleich zu den anderen Normen. Gute Ergebnisse können noch mit der bereichsbasierten Konfliktlösungsstrategie (1.50%) erreicht werden; regel- (−1.21%) und kernbasierte Strategie (−0.29%) liegen beide unter dem Durchschnitt.

Luka-Norm Die Luka-Norm zeigt das gleiche Bild wie die Minimum/Maximum- und Produkt-Norm. Die regelbasierte Konfliktlösung (−3.83%) liegt mit Abstand weiter unter dem Durchschnitt; kern- (1.71%) und bereichsbasierte (2.12%) schneiden besser ab.

Yager$_2$-Norm Bei dieser Norm sind die Ergebnisse wieder eindeutig. Die regel-basierte Strategie (−5.27%) liegt weit unter dem Durchschnitt; dagegen sind die kern- (2.59%) und bereichsbasierte Konfliktlösung (2.46%) überdurchschnittlich gut.

Zusammenfassung der Fuzzy-Normen

In der Abbildung 3.2 sind die mittleren Abweichungen nach Normen gruppiert aufgetragen. Auch hier gibt es keine Unterschiede bezüglich der Konfliktlösungsstrategien in den einzelnen Gruppen. Es stellt sich heraus, dass die regelbasierte schlechtere Ergebnisse liefert als die beiden anderen Strategien. Weitere Tests werden auf Basis der Minimum/Maximum-Norm durchgeführt, da diese einfach zu berechnen ist und stabiler gegenüber den anderen zu sein scheint.

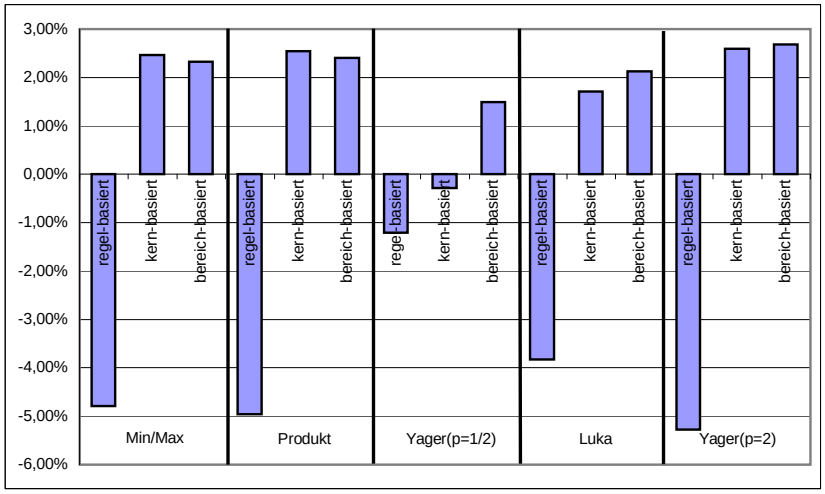


Abbildung 3.2: Prozentuale Abweichungen der Ergebnisse vom mittleren Fehler über alle Datensätze bei Aufteilung in die fünf Normen – Minimum/Maximum, Produkt, $\text{Yager}_{1/2}$, Luka und Yager_2.

3.4 Ergebnisse auf den Benchmark-Datensätzen

Die Tabelle 3.5 zeigt die Ergebnisse (entnommen Michie u. a. (1994); Berthold und Diamond (1998)) im Vergleich zum k-Nächste–Nachbarn Verfahren, Multi-Layer-Perzeptron und Entscheidungsbaumverfahren $c4.5$ Quinlan (1993). Des Weiteren wird der Fuzzy-Regellerner mit dem Mixed–Rule–Learner (MRL, Berthold (2003)) und Probabilistischen Neuronalen Netzen (PNN) verglichen, da der zugrunde liegende lokale Lernalgorithmus (vorgestellt in Berthold und Diamond (1998)) mit dem hier verwendeten verwandt ist.

Vergleicht man den Fuzzy-Regellerner mit den anderen Verfahren, stellt sich heraus, dass es in der Ansammlung von verschiedenen Verfahren keinen optimalen Algorithmus gibt, der auf alle Datensätze beste Ergebnisse liefert. Vielmehr passt die zugrunde liegende Struktur des einen oder anderen Verfahrens mal mehr, mal weniger zu dem jeweiligen Datensatz. Das erschwert es, ein bestimmtes Verfahren zu favorisieren. Dennoch hat sich gezeigt, dass der Fuzzy-Regellerner auf einigen Datensätzen respektable Ergebnisse im Vergleich zu anderen Verfahren des StatLog-Projekts liefert. Im Folgenden wird auf die Ergebnisse der einzelnen Datensätze eingegangen und Vor- und Nachteile der Verfahren diskutiert.

Tabelle 3.5: Klassifikationsfehler (in %) auf den Testdaten des StatLog-Projekts unter Verwendung des FRL, MRL, PNN, *k*NN, *c*4.5 und MLP. Für den Fuzzy-Regellerner und für alle verwendeten Verfahren des StatLog-Projekts werden jeweils das beste und das schlechteste Ergebnis angegeben.

Datensatz	FRL				StatLog-Verfahren				
	best	worst	MRL	PNN	best	worst	*k*NN	*c*4.5	MLP
Diabetes	26.3	32.3	27.7	24.1	22.3	32.4	32.4	27.0	24.8
Aust. Cred.	16.7	18.8	19.1	16.1	13.1	20.7	18.1	15.5	15.4
Vehicle	29.9	46.3	33.9	29.9	15.0	55.8	27.5	26.6	20.7
Segment	3.9	13.8	4.0	3.9	3.0	45.5	7.7	4.0	5.4
Shuttle	0.06	0.09	0.03	0.12	0.01	6.7	0.44	0.10	0.43
SatImage	13.8	29.7	14.3	8.9	9.4	28.7	9.4	15.0	13.9
DNA	32.0	36.9	28.6	16.4	4.1	33.9	14.6	7.6	8.8
Letter	14.1	29.5	11.5	6.4	6.4	59.4	6.8	13.2	32.7

Pima Indians Diabetes Daten

Bei diesem Datensatz liegen der PNN mit 24.10% und das Multi-Layer-Perzeptron mit 24.80% an der Spitze, gefolgt vom FRL mit 26.30%. Da aber hier mit 12-facher Kreuz-Validierung gearbeitet wird und die Eingabedaten nur 768 Muster umfassen, ist der Vorsprung statistisch kaum relevant. Der MRL (27.70%) und das Entscheidungsbaumverfahren *c*4.5 (27.00%) liegen knapp dahinter, aber immer noch besser als das *k*-Nächste-Nachbar Verfahren mit 32.40%.

Australian Credit Approval

Auf diesen Daten belegt der FRL mit 16.67% einen mittleren Platz. Der *k*-Nächste-Nachbar Algorithmus liegt mit 18.10% hinter dem FRL und selbst das Entscheidungsbaumverfahren *c*4.5 erreicht nur 15.50%. Das Multi-Layer-Perzeptron ist mit 15.40% ebenfalls unwesentlich besser als der FRL. Der PNN (16.10%) liegt knapp vor dem FRL. Bei Testmengen mit je 69 Mustern sind allerdings Abweichungen im Prozentbereich statistisch nicht relevant.

Vehicle Silhouettes Daten

Auf diesen Daten erreicht der FRL eine durchschnittliche Erkennungsrate (29.91%), wie auch das PNN mit 29.90%. Der MRL (33.90%) liegt knapp hinter dem FRL. Das beste der hier behandelten Verfahren ist auf diesem Datensatz klar das Multi-Layer-Perzeptron mit 20.70%. *k*-Nächster-Nachbar Verfahren (27.50%) und Entscheidungsbaumverfahren *c*4.5 (26.60%) sind etwas besser als der FRL. Das bedeutet, dass dieser Datensatz Strukturen aufweisen muss, die sich gut mithilfe von sogenannten Hyperebenen trennen lassen. Dieses wird ebenfalls durch die gute Klassifikationsleistung des MLP unterstrichen.

Image Segmentation Daten

Die besten Ergebnisse bei diesem Datensatz liegen im Bereich von 3% bis 4% durchschnittlicher Fehler. Der FRL schneidet mit 3.85% sehr gut ab und liefert sogar das beste Ergebnis für die hier vorgestellten Verfahren. Somit liegt das verwendete Verfahren knapp vor dem PNN (3.90%), MRL (4.00%) und dem Entscheidungsbaumverfahren *c*4.5 (4.00%). Das Multi-Layer-Perzeptron (5.40%) und der *k*-Nächste-Nachbar Verfahren (7.70%) klassifizieren nur geringfügig schlechter.

Shuttle Landing Control Daten

Auf diesem Datensatz schneidet der Mixed-Rule-Learner mit einem Fehler von 0.03% am besten ab; direkt gefolgt von dem Fuzzy-Regellerner mit 0.06%. Dieses lässt sich damit begründen, dass sich zwei Klassen in diesem Datensatz perfekt durch eine achsenparallele Gerade trennen lassen, wie sie durch die rechteckigen Regeln der beiden Verfahren erzeugt werden. Geringfügig schlechter ist das Entscheidungsbaumverfahren *c*4.5 (0.10%) und der klassische PNN-Algorithmus mit 0.12%. In Michie u. a. (1994) kann man in einer zwei-dimensionalen Darstellung der Daten über dem ersten und letzten Attribut sehen, dass die Daten zweier Klassen entlang einer Trenngeraden extrem dicht beieinander liegen. Dieser Umstand ist natürlich günstig für Verfahren wie dem Entscheidungsbaumverfahren *c*4.5, welches auf Trenngeraden beruht, wie sie vom FRL und MRL durch rechteckige Regeln erzeugt werden, wodurch eine klare Trennung der Daten ermöglicht wird.

Landsat-Satellite Daten

Auf diesem Datensatz liegt der FRL mit 13.75% im guten Mittelfeld. Besser sind nur noch der PNN (8.90%) und *k*-Nächste-Nachbar Verfahren (9.40%). Knapp hinter dem FRL, aber noch im akzeptablen Bereich, ist der MRL (14.30%), Multi-Layer-Perzeptron (13.90%) und *c*4.5 (15.00%).

DNA Sequence Daten

Auf diesem Datensatz sind beide Fuzzy-Regellerner – FRL (31.96%) und MRL (28.60%), Schlusslicht aller im StatLog-Projekt getesteten Verfahren; was erstens an der hohen Dimensionalität des Eingaberaumes liegt (wobei 75% der Merkmale theoretisch nutzlos sind und zusätzlich noch zufällig verrauscht sind Michie u. a. (1994)) und zum anderen eine zufällig arbeitende Heuristik zum Verkleinern der rechteckigen Prototypen benutzt wird. Da die Merkmalswerte nur aus 0 oder 1 bestehen und somit der Volumenverlust in jeder Dimension gleich ist, wird der Algorithmus in die Irre geführt. Das bedeutet, dass das Lernverfahren abhängig von der Implementierung ist; das Verfahren kann die erste, die letzte oder eine zufällige Dimension auswählen. Ein weiterer Nachteil ist die verwendete Euklidische Distanz, die dazu führt, dass für fast jedes Eingabemuster eine eigene Regel angelegt wird. Mit anderen Worten: Die Daten werden auswendig gelernt. Wie man in Tabelle 3.5 sehen kann, liefern das Entscheidungsbaumverfahren *c*4.5 (7.60%) und das Multi-Layer-Perzeptron (8.80%) die besten Ergebnisse; bessere können nur noch von klassischen statistischen Verfahren (6%) erreicht werden. Im Mittelfeld findet man den PNN (16.40%) und das *k*-Nächste-Nachbar Verfahren (14.60%).

Letter Recognition Daten

Mit einem Fehler von 14.24% auf den Testdaten liegt der FRL im Vergleich zu den anderen Klassifikatoren des StatLog-Projekts im Mittelfeld. Bessere Ergebnisse erreicht das *k*-Nächste-Nachbar Verfahren (6.80%) und das Entscheidungsbaumverfahren *c*4.5 (13.20%). Neuronale Netze liefern schlechtere Ergebnisse. Das Multi-Layer-Perzeptron erreicht lediglich 32.70%. Interessant ist die herausragende Klassifikationsleistung des klassischen PNN mit 6.40%. Bei diesem Datensatz sind alle Attribute skaliert und scheinen ähnlich wichtig für die Klassifikation zu sein. Ein Indiz dafür ist die gute

Leistung des kNN. Da der FRL nur lokale Fuzzy-Zugehörigkeitsfunktionen erzeugt, profitiert auch er von der Skalierung der Daten und davon, dass alle Attribute Informationen tragen. Die große Anzahl von verwendeten Zugehörigkeitsfunktionen deutet auf einen verrauschten Datensatz hin, was ebenfalls die Probleme des Entscheidungsbaumverfahrens $c4.5$ erklärt.

3.5 Zusammenfassung und Ergebnisse

In diesem Kapitel wurden ein bekanntes Verfahren für das Training von Fuzzy-Regelsystemen auf acht Benchmark-Datensätzen validiert. Dabei wurde dieses auf verschiedene Fuzzy-Normen und Konfliktlösungsstrategien untersucht. Der Einfluss dieser Parameter führt zu sehr gemischten Klassifikationsergebnissen. Die nichtparametrisierten Fuzzy-Normen – Minimum/Maximum und Produkt – liefern oft bessere Ergebnisse und zeigen oft niedrigere Fehlklassifikationswahrscheinlichkeiten als die Funktionen aus der Familie der Yager-Normen, wobei die $\text{Yager}_{1/2}$-Norm die schlechtesten Ergebnisse erzielt. Mit der Yager_2-Norm werden ebenfalls gute Ergebnisse erreicht. Bei den drei getesteten Strategien zeigt sich ein gleichförmiges Bild. Die beiden kern- und bereichsbasierten Konfliktlösungen schneiden besser ab als die regelbasierte, die zum Teil erhebliche Defizite in der Modellgüte aufweist. Diese Nachteile lassen sich damit erklären, dass der Volumenverlust der kompletten Regel berechnet wird, um die Shrink-Operation auszuführen. Das führt oftmals zur Erzeugung von unsymmetrischen Regeln bezüglich des Ankerpunktes.

Im Vergleich zu anderen Standard-Lernverfahren erzeugt der Fuzzy-Regellerner gute Ergebnisse und ist damit ein geeignetes Verfahren für unsere weiteren Untersuchungen. Auf der einen Seite erreicht man gute Klassifikationsergebnisse, auf der anderen Seite leidet aber das Verständnis der Regelbasis oftmals unter der großen Anzahl von Regeln. Bevor in Kapitel 5 die hierarchische Erweiterung des klassischen Fuzzy-Regellerners beschrieben wird, werden in Kapitel 4 hierarchische Modelle eingeführt und eine Formalisierung für regelbasierte Ansätze definiert. Diese wird dann aufgegriffen, um unseren hierarchischen Regellernansatz entsprechend einzuordnen und zu diskutieren. In Kapitel 6 wird basierend auf den Benchmark-Daten eine Evaluierung des hierarchischen Verfahrens vorgenommen.

Kapitel 4

Hierarchische Modelle

Dieses Kapitel beschreibt Ansätze des Lernens hierarchischer Modelle. Diese haben aufgrund ihrer graduellen Struktur wesentliche Vorteile gegenüber Standard-Ansätzen, wenn die Verständlichkeit sowie Explorationsfähigkeit des Modells im Vordergrund stehen. Im nachfolgenden Kapitel 5 wird eine spezielle Erweiterung zum Lernen hierarchischer Fuzzy-Regelsysteme betrachtet und diskutiert. Es wird gezeigt, wie sich dieser Ansatz auf lokale Regellernverfahren verallgemeinern lässt, um hierarchisch, schichtenbasierte Regelsysteme zu erzeugen.

Beim menschlichen Lernen verwendet man möglichst eine mehrstufige Strategie, bei der von einer einfachen zu einer komplexeren Ebene aufgestiegen wird; im Gegensatz zu einem einstufigen Verfahren, bei dem versucht wird, das gesammelte Wissen in einem Gesamtkonzept zu vereinen. Eine der grundlegenden Eigenschaften im menschlichen Lernen und im Problemlösen ist die Fähigkeit, die Welt in unterschiedlichen Detaillierungsgraden zu erklären und leicht zwischen den Abstraktionsebenen zu übersetzen. Computer können komplexe Probleme im Allgemeinen kaum abstrahieren. Das ist einer der Gründe, warum Menschen dem Computer beim Lernen weit überlegen sind. Komplexe Zusammenhänge können schnell verstanden werden, weil Informationen parallel verarbeitet werden und das Lernen neuer Dinge durch einfache Regeln hierarchisch approximiert wird.

Hierarchisches Lernen kann als das Finden eines Modells verstanden werden, in dem von niedrigeren Abstraktionsniveaus zu einem höheren Hypothesenraum übersetzt wird. Da es häufig einen großen semantischen Abstand zwischen den niedrigen Eigenschaftsräumen und den Konzepträumen gibt, ist es im Allgemeinen schwierig und aufwendig, ein geeignetes Modell zu finden. Häufig bedarf es vieler Trainings-

muster und/oder Trainingsepochen, um dem zugrunde liegenden Konzept so nahe wie möglich zu kommen. Um die Komplexität beim maschinellen Lernen zu reduzieren, ist es möglich, die Eigenschaften des menschlichen Lernens zu adaptieren.

In der Natur läuft das Lernen auf vielen Abstraktionsebenen ab. Ein Kind lernt zuerst auf unterster Ebene einfache motorische oder sprachliche Fähigkeiten. Wenn diese ausreichend trainiert und verfeinert sind, werden sie dauerhaft gespeichert und können später jederzeit abgerufen und angewendet werden. Erst später wird auf Grundlage des einfachen Wissens, komplexeres Wissen gelernt und gespeichert. Hierbei baut komplexes Wissen immer auf einfachem Wissen auf. Jede gelernte Fähigkeit wird zu einem Modul gekapselt und kann Bedarf schnell abgerufen werden. Module auf den oberen Ebenen nutzen dieses Wissen der darunter liegenden Einheiten, um schnell neue Fähigkeiten zu erwerben. Komplexe Aktionen können zügig ausgeführt werden, da Basiswissen direkt abgefragt werden kann, ohne das dieses Grundwissen erst erworben werden muss. Das Lernen auf mehreren Ebenen wird als hierarchisches Lernen bezeichnet, welches im Folgenden auf das Lernen von hierarchischen Modellen übertragen werden soll.

4.1 Lernen hierarchischer Modelle

Im Bereich des Data Minings existieren Verfahren, die hierarchische Modelle automatisch auf Daten erlernen. In diesem Abschnitt werden zwei unterschiedliche Methoden, Entscheidungsbäume und hierarchisches Clustering, vorgestellt und diskutiert. Diese Verfahren generieren Modelle, die im Zusammenhang mit Lernverfahren zur Erzeugung von hierarchischen Modellen interessant sind.

Entscheidungsbaum-Verfahren

Ein populäres Verfahren, das hierarchische Modelle erzeugt, sind Entscheidungsbäume (Breiman u. a., 1984; Quinlan, 1993; Murthy, 1997). Diese generieren für den Anwender leicht verständliche Modelle und sind relativ einfach zu bedienen. Entscheidungsbaum-Verfahren versuchen, die Wahrscheinlichkeitsverteilung der Trainingsdaten der verschiedenen Klassen zu approximieren. Der Trainingsalgorithmus induziert hierbei eine Baumstruktur auf Basis einer globalen Partitionierung des Eingaberaumes. Der Eingaberaum wird beginnend beim Wurzelknoten durch eine immer

feiner werdende Unterteilung der Merkmale partitioniert.

Die durch ein Entscheidungsbaumverfahren gelöste Lernaufgabe besteht aus einer Menge X von numerischen und diskreten Merkmalen E, den Trainingsbeispielen, denen jeweils eine Klasse aus der Menge Y zugeordnet ist. Sei $\mathcal{R}$ die Menge aller auf einem Datensatz $\mathcal{T}$ konstruierbaren Entscheidungsbäume auf der Attributmenge $\mathcal{E}$, dann wächst die Anzahl möglicher Bäume exponentiell mit der Anzahl der Elemente der diskreten Attribute E, weil in jedem Pfad eines Entscheidungsbaumes die Attribute in unterschiedlicher Reihenfolge vorkommen können. Daher ist es nicht praktikabel, alle möglichen Entscheidungsbäume zu durchsuchen und zu vergleichen. Stattdessen wird der Baum schrittweise beginnend beim Wurzelknoten aufgebaut. Unter Verwendung eines Maßes für den Informationsgewinn, der entsteht, wenn eine bestimmte Attributaufteilung vorgenommen wird, durchsucht man nur einen kleinen Teil des ansonsten sehr viel größeren Suchraums aller Hypothesen. Hierfür wird jeweils für die Aufteilung eines Attributes der größte, lokale Gewinn bezüglich eines Attributauswahlmaßes bestimmt. Hierbei werden einfache und kurze Hypothesen gegenüber langen und detaillierteren bevorzugt. Deswegen nennt man solche Verfahren auch „greedy“ (gierig), weil sie während der Konstruktion für den aktuellen Zeitpunkt den besten Splitt auf den Trainingsdaten sofort ausführen.

Entscheidungsbäume lassen sich aufgrund ihrer Struktur leicht in Regelsysteme oder sogenannte Entscheidungslisten (Rivest, 1987) überführen und können deswegen auch gut als flaches Regelmodell interpretiert werden. Hierbei wird jeder Pfad im Baum zu einer eigenen Regel ausgehend vom Wurzel- zum Blattknoten konvertiert. Der induzierte Regelbaum stellt eine Hierarchie unter der Menge der Merkmale dar, siehe Abbildung 4.1. Ein weiteres Detail, welches oft im Zusammenhang zur Verbesserung der Interpretierbarkeit von Entscheidungsbäumen diskutiert wird, sind Stutz- (oder engl. „Pruning“)-Verfahren (Furnkranz, 1997). Diese Methode, die „unwichtige“ Zweige entfernt, verkleinert den Baum, sodass dieser bezüglich der Validierungsdaten weiterhin einen generellen Klassifikator darstellt. Durch Löschen dieser Teile gehen wiederum Detailinformationen verloren, die helfen können, das komplette Konzept hinter den Daten zu erklären. Wiederum wird deutlich, dass hier ein Kompromiss zwischen Genauigkeit und Interpretierbarkeit gefunden werden muss. Es wird zwar ein generelleres Modell erzeugt, welches aber einen niedrigeren Detailgrad mit geringerer Ausdrucksstärke in Bezug auf den Hypothesenumfang besitzt.

Die Entscheidungsbaumlerner sind Vertreter für Verfahren, die ein hierarchisches

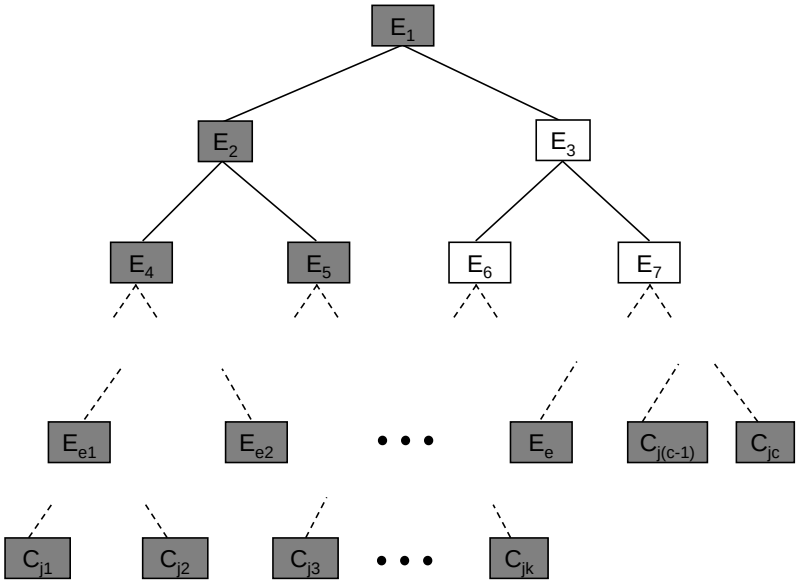


Abbildung 4.1: Modell eines Entscheidungsbaumes wie es typischerweise durch einen Lerner induziert wird. In den ersten $n-1$ Schichten sieht man mögliche binäre Aufteilungen auf der Attributmenge $\mathcal{E}$, sowie auf der untersten Ebenen die Vorhersage der Klassen C.

Modell erzeugen. Diese generieren basierend auf einer baumartigen Aufteilung der Attribute ein globales Modell, welches durch eine Menge von Regeln interpretiert werden kann. Teile dieses Baumes können leicht entfernt werden, um das Modell zu generalisieren. Dadurch ergibt es sich eine variable, hierarchische Regelstruktur.

Hierarchische Clustering-Verfahren

Eine weitere Familie von Verfahren sind die hierarchischen Clustering-Verfahren (Jain und Dubes, 1988). Diese nicht-überwachten Ansätze sind zwar eher untypisch in diesem Bereich, sollen aber aufgrund ihrer hierarchischen Modellstruktur hier besprochen werden. Allgemein erzeugen Clustering-Verfahren Gruppen von Objekten mit ähnlichen Eigenschaften. Bei den hierarchischen Clustering-Verfahren werden nacheinander mehrere Clustering-Lösungen generiert. Diese Serie von ineinander geschachtelten Clustern bestehen auf der ersten Ebene aus den einelementigen Ob-

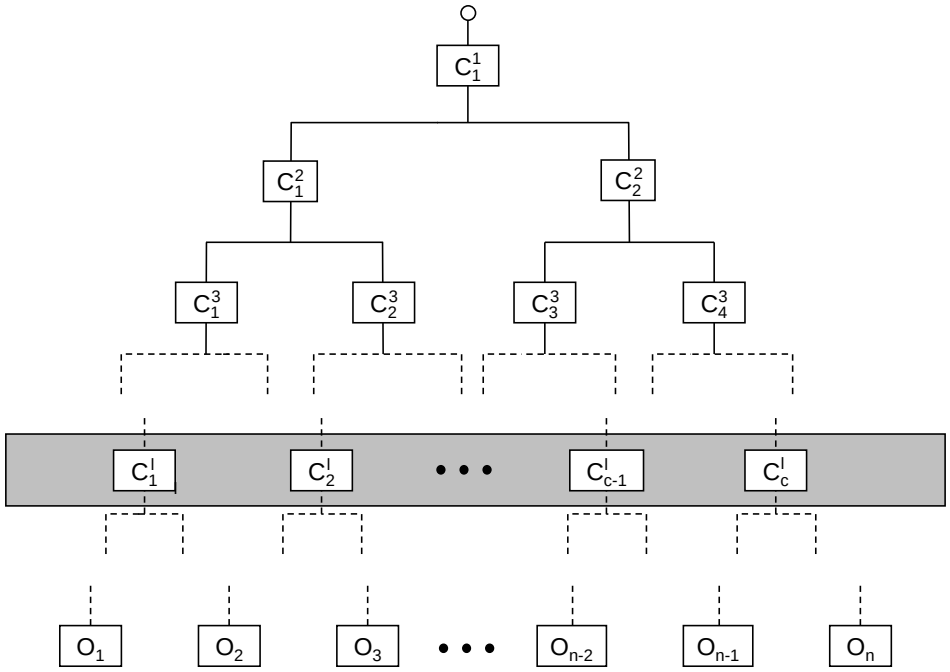


Abbildung 4.2: Das Diagramm zeigt schematisch ein hierarchisches Modell, wie es von einem agglomerativen Clustering-Verfahren generiert wird. Auf unterster Schicht befinden sich die zu clusternden Objekte O_1 bis O_n. Diese werden sukzessive zu Clustern zusammengefasst. Durch Schneiden des Dendogramms in beliebiger Höhe, kann eine beliebige Menge von Cluster extrahiert werden.

jekten, den Trainingsdaten. Weiter oben werden immer paarweise zwei Objekte, die dann wieder Cluster sein können, zusammengefasst bis am Ende ein einzelner Cluster alle Objekte vereint. Dies geschieht dadurch, dass in jedem Schritt des Verfahrens die beiden einander ähnlichsten Cluster zu einem neuen Cluster zusammengefasst werden. Die Ähnlichkeit zweier einelementiger Cluster kann dabei einfach durch die Ähnlichkeit ihrer beiden einzelnen Objekte bestimmt werden. Werden zwei Cluster zusammengefasst, wird die Ähnlichkeit des neuen Clusters zu allen existierenden Clustern berechnet; es gewinnen immer die beiden Objekte oder Cluster, die am ähnlichsten zueinander sind. Das Ergebnis dieses sogenannten agglomerativen Clusterings ist eine Hierarchie von ineinander geschachtelten Clusterings.

Diese Hierarchie von möglichen Clustern kann in einem Dendrogramm dargestellt werden, in dem man Gruppen von ähnlichen Objekten erkennen kann, um eine optimale Anzahl von Clustern zu bestimmen, siehe Abbildung 4.2. Diese hierarchische

Struktur hat den Vorteil, dass in jeder Ebene der Hierarchie ein nicht-hierarchisches Clustering mit der gewünschten Anzahl von Clustern abgeleitet werden kann. Die Hierarchie von ähnlichen Paaren von Instanzen kann aufsteigend durch verschiedene Methoden des agglomerativen Clusterings erzeugt werden: Single-Linkage (minimaler Abstand), Complete-Linkage (maximaler Abstand) und Average-Linkage Clustering (gemittelter Abstand).

Aufgrund der generierten hierarchischen Modellstruktur lassen sich Clustering- und Entscheidungsbaumverfahren ebenfalls der Gruppe der hierarchischen Lerner zuordnen. Im Mittelpunkt der Untersuchungen stehen weiterhin Regellernverfahren, auf deren Grundlagen hierarchische Modelle generiert werden sollen.

4.2 Hierarchische Regelmodelle

In diesem Abschnitt werden Regellernverfahren beschrieben, die Ausreißer durch geeignete Modelle abbilden können. Es werden Verfahren diskutiert, die sich mit dem Lernen von Ausnahmeregeln beschäftigen. Diese Erweiterung von Regellernverfahren wird näher beleuchtet, bevor anschließend hierarchische Fuzzy-Regelmodelle betrachtet werden.

Die Idee des Lernens von Ausnahmeregeln, wie sie von Dejean u. a. (2002) und Kivinen u. a. (1994) verfolgt wird, basiert auf dem Ansatz, Ausreißer in Daten explizit zu modellieren, sodass eine Regelbasis möglichst klein und überschaubar bleibt. Regeln auf höheren Schichten haben Vorrang gegenüber Regeln auf den unteren Niveaus, die auch als Ausnahmen gesehen werden können. Damit ergibt sich ein Regelmodell, in dem Knoten entweder die Vorbedingung einer Regel repräsentieren oder eine bestimmte Klasse vorhersagen. Die Kanten innerhalb dieser Struktur können als Übergang zwischen Bedingung und Folgerung der Regel verstanden werden.

In Gras und Kuntz (2006) wird ein weiterer Ansatz zum Lernen strukturierter Regelbasen vorgeschlagen. Gras' Ansatz basiert auf einer gerichteten Hierarchie mit graduellen Schichten von Regeln. Knoten innerhalb dieser Hierarchie sind Regeln, deren Folgerungen wieder Regeln sein können und damit als Verschachtelung von Regeln gesehen werden können. In Scheffer (1995) wird ein Ansatz vorgestellt, der hierarchisch verschachtelte Bedingungen kombiniert und so eine Hierarchie basierend auf dem Merkmalsraum erzeugt.

$$
\begin{array}{lllllll}
R_1: & \texttt{IF} & x_1 \in E_{1,1} & \times \cdots \times & x_n \in E_{n,1} & \texttt{THEN} & R'_1 \\
R'_1: & \texttt{IF} & x_1 \in E'_{1,1} & \times \cdots \times & x_n \in E'_{n,1} & \texttt{THEN} & y_1 \\
 & & \vdots & & \vdots & & \vdots \\
R_{r_1}: & \texttt{IF} & x_1 \in E_{1,r_1} & \times \cdots \times & x_n \in E_{n,r_1} & \texttt{THEN} & R'_{r_1} \\
R'_{r_1}: & \texttt{IF} & x_1 \in E'_{1,r_1} & \times \cdots \times & x_n \in E'_{n,r_1} & \texttt{THEN} & y_{r_1} \\
 & & \vdots & & \vdots & & \vdots \\
R_j: & \texttt{IF} & x_1 \in E_{1,j} & \times \cdots \times & x_n \in E_{n,j} & \texttt{THEN} & R'_j \\
R'_j: & \texttt{IF} & x_1 \in E'_{1,j} & \times \cdots \times & x_n \in E'_{n,j} & \texttt{THEN} & y_j \\
 & & \vdots & & \vdots & & \vdots \\
R_{r_c}: & \texttt{IF} & x_1 \in E_{1,r_c} & \times \cdots \times & x_n \in E_{n,r_c} & \texttt{THEN} & R'_{r_c} \\
R'_{r_c}: & \texttt{IF} & x_1 \in E'_{1,r_c} & \times \cdots \times & x_n \in E'_{n,r_c} & \texttt{THEN} & y_c.
\end{array}
$$

Damit erweitert sich unsere Definition einer flachen Regelbasis, in der die Folgerung wieder eine Regel sein kann; statt eine bestimmt Klasse vorherzusagen. Die Hypothesensprache ist weiterhin gegeben in Form von Regeln, die jetzt Ausnahmen modellieren können. Die Hypothesensprache wird dadurch erweitert und kann abstraktere Konzepte leichter, aber auf Kosten eines komplexeren Regelsystems, beschreiben. Diese kann also nicht nur klassische, achsenparallele Rechtecke darstellen, sondern auch innerhalb einer Regel wiederum Regeln beinhalten. Damit ergibt sich ein erster Ansatz strukturierter Regelmodelle oder einfacher Regelhierarchien, die im Folgenden genauer analysiert werden sollen.

Die genannten Ansätze generieren strukturierte Modelle mit unterschiedlicher Komplexität der Hypothesen. Eine Hierarchie, wie sie im Folgenden betrachtet wird, kann beliebig komplexe Konzepte abbilden. Jedes Modell in der Hierarchie genügt einer einfachen Hypothesensprache. Das hierarchische System wird hierbei von einem Standardlerner generiert. Dieses System erklärt alle Feinheiten der Daten, bleibt aber dennoch generell, um verständlich und interpretierbar zu sein. Hierfür muss eine eindeutige Beziehung zwischen den Regeln existieren, damit das Modell visuell exploriert werden kann. Basierend auf der Definition eines einfachen Regelmodells, werden nun erweiterte Regellernmethoden diskutiert. Diese Regellerner versuchen, durch Modellierung von Ausnahmen, dem Problem von umfangreichen Regelbasen zu begegnen; generieren aber mitunter zu viele Details, um – basierend auf einer

einfachen Hypothesensprache – verständlich zu sein.

Eine mögliche Erweiterung der einfachen Regellernverfahren ist eine variable Granulierung der Regelbasis. Diese kann gerade bei den partitionierenden Methoden dadurch erreicht werden, dass in Bereichen, in denen komplexe Konzepte abgebildet werden sollen, feinere Partitionierungen der Eingabedimensionen verwendet werden. Einen Vorschlag, Fuzzy-Modelle hierarchisch zu lernen, findet man in Nozaki u. a. (1996) und Ishibuchi u. a. (1992).

In diesen Ansätzen werden Fuzzy-Modelle unterschiedlicher Granulierung erzeugt. Als Initialisierung nutzt man eine grobe Granulierung des Eingaberaumes; gewöhnlich zwei Fuzzy-Zugehörigkeitsfunktionen für jedes Merkmal. Das resultierende Regelsystem wird im nächsten Schritt mit feinerer Partitionierung der Merkmale trainiert. Als Ergebnis erhält man ein mehrschichtiges Regelsystem. Im letzten Schritt werden alle Regeln mit niedriger Aktivität entfernt und es verbleiben nur die Regeln im System, die eine hohe Korrektheit auf den Trainingsdaten aufweisen. Will man in diesen Systemen schließen, so müssen die Aktivitäten des zu klassifizierenden Musters über alle Schichten addiert werden. Der maximale Wert der summierten Zugehörigkeitsgrade ergibt die finale Klassenausgabe. Der Nachteil dieses Verfahrens ist, dass die Anzahl der anzupassenden Regeln exponentiell mit der Granulierung des Merkmalsraumes über die Anzahl der Dimensionen steigt. Das hat zur Folge, dass die Interpretierbarkeit der Modelle stark abnimmt, da unter Umständen viele Regeln erzeugt werden, je feiner die Dimensionen partitioniert sind. Das gelernte Regelsystem besteht am Ende aus $|L|$ Regelmodellen $\mathcal{R}^l$ mit aufsteigender Granulierung der Eingabe. Insgesamt werden über L Level $2^2 + 3^2 + \cdots + L^2 = \frac{1}{6}L(L+1)(2L+1) - 1$ Regeln erzeugt, d. h., die Anzahl der Regeln wächst kubisch über die Anzahl der Schichten. Diese Anzahl kann zwar durch Löschen unnötiger Regeln noch reduziert werden, dennoch entstehen durch die globale Granulierung der Eingabe eine Vielzahl hierarchisch-überlappender Fuzzy-Regeln.

In Kivinen u. a. (1992) wird ein Algorithmus zum Lernen eines hierarchischen Klassifizierers beschrieben. Dieser besteht aus einer Menge von Regeln, die in Schichten organisiert sind, wobei jede Schicht eine bestimmte Klassenausgabe generiert. Regeln bestehen hierbei aus einem Konzept, dem Antezedens und der Konklusion der jeweiligen Schicht in der Hierarchie. Regeln weiter oben in der Hierarchie haben Vorrang gegenüber Regeln in den unteren Schichten, die als Ausnahmeregeln interpretiert werden können. In Skowron u. a. (2005) wird ein weiterer hierarchischer Ansatz vor-

gestellt, der den Problemen der Überlappung und ungenügenden Überdeckung von Modellen in Multi-Klassifikationsproblemen begegnet. Die erzeugte mehrschichtige Struktur besteht aus einer Serie von Modellen mit unterschiedlicher Granularität.

4.3 Zusammenfassung

Die beschriebenen Ansätze verwenden zwar einfache Hypothesensprachen, erzeugen aber oftmals komplexe Modelle, da die Struktur während des Lernens aufgeweicht wird. Gerade die angesprochenen Regelmodelle leiden häufig darunter, dass sie viele Regeln enthalten, um zum Beispiel alle Ausnahmen und Artefakte in den Daten zu modellieren. Die vorgestellten Methoden versuchen, diesem Problem zu begegnen, indem sie komplexere Hypothesen verwenden. Hierdurch erhöht sich auf der anderen Seite die Komplexität des Modells in Bezug auf die Struktur der Regelbasis. Damit verringert sich die Verständlichkeit des Systems, die aber gerade notwendig ist, um das Modell später zu verstehen und visuell zu explorieren.

Diesem Problem wird im anschließenden Kapitel 5 begegnet. Es wird ein allgemeiner Ansatz vorgestellt, um schichtenbasierte Modelle zu erzeugen. Diese Hierarchien von Modellen genügen in jeder Ebene der gleicher Hypothesensprache unter Verwendung der ursprünglich gegebenen Regelstruktur. Die Aufteilung in Modelle mit unterschiedlichen Detailgraden, ermöglicht es nun, das Gesamtmodell graduell und ebenenübergreifend zu verstehen und zu explorieren. Es wird eine Erweiterung des Fuzzy-Regellerners aufgezeigt, die basierend auf einer Filterheuristik, Ebenen von Regelmodellen unterschiedlicher Detailgrade generiert. Die hierarchischen Regelmodelle genügen in jeder Schicht der gleichen, einfachen Hypothesensprache; analog zu der ursprünglichen Definition von flachen Regelmodellen. Sie sind dennoch ausdrucksstärker, weil sie Ausnahmen in separaten Modellen erklären, die aber durch ihre Ebenenbeziehung zum hierarchischen Gesamtmodell gehören und so im Verbund mit allen Modellen in der Hierarchie nun verstanden werden können.

Kapitel 5

Hierarchische Fuzzy-Regelmodelle

Dieses Kapitel beschäftigt sich mit dem Lernen hierarchischer Fuzzy-Regelmodelle und greift den Fuzzy-Regellerner aus Kapitel 2.4 auf, um hierauf die Idee des hierarchischen Regellernens zu erweitern. Dieser Ansatz wurde in Kapitel 4 vorgestellt und soll nun auf ein konkretes Lernverfahren angewendet werden.

5.1 Erzeugung von Regelhierarchien

Eine Hierarchie von Regelmodellen vereint die Eigenschaften eines modularen Systems, welches auf der einen Seite alle Feinheiten erklärt und auf der anderen Seite gleichzeitig grob das Konzept hinter den Daten skizziert. Damit wird das zugrunde liegende Konzept über die Ebenen der Hierarchie graduell abstrahiert und kann so auf allen Ebenen eigenständig exploriert werden.

Eine Regelhierarchie kann als ein mehrschichtiges System verstanden werden, in dem auf niedrigen Ebenen alle Details der Daten beschrieben werden. In den Schichten darüber findet man Verallgemeinerungen, die es erlauben, von dem eigentlichen Problem zu abstrahieren. Hieraus lässt sich (neues) Wissen ableiten, um eine konkrete Aufgabenstellung, aber auch neue Probleme effektiv lösen zu können. Diese Wissenshierarchie beschreibt damit Konzepte durch einen variablen Grad der Abstraktion.

Im Allgemeinen ist eine Hierarchie ein System von Elementen, die einander über- bzw. untergeordnet sind. Die darauf definierte Ordnungsrelation stellt einen gerichteten azyklischen Grafen dar, in dem die Elemente eine Wertigkeit oder Rangordnung besitzen. In einer Hierarchie von Regelmodellen beschreibt jede Ebene genau eine

Regelbasis, in der die Knoten Regeln sind, die über eine wohldefinierte Beziehung ebenenübergreifend miteinander verknüpft sind.

In Anlehnung an unsere Definition eines Regelsystems siehe 2.1, wird nun ein hierarchisches Regelsystem betrachtet, das ausgehend von einer Regelbasis $\mathcal{R}$ definiert wird. Diese enthält Regeln $R^l_1, \ldots, R^l_j, \ldots, R^l_{r_c}$ mit $l \in L$. Damit kann jedem Level l eine Anzahl von Regeln R^l_j zugeordnet werden. Es ergibt sich das komplette Regelsystem $\mathcal{R} = \{\mathcal{R}^1, \ldots, \mathcal{R}^l, \ldots, \mathcal{R}^L\}$ mit $l \in \mathcal{L}$:

$$
\begin{array}{lllllllll}
R^1_1: & \texttt{IF} & x_1 \in E^1_{1,1} & \times & \cdots & \times & x_n \in E^1_{n,1} & \texttt{THEN} & y_1 \\
 & & \vdots & & & & \vdots & & \vdots \\
R^1_j: & \texttt{IF} & x_1 \in E^1_{1,j} & \times & \cdots & \times & x_n \in E^1_{n,j} & \texttt{THEN} & y_j \\
 & & \vdots & & & & \vdots & & \vdots \\
R^1_{r_1}: & \texttt{IF} & x_1 \in E^1_{1,r_c} & \times & \cdots & \times & x_n \in E^1_{n,r_c} & \texttt{THEN} & y_c \\
 & & \vdots & & & & \vdots & & \vdots \\
\hline
 & & \vdots & & & & \vdots & & \vdots \\
R^l_1: & \texttt{IF} & x_1 \in E^l_{1,1} & \times & \cdots & \times & x_n \in E^l_{n,1} & \texttt{THEN} & y_1 \\
 & & \vdots & & & & \vdots & & \vdots \\
R^l_j: & \texttt{IF} & x_1 \in E^l_{1,j} & \times & \cdots & \times & x_n \in E^l_{n,j} & \texttt{THEN} & y_j \\
 & & \vdots & & & & \vdots & & \vdots \\
R^l_{r_c}: & \texttt{IF} & x_1 \in E^l_{1,r_c} & \times & \cdots & \times & x_n \in E^l_{n,r_c} & \texttt{THEN} & y_c \\
 & & \vdots & & & & \vdots & & \vdots \\
\hline
 & & \vdots & & & & \vdots & & \vdots \\
R^L_1: & \texttt{IF} & x_1 \in E^L_{1,1} & \times & \cdots & \times & x_n \in E^L_{n,1} & \texttt{THEN} & y_1 \\
 & & \vdots & & & & \vdots & & \vdots \\
R^L_j: & \texttt{IF} & x_1 \in E^L_{1,j} & \times & \cdots & \times & x_n \in E^L_{n,j} & \texttt{THEN} & y_j \\
 & & \vdots & & & & \vdots & & \vdots \\
R^L_{r_c}: & \texttt{IF} & x_1 \in E^L_{1,r_c} & \times & \cdots & \times & x_n \in E^L_{n,r_c} & \texttt{THEN} & y_c
\end{array}
$$

In Kapitel 2 wurde ein konstruktiver Trainingsalgorithmus vorgestellt, der Fuzzy-Regeln automatisch aus Daten lernen kann. Die erzeugten Regelbasen können unter Umständen viele kleine Detailregeln enthalten, um das zugrunde liegende Konzept vollständig zu beschreiben. Fuzzy-Regeln erlauben es zwar, unscharfe Klassengrenzen zu modellieren, benötigen aber dennoch viele Regeln, wenn zum Beispiel komplexe

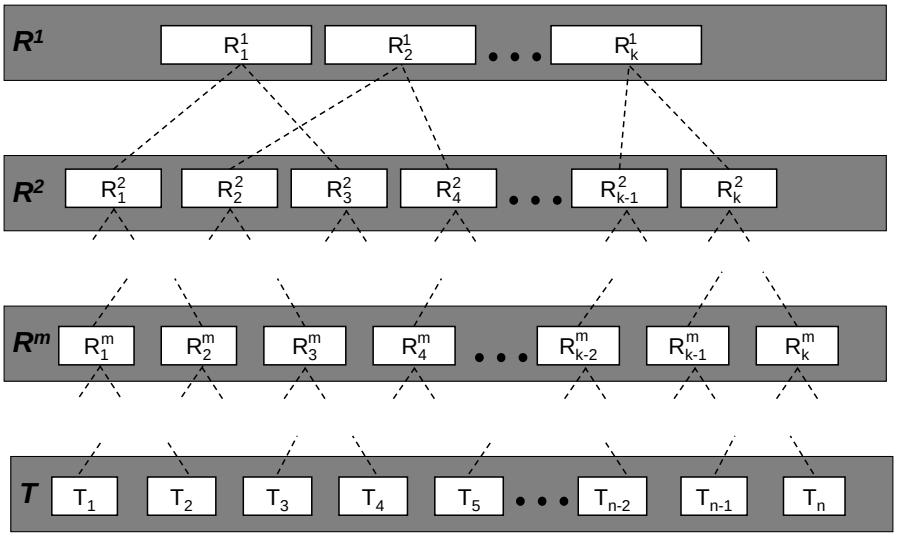


Abbildung 5.1: Struktur einer Regelhierarchie, wie sie durch einen hierarchischen Regellernansatz induziert wird.

Konzepte bestehend aus Artefakten und vielen Details approximiert werden sollen. Hierarchische Regellernansätze versuchen diesem Problem zu begegnen. Im Speziellen werden hierarchische Fuzzy-Regelsysteme diskutiert, die durch Aufspaltung einer komplexen Regelbasis beliebig komplexe Konzepte durch einfache Regeln abbilden können. Sind in unserem Ansatz die Elemente der Hierarchie Hypothesen in Form von Regeln, ergibt sich ein mehrschichtiges System von Einzelmodellen. In diesem System beschreiben Regeln auf niedriger Ebene zwar das Konzept, sind aber dennoch meist zu komplex, um für einen Experten nützlich zu sein.

Das Beispiel in Abbildung 5.1 zeigt schematisch die Hierarchieebenen $\mathcal{R}^1, \ldots, \mathcal{R}^m$ mit den jeweiligen Regeln $R_1, \ldots, R_k$. Diese Regeln sind schichtenweise verknüpft. Die unterste Schicht verweist direkt auf die Eingabedaten $\mathcal{T}$ mit $T_1, \ldots, T_n$. Anstatt flacher Regelmodelle wird nun diese Struktur weiter verwendet, um auf Basis eines einfach Lernverfahrens diese automatisch zu induzieren.

Ziel ist es, auf Basis eines lokalen Trainingsalgorithmus eine Hierarchie von Modellen zu generieren, die eine graduelle, hierarchische Exploration des Modells erlaubt und gleichzeitig gute Klassifikationsfähigkeiten besitzt. Dem Anwender wird

die Möglichkeit gegeben zu entscheiden, zu welchem Grad der Abstraktion er das Modell betrachten möchte, um die für ihn relevanten Details zu extrahieren.

Es werden zwei Varianten von Modellhierarchien betrachtet, die sich direkt aus der Idee des Filterns von Detailmodellen ableiten lassen. Die generierten Hierarchien unterscheiden sich nicht in der Konstruktion, die in beiden Varianten auf Basis der Detailmodelle geschieht. Beide Möglichkeiten können für die Generierung der Hierarchie verwendet werden. Die Hierarchie wird jeweils durch die Modelle gebildet, die nach Filtern unter Verwendung der Detailmodelle erhalten bleibt. Es sollen beiden Ansätze vorgestellt werden, um die resultierenden Hierarchien näher zu analysieren.

Im Folgenden wird das Filterverfahren zum Konstruieren von Regelhierarchien beschrieben. Es werden Möglichkeiten zur Erzeugung einer Regelhierarchie und Filterheuristiken aufgezeigt, an denen beispielhaft die Funktionsweise des Verfahrens erläutert wird.

5.2 Lernen hierarchischer Fuzzy-Regelmodelle

Die Regellernverfahren generieren oftmals viele Detailregeln, die aber eine Interpretation des Modells erschweren. Es ist wünschenswert, kleine Regeln zu extrahieren, um sie anschließend in einem Detailmodell zu erklären. Dieses Detailmodell dient dann als Filter für die Trainingsdaten, um ein generelleres System aus den verbliebenen Daten zu generieren. Dieses Modell beinhaltet weniger Regeln und bleibt somit interpretierbar, da „größere“ Bereiche des Eingaberaumes zusammengefasst werden. Führt man dieses Verfahren weiter fort, erhält man eine Hierarchie, in der die unteren Modellschichten alle Feinheiten der Daten erklären, dazu aber möglicherweise viele kleine Regeln benötigen. In den oberen Schichten besteht das Modell aus nur wenigen generelleren Regeln, die dadurch aber bessere explorative Eigenschaften bieten.

Im folgenden Abschnitt sollen die Möglichkeiten zur Erzeugung einer Regelhierarchie beschrieben werden. Der hierarchische Ansatz erzeugt auf Basis des in Kapitel 2 vorgestellten Regellerners ein hierarchisches Fuzzy-Regelmodell (Gabriel und Berthold, 2003), welches aufgrund seiner Struktur beliebig komplexe Konzepte durch einfache Fuzzy-Regeln beschreiben kann.

Die verwendete Methodik basiert auf dem in Berthold (2000) vorgestellten Ansatz. Dieser geht von einem Fuzzy-Regelmodell aus, wie es im vorigen Kapitel beschrieben wurde. Im ersten Schritt wird ein Detailmodell extrahiert, wobei alle Muster entfernt

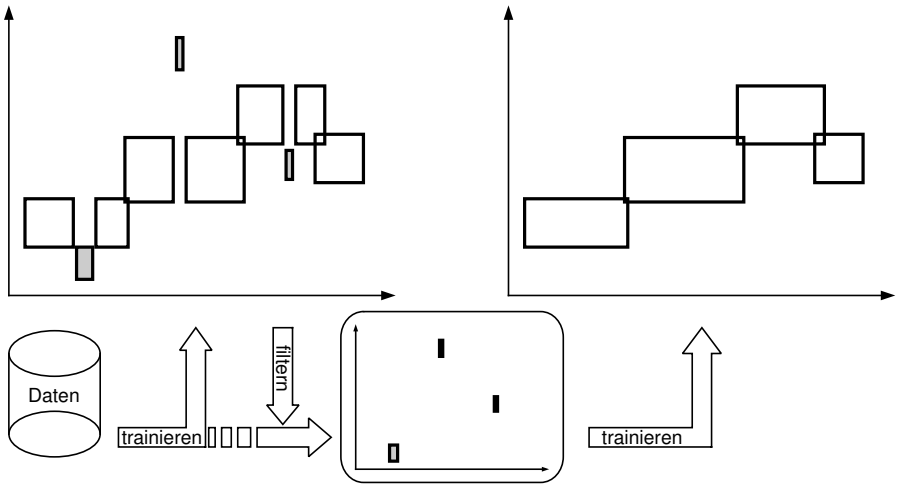


Abbildung 5.2: Im ersten Schritt wird ein Regelmodell aus den Eingabedaten erzeugt. Aus diesem wird anschließend das Detailmodell extrahiert, das im nächsten Schritt als Filter verwendet wird, um ein generelleres Modell zu trainieren; Abbildung nach Berthold (2000).

werden, die unter eine gegebene Relevanzschranke fallen. Dieses Modell wird anschließend als Filter benutzt, um seltene Detailmuster aus den Daten zu entfernen. Im nächsten Schritt wird mit den verbliebenen Daten ein generelleres Modell gelernt. In Abbildung 5.2 ist dieses Vorgehen verdeutlicht. Im ersten Schritt der Modellerzeugung wird mithilfe der gesamten Eingabedaten ein Regelmodell trainiert (oben links). Dieses Modell enthält drei „kleinere“ Regeln, die grau hinterlegt sind, und einige „größere“ Regeln [1]. Diese Regeln werden im nächsten Schritt aus dem Modell extrahiert und gehen in das Filtermodell ein (Mitte unten). Dieses Modell wird anschließend im zweiten Trainingsschritt zum Filtern der Daten verwendet, d. h., weniger Daten gehen in das nächste Modell ein (oben rechts). Dieses Modell enthält weniger Regeln, die aber größere Bereiche des Eingaberaumes abdecken. Am Ende dieses zweistufigen Prozesses existieren zwei Modelle: ein spezielles Modell mit kleinen Regeln und ein generelleres mit wenigen großen Regeln.

Der Algorithmus beschreibt das generelle Vorgehen zur Erzeugung einer Hierarchie von Regelmodellen. Nachdem die Trainingsdaten T zum Zeitpunkt $i = 0$ initialisiert wurden, wird in jeder Iteration ein Regelmodell trainiert, Zeile (4). Auf dieser Regelbasis wird anschließend der Ausreißerschwellwert θ_{outlier} angewendet, um Regeln mit niedriger Relevanz zu extrahieren, siehe Zeile (5). Im nächsten Schritt

[1] Unter der Größe einer Regel wird hier die Fläche des Eingaberaumes verstanden, die von der Regel erklärt wird. In höherdimensionalen Räumen ist das Hypervolumen einer Regel relevant.

werden alle verbliebenen Trainingsmuster, die unterhalb der Aktivierungsgrenze von θ_{filter} liegen, eliminiert. Dieses Verfahren wird so lange fortgesetzt, bis keine Regeln mehr unterhalb des Ausreißerschwellwerts liegen. Danach wird die nächste Regelebene generiert. Damit bilden alle Regeln des zuletzt generierten Modells die oberste Schicht in der Hierarchie. Die resultierende Hierarchie enthält am Ende der Trainingsprozedur alle Ausreißermodelle sowie die Regeln aus der obersten Ebene.

ALGORITHM hFRL

(1) $T_0 = T$

(2) $i \leftarrow 0$

(3) `REPEAT`

(4) $\quad \mathcal{R}_i = \text{FRL}(T_i)$

(5) $\quad \mathcal{R}_i^{\text{outlier}} = \{R \in \mathcal{R}_i \mid \Phi(R) \leq \theta_{\text{outlier}}\}$

(6) $\quad T_{i+1} = T_i \setminus \left\{(\vec{x}, k) \in T_i \mid \exists R \in \mathcal{R}_i^{\text{outlier}} : \mu_R(\vec{x}) \geq \theta_{\text{filter}}\right\}$

(7) $\quad i \leftarrow i + 1$

(8) `WHILE` $|\mathcal{R}_{i-1}^{\text{outlier}}| > 0$

Abbildung 5.3 zeigt ein typisches Beispiel der Erzeugung einer Regelhierarchie, die aus einem zweidimensionalen Datensatz mit drei Klassen (rot, grün und blau) erzeugt wird. Das im ersten Schritt generierte Regelmodell *Untere Ebene* wird basierend auf dem Filterkriterium gefiltert. Hierbei werden alle Regeln in das Ausreißerregelmodell *Ausreißerebene 1* übernommen, die nach einem bestimmten Filterkriterium für die aktuelle Modellebene „uninteressant“ sind. Die restlichen Regeln verbleiben im Modell *Stabile Ebene 1*. Das Ausreißermodell wird anschließend verwendet, um die Trainingsdaten zu filtern. Hierbei werden alle Muster entfernt, die von Regeln des Ausreißermodells abgedeckt werden. Mit den verbliebenen Daten wird dann im zweiten Schritt das nächste Modell generiert. Nach einer weiteren Epoche bleibt am Ende ein finales Modell *Stabile Top-Ebene* erhalten, welches das oberste Modell der Hierarchie darstellt.

Es werden zwei Varianten von Regelhierarchien betrachtet: eine Hierarchie von Detailmodellen (siehe Abbildung 5.4), die auf den unteren Schichten alle Regeln enthalten, um alle Feinheiten, aber auch Ausreißer in den Daten zu beschreiben, und auf der obersten Schicht die Daten sehr allgemein durch nur wenige Regeln erklärt und eine Hierarchie von robusten (generellen) Regelmodellen (siehe Abbildung 5.5), die auf der Basis der Detailmodelle entstehen. Diese Modellhierarchien basieren auf dem

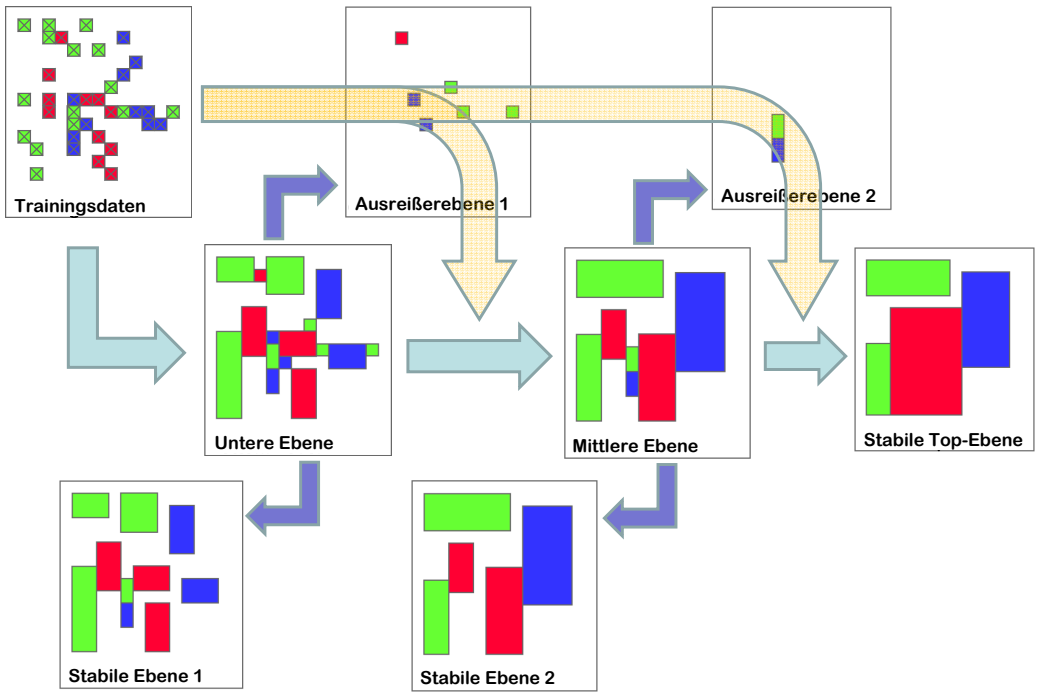


Abbildung 5.3: Beispiel der Erzeugung einer dreischichtigen Regelhierarchie aus Detail- und Grobmodellen: Durch sukzessives Filtern der Trainingsdaten entstehen in jedem Schritt eine Ausreißer- und ein Stabile Ebene. Diese Modelle zusammen mit der Stabilen Top-Ebene werden anschließend zur resultierenden Hierarchie formiert.

Regelmodell jeder Schicht, das nach Filtern mit dem darunter liegenden Detailregeln erhalten bleibt. In beiden Fällen entsteht eine Hierarchie von Regelmodellen, die auf unterschiedliche Art das Konzept der Daten beschreibt. Im ersten Fall erhält man eine Hierarchie von Detailmodellen, die nur wenige Regeln in jeder Schicht aufweisen und

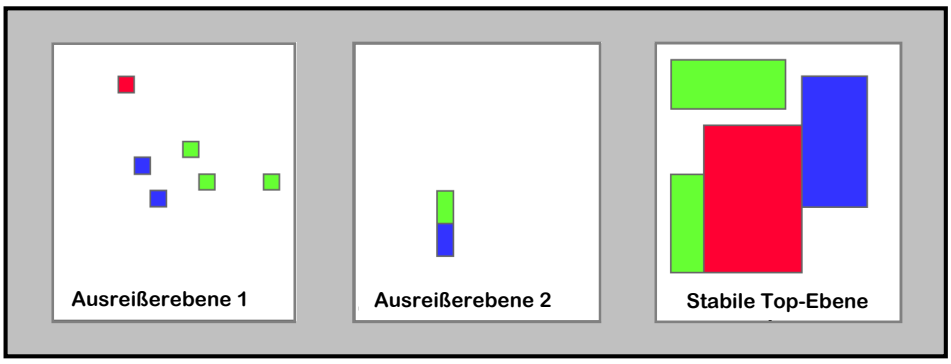


Abbildung 5.4: Modellhierarchie aus Ausreißer-Regelsystemen.

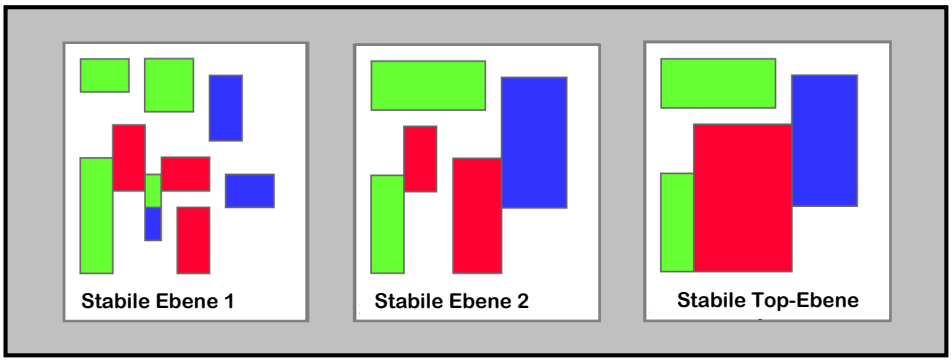


Abbildung 5.5: Modellhierarchie aus stabilen Regelsystemen.

daher für die Visualisierung und Exploration besser geeignet ist. Im zweiten Fall ergibt sich eine Hierarchie von stark überlappenden Modellen, in der bestimmte Bereiche der Daten mehrfach erklärt und dadurch verstärkt werden. Diese Art von Regelmodellen eignen sich besonders für die Klassifikation unbekannter Datenpunkte.

Der beschriebene hierarchische Regellernansatz extrahiert Regeln mit niedriger Relevanz, die aber dennoch wichtige Details des Konzepts beschreiben können und deswegen Teil des Gesamtmodells (der Hierarchie) bleiben. Eine geeignete Strategie zum Filtern von Regeln mit niedriger Relevanz ist daher notwendig. Im Folgenden werden zwei Filterheuristiken beschrieben, um regel- und modellbasiert Regeln aus lokalen Modellen extrahieren zu können.

Filterheuristiken auf Fuzzy-Regelhierarchien

Zur Generierung einer Hierarchie von Regelmodellen ist es notwendig, Regeln bewerten und nicht relevante Regeln filtern zu können. Diese Detail- oder Ausreißerregeln lassen sich über einen Filterschwellwert θ_{outlier} extrahieren:

$$R^{\text{outlier}} = \{R \in \mathcal{R} \mid \Phi(R) \leq \theta_{\text{outlier}}\}.$$

Die extrahierten Regeln werden als Filter für die Daten verwendet, um die nächste Hierarchieebene zu erzeugen. Die so generierte Hierarchie von Filtermodellen besteht aus nur wenigen, allgemeinen Regeln auf den höheren Hierarchiestufen. Weiter unten in der Hierarchie findet man speziellere Regeln, die alle Details der Daten erklären. Um die Trainingsdaten basierend auf einem Detailmodell zu filtern, wird der Schwellwert

der Aktivierung θ_{filter} ($0 \leq \theta_{\text{filter}} \leq 1$) definiert:

$$T^{\text{relevant}} = T \setminus \left\{(\vec{x}, k) \in T \mid \exists R \in \mathcal{R}^{\text{outlier}} : \mu_R(\vec{x}) \leq \theta_{\text{filter}}\right\}.$$

Damit ergibt sich ein Relevanzmaß für die Extraktion von Ausreißerregeln, die nachfolgend verwendet wird, um die regel- und modellbasierte Filterstrategie zu besprechen.

Regelbasierte Filterheuristik

Die hierarchische Filterstrategie entfernt, siehe Zeile (6) des hierarchischen Algorithmus, alle Trainingsmuster, die unterhalb der Ausreißerschranke basierend auf jeder individuellen Regel liegen:

$$T_{i+1} \quad = \quad T_i \setminus \left\{ (\vec{x}, k) \in T_i \mid \exists R \in \mathcal{R}_i \setminus R_i^{outlier} : \mu_R(\vec{x}) \geq \theta_{\text{filter}}\right\}.$$

Das heißt, es werden alle Trainingsmuster entfernt, die von mindestens einer Regel aus dem Detailmodell (besteht aus Ausreißerregeln) abgedeckt werden. Die Abbildung 5.6 zeigt ein zweidimensionales Beispiel mit Mustern der zwei Klassen **+** und ●. Die abgebildeten Rechtecke umranden den Kernbereich jeder Regel. In diesem Beispiel resultiert eine dreischichtige Regelhierarchie unter Verwendung des Ausreißerschwellwerts $\theta_{\text{outlier}} = 2$ und einer Aktivierung von $\theta_{\text{filter}} = 1$. Das bedeutet, dass Regeln, die zwei oder weniger Muster abdecken, in das Ausreißermodell verschoben werden. Alle anderen Muster innerhalb des Kernbereichs einer Ausreißerregel (mit

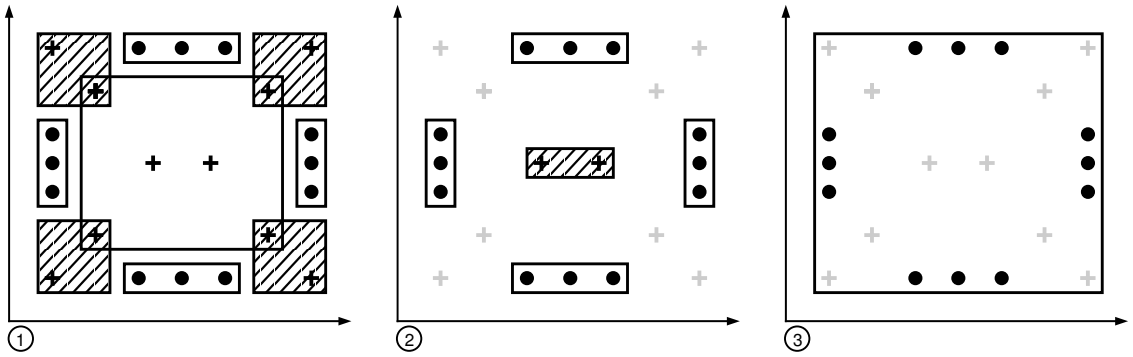


Abbildung 5.6: Zweidimensionales Beispiel eines dreischichtigen, hierarchischen Regelmodells unter Verwendung der regelbasierten Filterstrategie. Gelöschte Muster werden grau und Detailregeln schraffiert dargestellt.

einem Zugehörigkeitsgrad von Eins) werden gelöscht.

Durch den gegebenen Filterschwellwert von 2 werden alle Regeln entfernt, die weniger als zwei Datenpunkte erklären. Hieraus folgt, dass in diesem Beispiel im ersten Schritt vier Regeln ausgewählt werden, die zwei Muster der Klasse **+** in den Eckbereichen enthalten, siehe Bild (1), sodass anschließend das Regelsystem, wie in Bild (2) zu sehen, entsteht. Dabei hat sich die größte Regel verkleinert und deckt nur noch zwei Muster ab. Die anderen Regeln bleiben unverändert erhalten. Im nächsten Schritt werden wiederum alle Muster entfernt, die von Regeln abgedeckt werden, die unter den Grenzwert fallen, sodass am Ende das System in Bild (3) entsteht. In diesem Modell existiert keine Regel, die Eingabemuster der Klasse **+** erklärt, da diese in den vorherigen Schichten entfernt wurden. Vielmehr wird der größte Teil des Eingaberaumes durch eine einzelne Regel der Klasse ● abgedeckt.

Das resultierende Ergebnis stellt am Ende eine nicht zufriedenstellende Approximation auf den Eingabedaten dar. Im nächsten Abschnitt wird man sehen, dass es bei der Auswahl der Muster wichtig ist, das ganze System während der Ausreißerauswahl zu betrachten, um Regeln der Detailmodelle in Bezug auf das Gesamtsystem zu extrahieren.

Modellbasierte Filterheuristik

Der regelbasierte Filteransatz löscht Muster sofort, wenn diese von einer Regel abgedeckt werden, die als Ausreißerregel identifiziert wird. Das hat den Nachteil, dass oftmals Muster entfernt werden, die gleichzeitig auch von Regeln abgedeckt werden, die nicht unterhalb der Filterschranke liegen. Eine daraus folgende Erweiterung ist die modellbasierte Filterheuristik, welche basierend auf dem Gesamtmodell entscheidet, ob ein Muster aus den Trainingsdaten gelöscht wird:

$$T_{i+1} = T_i \setminus \left\{ (\vec{x}, k) \in T_i \mid \exists R \in \mathcal{R}^{outlier} : \mu_R(\vec{x}) \geq \theta_{\text{filter}} \ \wedge \nexists R \in \mathcal{R}_i \setminus \mathcal{R}_i^{outlier} \right\}.$$

Der modellbasierte Ansatz arbeitet ähnlich zu dem oben beschriebenen regelbasierten, löst aber das Problem, dass Muster entfernt werden, die ebenfalls von Nicht-Ausreißerregeln erklärt werden. Zusätzlich zum regelbasierten Ansatz überprüft der modellbasierte, ob die erklärten Muster zusätzlich von einer Nicht-Detailregel abgedeckt werden. Solche Muster werden nicht entfernt, die Detailregel selbst geht aber in das Ausreißermodell ein.

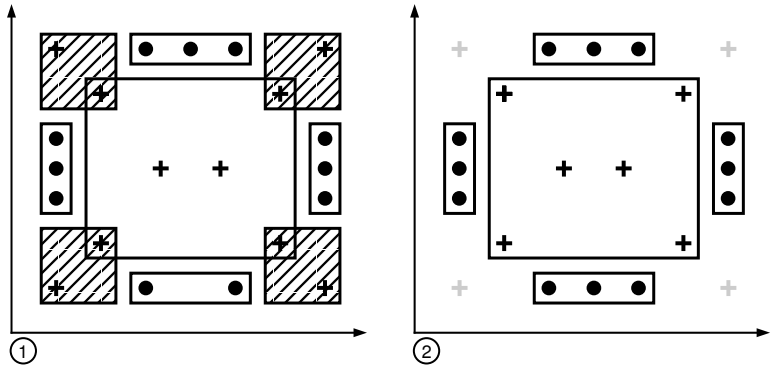


Abbildung 5.7: Zweidimensionales Beispiel eines zweischichtigen hierarchischen Regelmodells unter Verwendung der modellbasierten Filterstrategie. Gelöschte Muster werden grau und Detailregeln schraffiert dargestellt.

Die Abbildung 5.7 zeigt das Beispiel mit den gleichen Daten, aber unter Verwendung der modellbasierten Filterstrategie. Wieder sind die vier Regeln in den Eckbereichen markiert, siehe Bild (1). Nur werden jetzt nicht beide Muster entfernt, sondern nur die Muster, die ausschließlich von Regeln abgedeckt werden, die unter den Support von Zwei fallen. D. h., in diesem Beispiel bleiben die Muster, die zusätzlich von größeren Regeln (fallen nicht unter diese Schranke) abgedeckt werden, erhalten. Im Vergleich zum regelbasierten Filter ergibt sich im Bild (2) eine größere Regel. Weiteres Entfernen von Mustern ist nicht notwendig, da keine Regel mehr unter den definierten Grenzwert fällt. Dieses Vorgehen bezieht das Modell aller Regeln in die Auswertung ein, die nicht unter die Schranke fallen. So wird sichergestellt, dass bezüglich des Gesamtmodells größere, schon existierende Regeln erhalten bleiben.

Evaluierung der Filterheuristik

Die Auswahl dieser Regeln kann auf verschiedene Art und Weise geschehen: zum Beispiel kann man Regeln nach dem Volumen ihrer Hyperquader, nach der Anzahl der erklärten Muster (Anzahl der Muster im Einflussbereich der Regel) oder der Anzahl der Muster im Kernbereich auswählen. Eine andere Möglichkeit ist, das Verhältnis zwischen Anzahl und Volumen zu bilden, um so die Regeln durch die Dichte der Verteilung der Daten zu bewerten. Die weiteren Untersuchungen beziehen sich auf die Anzahl der erklärten Muster als Ausreißerkriterium. Damit geht man davon aus, dass kleinere Regeln eine niedrigere Relevanz für das Gesamtsystem haben als größere. Die Anzahl der Muster, die eine Regel mindestens abdecken muss, um in das Detailm-

odell einzugehen, wird im Folgenden als Ausreißerschwellwert bezeichnet. In diesem Abschnitt soll untersucht werden, welchen Einfluss der gewählte Filterwert auf die Anzahl der generierten Ebenen und Regeln, sowie auf den Klassifikationsfehler hat.

Die Abbildung 5.8 zeigt die Ergebnisse auf den Landsat-Satellite Daten unter Verwendung eines Filterschwellwerts zwischen 1 und 150. Aufgetragen sind die Anzahl der Ebenen, die Regeln in der Top-Ebene und über alle Ebenen sowie den Klassifikationsfehler in % . Interessant ist zu sehen, dass die Anzahl der Regeln (grün) über alle Hierarchieebenen bis zu einem Filterschwellwert von 25 abnimmt und dann relativ konstant weiter verläuft. Gegenläufig hierzu verläuft der Klassifikationsfehler (violett) über alle Ebenen der Hierarchie. Die Anzahl der Hierarchieebenen (rot) sowie die Anzahl der Regeln in der Top-Ebene (blau) verringern sich mit steigenden Filterschwellwert. Bei einem Filterschwellwert von 25 erhält man eine Regelmodellhierarchie, in der mit rund der Hälfte der Regeln eine ähnliche Klassifikationsgüte erreicht wird. Die Regeln sind über 5 Ebenen verteilt, wobei die Top-Ebene nur sehr wenige Regeln im Vergleich zu den Modellen mit niedrigerem Filterwert enthält. Die

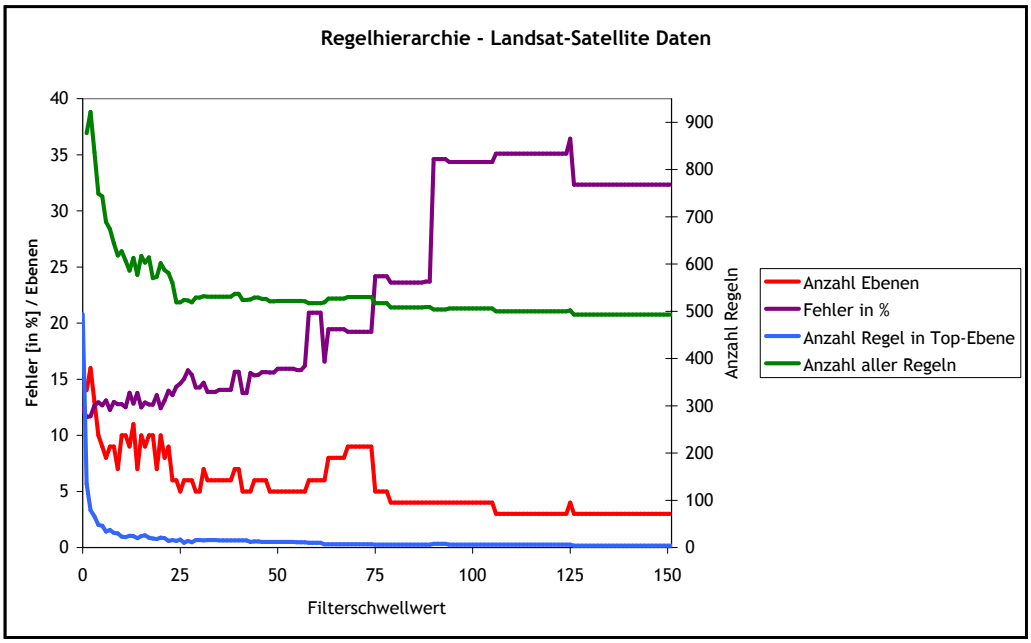


Abbildung 5.8: Test verschiedener Filterwerte (von 1 bis 150) auf den Landsat-Satellite Daten. Im Diagramm sind auf der linken Seite die Anzahl der generierten Ebenen (rot) sowie der Fehler in % (violett) aufgetragen. Auf der rechten Seite sieht man die Anzahl der Regeln auf oberster Ebene (blau) sowie aufsummiert über alle Ebenen (grün).

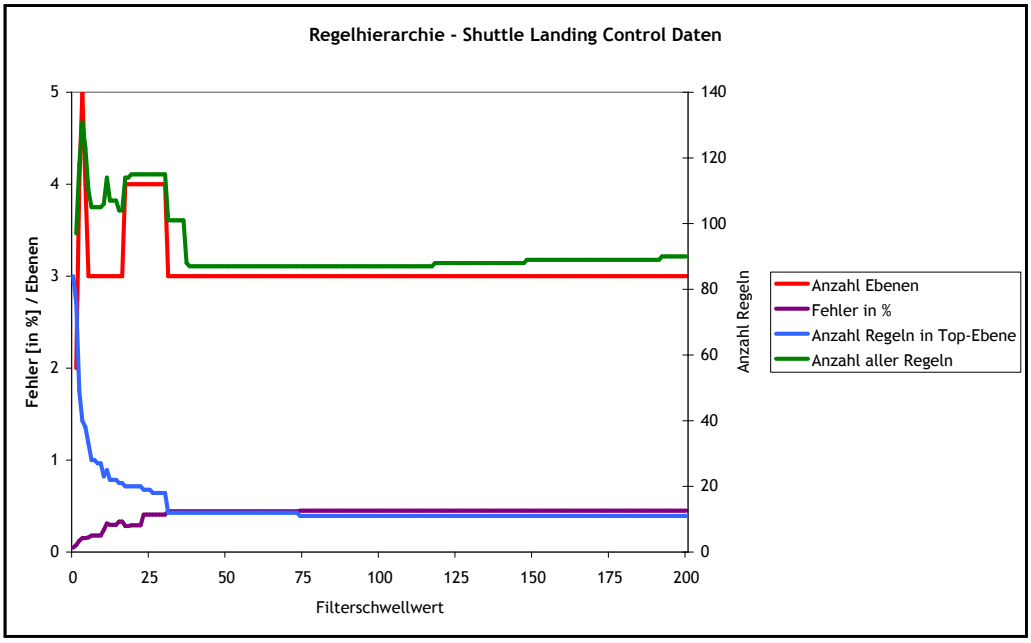


Abbildung 5.9: Test verschiedener Filterwerte (von 1 bis 200) auf den Landsat-Satellite Daten. Im Diagramm sind auf der linken Seite die Anzahl der generierten Ebenen (rot) sowie der Fehler in % (violett) aufgetragen. Auf der rechten Seite sieht man die Anzahl der Regeln auf oberster Ebene (blau) sowie aufsummiert über alle Ebenen (grün).

Regelhierarchie zeigt in diesem Beispiel eine starke Abnahme der Regelanzahl, die über mehrere Ebenen verteilt sind; die Klassifikationsergebnisse über alle Hierarchien sind für einen niedrigen Schwellenwert (< 25) akzeptabel.

In der Abbildung 5.9 sieht man die Ergebnisse auf den Shuttle Landing Control Daten unter Verwendung eines Filterschwellwerts zwischen 1 und 200. Interessant ist zu sehen, dass die Anzahl der Ebenen relativ konstant bei 3 bleibt, aber die Anzahl der Regeln in der Top-Ebene und in der gesamten Hierarchie (um 50%) abnimmt. Betrachtet man den Klassifikationsfehler, sieht man eine Verschlechterung mit wachsendem Filterschwellwert. In diesem Beispiel macht die Generierung der Hierarchie zwar Sinn, weil weniger Regeln erzeugt werden, das aber auf Kosten des Klassifikationsfehlers.

Klassifikation in Fuzzy-Regelhierarchien

Ein weiterer wichtiger Aspekt der hierarchischen Fuzzy-Regelmodelle ist die Klassifikationsfähigkeit solcher Systeme. Da jede Schicht getrennt betrachtet werden kann,

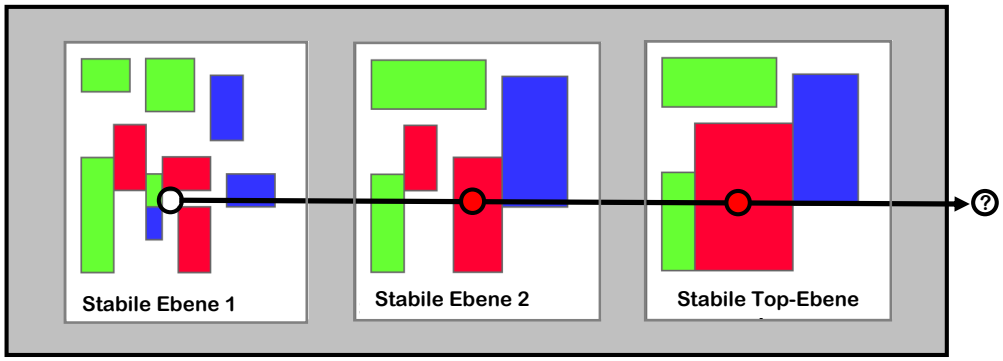


Abbildung 5.10: Beispiel einer Klassifikation in einer dreischichtigen Regelhierarchie mit den Stabilen Ebenen 1 und 2 sowie der Stabilen Top-Ebene.

muss die Klassenzugehörigkeit zu Beginn auf jeder Ebene berechnet werden, siehe hierzu auch 2.3, bevor im nächsten Schritt die finale Klassenausgabe berechnet werden kann. Hierzu werden verschiedene Ansätze betrachtet, die speziell die Fuzzy-Klassenausgaben verschiedener Regelbasen kombinieren.

Die Abbildung 5.10 zeigt beispielhaft ein dreischichtiges, hierarchisches Regelmodell. Die einzelnen Regelmodelle sind von den Stabilen Ebenen 1 (links, L_3) und 2 (Mitte, L_2) zur Stabilen Top-Ebene (rechts, L_1) geordnet. Jede Schicht für sich generiert eine Gesamtklassenausgabe, die es gilt, zu einer finalen Klassenausgabe zusammenzufassen. Im Folgenden sind verschiedene Möglichkeiten der Klassifikation in Regelhierarchien aufgezeigt, um die Fuzzy-Klassenausgabe für ein bestimmtes Muster zu kombinieren. Gegeben sei hierbei eine n-schichtige Regelhierarchie, in der jedes Modell mit L_i im Level i bezeichnet wird:

L_i-Klassifikation Diese Methode klassifiziert Muster anhand der jeweiligen Schicht. Dieses Verfahren stellt also einen guten Test dar, wenn die Leistungsfähigkeit der einzelnen Regelmodelle in jeder Schicht i überprüft werden soll. In unserem Beispiel sagt Level L_1 und Level L_2 Klasse rot und Level L_3 keine spezifische Klasse vorher.

$L_1 \rightarrow L_i$-Klassifikation Diese Art der Klassifikation summiert die vorhergesagten Klassen der Einzelmodelle von L_1 bis L_i in der Hierarchie auf. Die Klasse mit den meisten Stimmen bestimmt die Ausgabe. In diesem Beispiel liefert eine $L_1 \rightarrow L_2$-

oder $L_1 \rightarrow L_3$-Klassifikation Klasse rot.

$L_1[\rightarrow L_i]$**-Klassifikation** Diese bedingte Klassifikation geht so lange in der Hierarchie abwärts, bis ein Modell L_i gefunden wird, das das Eingabemuster klassifiziert. In jenem Beispiel liefert Modell L_1 schon ein Ergebnis, Klasse rot. Diese Art der Klassifikation soll zeigen, inwieweit das oberste Modell einen guten Klassifikator darstellt.

$L_n \rightarrow L_i$**-Klassifikation** Diese Methode führt die Klassifikation vom untersten zum obersten Modell durch. In diesem Fall ist die Klassifikation von L_3 zu L_2 unterschiedlich. Bezieht man die Majoritätsklassenentscheidung wird bei der Klassifikation von L_3 zu L_1 eindeutig Klasse rot vorhergesagt.

$L_n[\rightarrow L_i]$**-Klassifikation** Die bedingte Klassifikation vom untersten zum obersten Modell liefert in dem Beispiel immer die Klasse rot, da L_3 keine Ausgabe generiert. Diese Art der Klassifikation soll zeigen, ob das unterste Modell einen guten Klassifikator darstellt.

Bei den vorgestellten Klassifikationsmethoden wird zwischen der *top-down*- und der *bottom-up*-Klassifikation unterschieden. Diese lassen sich wiederum in bedingte und unbedingte Klassifikation unterteilen. Wobei bei der bedingten Klassifikation das Modell die Klassenausgabe generiert, welches zuerst eine positive Ausgabe erzeugt; und bei der unbedingten, die Klassifikation über die Anzahl der zuvor definierten Schichten durchgeführt wird. Damit lässt sich eine Art der Votierung über alle Hierarchiestufen durchführen. Sollte das Ergebnis nicht eindeutig sein, gibt bei der *top-down*-Methode das oberste, bei der *bottom-up* das unterste Modell die Klassifikation an. Alle diese Ansätze basieren auf der Klassifikation der Einzelmodelle. Diese Klassifikationsmethode kann genutzt werden, um die Leistungsfähigkeit der einzelnen Regelmodelle in jeder Schicht zu beurteilen.

5.3 Zusammenfassung

Der vorgestellte hierarchische Ansatz zum Lernen von Fuzzy-Regelhierarchien basiert auf einer Filterstrategie zum Extrahieren von lokalen Regelmodellen mit niedrigerer Relevanz für das Gesamtsystem. Hierbei wurde eine modellbasierte Erweiterung

gezeigt, die, im Gegensatz zur regelbasierten Strategie, das gesamte Regelmodell betrachtet, um Muster bezüglich einer oberen Schranke zu entfernen. Zusätzlich wurden Möglichkeiten der Klassifikation in einem hierarchischen Fuzzy-System vorgestellt, die in Kapitel 6 auf verschiedenen Benchmark-Datensätzen validiert werden.

Kapitel 6

Experimente: hierarchische Fuzzy-Regelmodelle

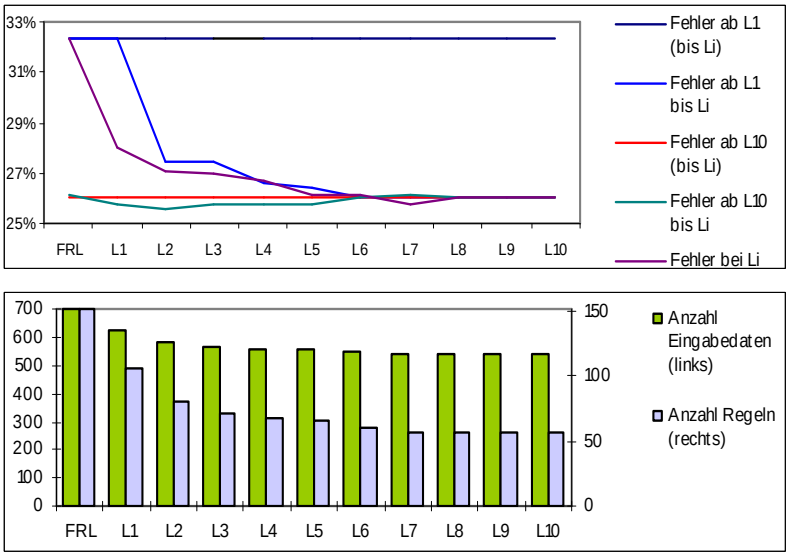
In diesem Kapitel wird der hierarchische Fuzzy-Regellerner auf denselben acht Benchmark-Datensätzen des StatLog-Projekts (siehe Kapitel 3) untersucht, um diesen mit den Ergebnissen aus Kapitel 3 vergleichen zu können. Für die Tests wird die Minimum/Maximum-Norm sowie die bereichsbasierte Konfliktlösungsstrategie (siehe Abschnitt 2.4) verwendet. In Kapitel 5 wurde gezeigt, dass ein Grenzwert von 5 (d. h. Regeln, die 5 oder weniger Muster abdecken) eine akzeptable Anzahl von Schichten im hierarchischen Regelmodell erzeugt. Alle Muster, die nicht im Kernbereich dieser Regeln liegen, werden markiert. Muster, die den Kernbereich dieser Regeln bilden, werden ebenfalls markiert, wenn sie nicht im Kernbereich einer anderen Regel sind, die den Grenzwert von 5 überschreitet. Alle markierten Muster gehen in das Detailmodell ein. Aus den verbliebenen Daten wird anschließend ein neues Modell gelernt. Das Verfahren wird so lange fortgeführt, bis keine Regel und keine Muster mehr die genannten Bedingungen erfüllen. Am Ende erhält man ein mehrschichtiges, hierarchisches Fuzzy-Regelmodell.

Die nachfolgenden Abbildungen und Tabellen zeigen jeweils die Generalisierungsfehler unter Anwendung der verschiedenen Klassifikationsmöglichkeiten in grafischer sowie in tabellarischer Form. Zusätzlich werden in den ersten beiden Spalten die Anzahl der verwendeten Trainingsdaten (Diagramm hellgrün) und die Anzahl der erzeugten Regeln (Diagramm hellblau) angegeben. In Spalte L_i (Diagramm violett) sind die Klassifikationsfehler der einzelnen Modelle (wobei L_i das Modell in Level i bezeichnet) zu sehen. D. h., die Klassifikation wird von den anderen Modellen in der

Hierarchie für jedes Modell L_i getrennt durchgeführt. Spalte $L_1[\rightarrow L_i]$ (Diagramm dunkelblau) zeigt den Generalisierungsfehler bei der Klassifikation vom generellsten Modell L_1 bis zu einem speziellsten Modell L_n. Das bedeutet, dass die Klassifikation eines Musters nur in Modell L_i durchgeführt wird, wenn Modell L_{i-1} das Muster nicht erklärt. Diese Möglichkeit der Klassifikation soll zeigen, dass alle Trainingsmuster in Modell L_1 erklärt sind, d. h., der Eingaberaum ist vollständig durch Fuzzy-Regeln abgedeckt. Die Ergebnisse der Klassifikation der Modelle L_1 bis L_i findet man in Spalte $L_1 \rightarrow L_i$ (Diagramm blau). Spalte $L_n[\rightarrow L_i]$ (Diagramm rot) zeigt prinzipiell die gleichen Verfahren, wobei hier die Klassifikation vom speziellsten Modell L_n bis zu einem generellsten Modell L_i bedingt durchgeführt wird. Hierbei ist es interessant zu sehen, ob Modell L_n allein eine gute Klassifikation erlaubt. Den Generalisierungsfehler der Modelle L_n bis L_i findet man in Spalte $L_n \rightarrow L_i$ dargestellt (Diagramm grün). Einzelheiten zu den Klassifikationsmethoden in Fuzzy-Regelhierarchien befinden sich in Abschnitt 5.2. Um die Klassifikationsgenauigkeit sowie die Regelanzahl direkt mit dem Fuzzy-Regellerner vergleichen zu können, sind jeweils in der obersten Zeile (FRL) jeder Tabelle die Anzahl der Regeln und der Fehler (in %) eingetragen.

Pima Indians Diabetes Daten

Als erstes Beispiel soll der Pima Indians Diabetes Datensatz für die Untersuchungen herangezogen werden. Die Ergebnisse sind in Abbildung 6.1 dargestellt. In diesem Beispiel hat der hierarchische Fuzzy-Regellerner 10 Epochen benötigt, um alle Detailregeln zu extrahieren und die Fuzzy-Regelhierarchie zu erzeugen. Man sieht, wie mit nur 78% der Eingabedaten (Level L_7) eine Regelbasis erzeugt wird, die fast zwei Drittel kleiner (57 Regeln) ist als im FRL-Modell (151 Regeln) und dessen Generalisierungsfähigkeit sich um 6.5% gegenüber dem FRL verbessert hat (siehe L_i-Klassifikation). Weiterhin sei bemerkt, dass alle Modelle in Spalte L_i eine bessere Generalisierungsfähigkeit liefern als das FRL-Modell. In den beiden bedingten Klassifikationen $L_1[\rightarrow L_i]$ und $L_{10}[\rightarrow L_i]$ zeigen sich jeweils die gleichen Ergebnisse in den Hierarchieschichten. Die Klassifikation über mehrere Hierarchieschichten gleichzeitig weist ähnlich gute Ergebnisse auf, teilweise geringfügig bessere als die pure L_i-Klassifikation (siehe Spalte $L_1 \rightarrow L_i$ im Modell L_2). Dieses Beispiel zeigt eine Verbesserung der Generalisierungsfähigkeit des hierarchischen Modells und der Einzelmodelle, sowie eine starke Abnahme der Zahl der erzeugten Regeln bei Verwendung weniger Eingabemuster.

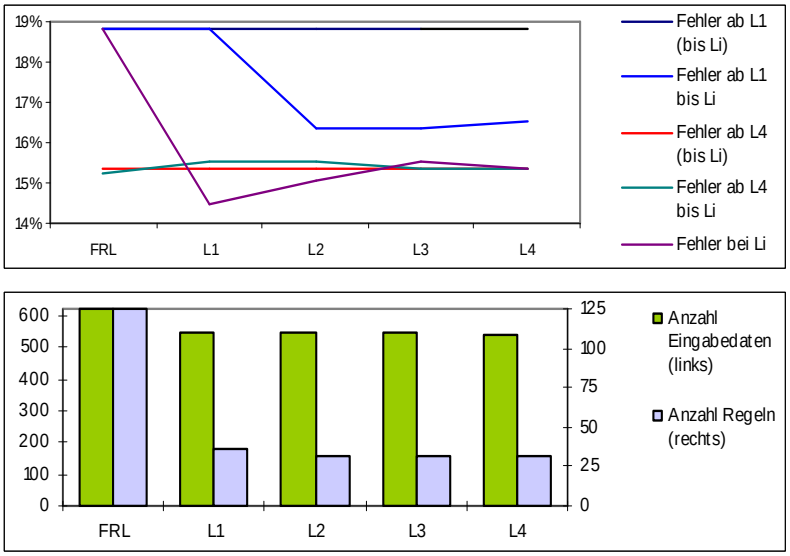


Level	Anzahl		Klassifikationsfehler [in %]				
L_i	Daten	Regeln	L_i	$L_1[\rightarrow L_i]$	$L_1 \rightarrow L_i$	$L_{10}[\rightarrow L_i]$	$L_{10} \rightarrow L_i$
FRL	704	151	32.29				
L_1	627	106	28.00	32.29	32.29	26.04	25.78
L_2	588	80	27.09	32.29	27.47	26.04	25.52
L_3	572	71	26.95	32.29	27.48	26.04	25.78
L_4	562	68	26.70	32.29	26.56	26.04	25.78
L_5	556	65	26.17	32.29	26.43	26.04	25.78
L_6	550	60	26.17	32.29	26.04	26.04	26.04
L_7	546	**57**	**25.78**	32.29	26.04	26.04	26.17
L_8	543	56	26.04	32.29	26.04	26.04	26.04
L_9	542	56	26.04	32.29	26.04	26.04	26.04
L_{10}	541	56	26.04	32.29	26.04	26.04	26.04

Abbildung 6.1: Generalisierungsfehler der fünf verwendeten Klassifikationsmethoden auf den Pima Indians Diabetes Daten. Der Trainingsprozess benötigt 10 Schichten, um das hierarchische Modell zu erzeugen.

Australian Credit Approval

Im nächsten Test wird der Australian Credit Approval Datensatz untersucht. Abbildung 6.2 verdeutlicht die Ergebnisse. Hier hat der Lernalgorithmus 4 Schichten benötigt, wobei im L_1–Modell schon eine Verbesserung der Generalisierungsfähigkeit von 4.4% zu erkennen ist und nur 88% der Beispieldaten verwendet werden.



Level	Anzahl		Klassifikationsfehler [in %]				
L_i	Daten	Regeln	L_i	$L_1[\rightarrow L_i]$	$L_1 \rightarrow L_i$	$L_4[\rightarrow L_i]$	$L_4 \rightarrow L_i$
FRL	621	125	18.84				
L_1	548	**36**	**14.49**	18.84	18.84	15.36	15.51
L_2	544	32	15.07	18.84	16.38	15.36	15.51
L_3	543	32	15.51	18.84	16.38	15.36	15.36
L_4	542	32	15.36	18.84	16.52	15.36	15.36

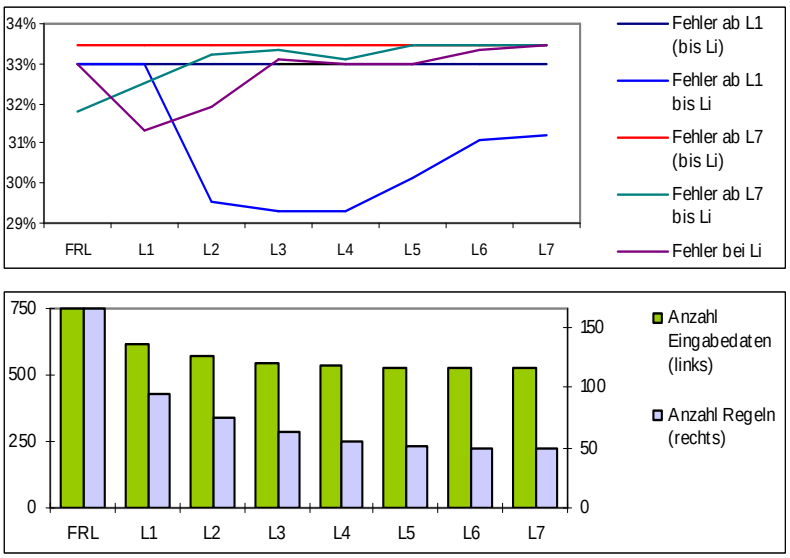
Abbildung 6.2: Generalisierungsfehler der fünf verwendeten Klassifikationsmethoden auf den Australian Credit Approval Datensatz. Der Trainingsprozess benötigt 4 Schichten, um das hierarchische Modell zu erzeugen.

Die Regelanzahl ist von 125 auf überschaubare 36 Regeln gefallen. In den $L_1[\rightarrow L_i]$ und $L_4[\rightarrow L_i]$-Klassifikationen gibt es keine Unterschiede in der Klassifikationsleistung der Einzelmodelle, die alle eine bessere Klassifikationsleistung im Vergleich zum FRL-Modell aufweisen. Dieser Datensatz zeigt, wie durch das Entfernen von Ausreißermustern zum einen eine deutlich kleinere Anzahl von Regeln erzeugt wird und zum anderen eine Verbesserung der Klassifikation erreicht werden kann. D. h., es wurden tatsächlich Muster entfernt, die zum Beispiel Ausreißer oder Artefakte in den Daten repräsentieren.

Vehicle Silhouettes Daten

Die Auswertung zum Vehicle Silhouettes Datensatz in Abbildung 6.3 zeigt wieder eine Verbesserung des Modells sowohl bezüglich der Generalisierungsfähigkeit als auch in der Anzahl der Regeln. Man sieht deutlich, dass sich mit geringerer Anzahl an Eingabedaten eine ebenso gute Klassifikationsleistung erreichen lässt, wobei weniger als ein Drittel der Regeln im Vergleich zum FRL-Modell benötigt werden.

Eine Verbesserung wird schon im Modell L_1 erreicht. Dabei fällt die Anzahl der Regeln um rund 44%. Eine bessere Leistung wird durch die Klassifikation der Modelle



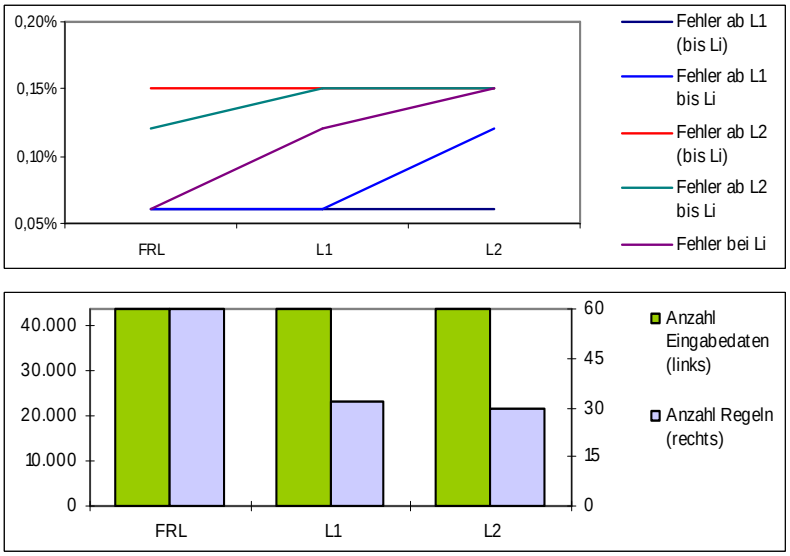
Level	Anzahl		Klassifikationsfehler [in %]				
L_i	Daten	Regeln	L_i	$L_1[\rightarrow L_i]$	$L_1 \rightarrow L_i$	$L_7[\rightarrow L_i]$	$L_7 \rightarrow L_i$
FRL	752	165	32.98				
L_1	621	**93**	**31.32**	32.98	32.98	33.45	32.51
L_2	575	75	31.91	32.98	29.55	33.45	33.21
L_3	550	62	33.10	32.98	29.31	33.45	33.33
L_4	538	55	32.98	32.98	29.32	33.45	33.10
L_5	531	51	32.98	32.98	30.14	33.45	33.45
L_6	528	50	33.33	32.98	31.09	33.45	33.45
L_7	527	50	33.45	32.98	31.20	33.45	33.45

Abbildung 6.3: Generalisierungsfehler der fünf verwendeten Klassifikationsmethoden auf den Vehicle Silhouettes Daten. Der Trainingsprozess benötigt 7 Schichten, um das hierarchische Modell zu erzeugen.

L_1 und L_2 erzielt. Man sieht erneut, wie sich die Größe der Regelbasis verringert und durch weniger Eingabedaten eine gleichwerte Klassifikationsleistung mit einem Drittel der Regeln erreichen lässt; was wiederum auf einen teilweise verrauschten Datensatz schließen lässt.

Shuttle Landing Control Daten

Abbildung 6.4 zeigt das 2-stufige hierarchische Modell, das auf Basis der Shuttle Landing Control Datenbank erzeugt wird. Man sieht, dass sich die Anzahl der Regeln um 50% verringert, wobei der Klassifikationsfehler geringfügig auf 0.12% (52 Muster) zunimmt. Alle anderen Klassifikationsmethoden und weitere Untersuchungen mit einem niedrigeren Schwellwert bringen keine Verbesserung der Modellleistung.

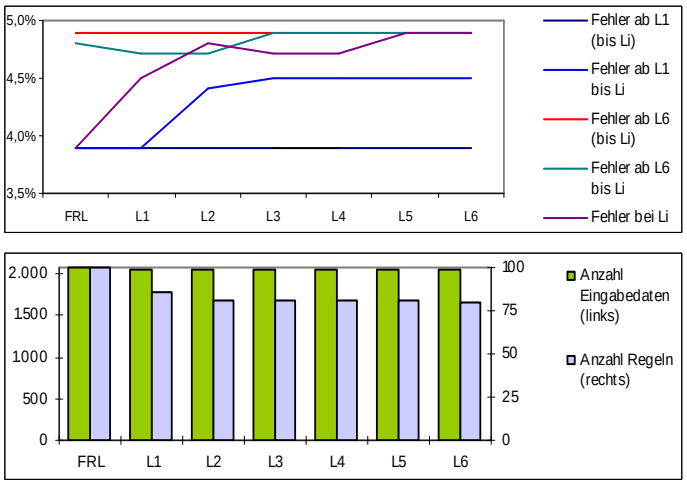


Level	Anzahl		Klassifikationsfehler [in %]				
L_i	Daten	Regeln	L_i	$L_1[\rightarrow L_i]$	$L_1 \rightarrow L_i$	$L_2[\rightarrow L_i]$	$L_2 \rightarrow L_i$
FRL	43500	60	**0.06**				
L_1	43451	32	0.12	0.06	0.06	0.15	0.15
L_2	43446	30	0.15	0.06	0.12	0.15	0.15

Abbildung 6.4: Generalisierungsfehler der fünf verwendeten Klassifikationsmethoden auf der Shuttle Landing Control Datenbank. Der Trainingsprozess benötigt 2 Schichten, um das hierarchische Modell zu erzeugen.

Image Segmentation Daten

Für die Image Segmentation Daten in Abbildung 6.5 konnte keine Verbesserung der Klassifikationsleistung erreicht werden. Das 6-stufige hierarchische Modell zeigt eine geringe Abnahme der Größe der Regelbasis (20%), wobei sich der Generalisierungsfehler um 0.99% (21 Muster) erhöht. Dieses Beispiel zeigt, dass sich die Anzahl der Regeln zwar verringert, was aber unter Umständen auf Kosten der Leistungsfähigkeit des Modells geht. D. h., es werden Muster entfernt, die für die Erklärung des zugrunde liegenden Konzepts notwendig sind, aber dennoch entfernt werden, weil sie von Detailregeln abgedeckt werden. Weitere Untersuchungen mit einem geringeren Grenzwert ergaben keine Verbesserung der Leistungsfähigkeit des Modells.

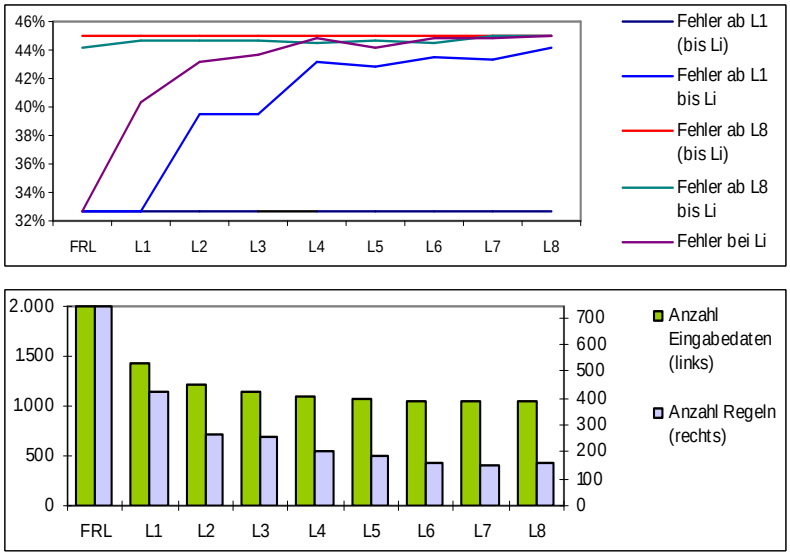


Level	Anzahl		Klassifikationsfehler [in %]				
L_i	Daten	Regeln	L_i	$L_1[\rightarrow L_i]$	$L_1 \rightarrow L_i$	$L_6[\rightarrow L_i]$	$L_6 \rightarrow L_i$
FRL	2079	96	**3.90**				
L_1	2060	82	4.50	3.90	3.90	4.89	4.72
L_2	2047	78	4.80	3.90	4.42	4.89	4.72
L_3	2045	78	4.72	3.90	4.50	4.89	4.89
L_4	2044	78	4.72	3.90	4.50	4.89	4.89
L_5	2043	78	4.89	3.90	4.50	4.89	4.89
L_6	2042	77	4.89	3.90	4.50	4.89	4.89

Abbildung 6.5: Generalisierungsfehler der fünf verwendeten Klassifikationsmethoden auf den Image Segmentation Daten. Der Trainingsprozess benötigt 6 Schichten, um das hierarchische Modell zu erzeugen.

DNA Sequence Daten

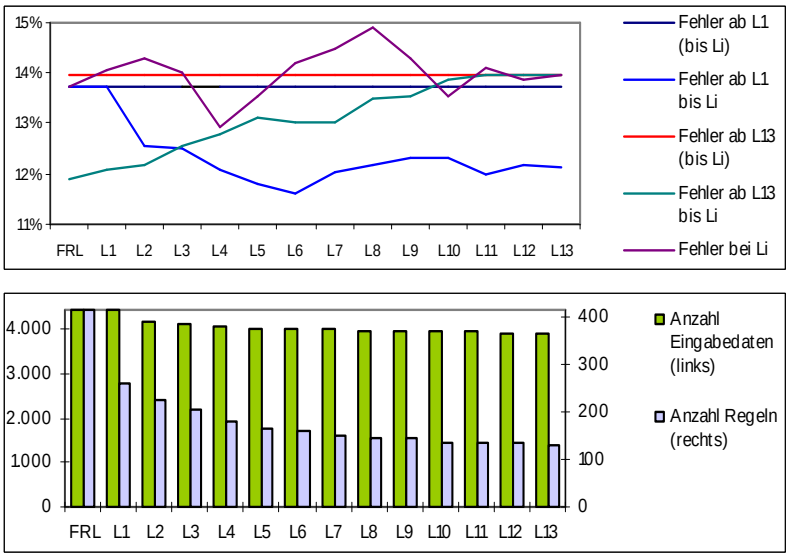
Die Auswertung des Tests auf dem DNA Sequence Datensatz in Abbildung 6.6 zeigt ebenfalls keine Verbesserung der Generalisierungsleistung in der Modellhierarchie im Vergleich zum FRL-Modell. Es werden Muster entfernt, die für die Erklärung des zugrunde liegenden Konzepts notwendig sind, aber dennoch nur durch kleine Detailregeln abgedeckt werden. Der Trainingsalgorithmus erzeugt bei diesem Beispiel ein 8-stufiges hierarchisches Modell. Man sieht, dass die Anzahl der Eingabemuster sich um fast 50% verringert, dagegen sich aber der Fehler um 12.22% (244 Muster)



Level	Anzahl		Klassifikationsfehler [in %]				
L_i	Daten	Regeln	L_i	$L_1[\rightarrow L_i]$	$L_1 \rightarrow L_i$	$L_8[\rightarrow L_i]$	$L_8 \rightarrow L_i$
FRL	2000	742	**32.72**				
L_1	1424	428	40.39	32.72	32.72	44.94	44.60
L_2	1212	262	43.17	32.72	39.54	44.94	44.60
L_3	1148	253	43.68	32.72	43.09	44.94	44.44
L_5	1070	183	44.18	32.72	42.83	44.94	44.69
L_6	1058	155	44.77	32.72	43.51	44.94	44.52
L_7	1057	146	44.77	32.72	43.34	44.94	44.94
L_8	1055	158	44.94	32.72	44.18	44.94	44.94

Abbildung 6.6: Generalisierungsfehler der fünf verwendeten Klassifikationsmethoden auf den DNA Sequence Daten. Der Trainingsprozess benötigt 8 Schichten, um das hierarchische Modell zu erzeugen.

stark erhöht. Es lässt sich dennoch eine starke Abnahme der Größe der Regelbasis verzeichnen, die hier auf fast 20% gegenüber dem FRL-Modell geschrumpft ist. Untersuchungen mit niedrigerem Grenzwert bringen keine weitere Verbesserung der Ergebnisse.



Level	Anzahl		Klassifikationsfehler [in %]				
L_i	Daten	Regeln	L_i	$L_1[\rightarrow L_i]$	$L_1 \rightarrow L_i$	$L_{13}[\rightarrow L_i]$	$L_{13} \rightarrow L_i$
FRL	4435	416	13.75				
L_1	4421	259	14.05	13.75	13.75	13.95	12.10
L_2	4158	224	14.30	13.75	12.55	13.95	12.20
L_3	4106	204	14.00	13.75	12.50	13.95	12.55
L_4	4051	**182**	**12.95**	13.75	12.10	13.95	12.80
L_5	4025	164	13.55	13.75	11.80	13.95	13.10
L_6	4009	162	14.20	13.75	11.60	13.95	13.00
L_7	3988	150	14.50	13.75	12.05	13.95	13.00
L_8	3976	146	14.90	13.75	12.20	13.95	13.50
L_9	3959	144	14.30	13.75	12.30	13.95	13.55
L_{10}	3945	137	13.55	13.75	12.30	13.95	13.85
L_{11}	3934	133	14.10	13.75	12.00	13.95	13.95
L_{12}	3924	133	13.85	13.75	12.20	13.95	13.95
L_{13}	3923	132	13.95	13.75	12.15	13.95	13.95

Abbildung 6.7: Generalisierungsfehler der fünf verwendeten Klassifikationsmethoden auf der Landsat-Satellite Datenbank. Der Trainingsprozess benötigt 13 Schichten, um das hierarchische Modell zu erzeugen.

Landsat-Satellite Daten

In diesem Test werden die Landsat-Satellite Daten auf die Generalisierungsfähigkeit untersucht. Wie in Abbildung 6.7 zu sehen, ergibt sich ein 13-stufiges Modell, bei dem sich die Anzahl der Daten zum letzten Level nur um 11.5% vermindert, dagegen aber die Anzahl der Regeln im Modell um 68% fällt und die Generalisierungsfähigkeit nur geringfügig abnimmt.

Die beste Einzelmodellleistung mit 12.95% lässt sich bei dem Modell L_4 erkennen. Geringfügig bessere Ergebnisse erreicht man bei der Klassifikation der Modelle L_1 bis L_6 (11.60%). Für die zweite und vierte Klassifikationsmethode gibt es wieder keine Unterschiede in den jeweiligen Schichten. Dieses Beispiel zeigt, wie durch einfaches Herausfiltern einiger weniger Muster eine wesentlich kleinere Regelbasis erzeugt wird und zusätzlich der Klassifikationsfehler verringert werden kann.

Letter Recognition Daten

Die Untersuchungen auf der Letter Recognition Datenbank in Abbildung 6.8 bringen keine Verbesserung in der Klassifikationsleistung. Es lässt sich aber erneut eine Abnahme der Größe der Regelbasis erkennen, wobei der Generalisierungsfehler um 7.2% ansteigt. Weitere Tests mit niedrigerem Grenzwert bringen keine Verbesserung der Leistungsfähigkeit des Modells.

Level	Anzahl		Klassifikationsfehler [in %]				
L_i	Daten	Regeln	L_i	$L_1[\rightarrow L_i]$	$L_1 \rightarrow L_i$	$L_4[\rightarrow L_i]$	$L_4 \rightarrow L_i$
FRL	15000	2369	**14.36**				
L_1	12760	1361	16.89	14.36	14.36	21.56	21.11
L_2	11821	1012	17.34	14.36	17.87	21.56	21.43
L_3	11377	835	19.83	14.36	18.41	21.56	21.56
L_4	11123	756	21.56	14.36	20.55	21.56	21.56

Abbildung 6.8: Generalisierungsfehler der fünf verwendeten Klassifikationsmethoden auf der Letter Recognition Datenbank. Der Trainingsprozess benötigt 4 Schichten, um das hierarchische Modell zu erzeugen.

6.1 Auswertung: hierarchische Fuzzy-Modelle

Tabelle 6.1 fasst die Ergebnisse im Vergleich zum flachen Fuzzy-Regellerner (FRL) zusammen. Die Tabelle zeigt den Generalisierungsfehler auf den verwendeten Datensätzen. Zusätzlich ist in Klammern die Zahl der Regeln angegeben. Man sieht für vier der acht Datensätze eine verbesserte Klassifikationsleistung. Weiterhin werden Regelbasen erzeugt, die im Vergleich zum FRL-Modell zum Teil sehr viel kleiner sind. Um das hierarchische Modell zu erzeugen, werden zwischen 2 und 13 Schichten generiert. Durch die schichtenartige Struktur des Fuzzy-Modells ist man in der Lage, in die Daten zu „zoomen“, um die gewünschte Information mit der erforderlichen Granularität zu extrahieren. Man sieht, dass sich kleinere Regelbasen erzeugen lassen, die ein ebenso gutes oder teilweise sogar besseres Generalisierungsverhalten aufweisen. Hierbei spielt die Größe der Regelbasis eine wichtige Rolle, da sich durch eine verringerte Anzahl der Regeln sowohl die Verständlichkeit als auch die praktische Anwendbarkeit des Modells erhöht. Unter Umständen ist es in bestimmten Anwendungen sinnvoll, geringe Einbußen in der Leistung des Modells hinzunehmen. Auf der anderen Seite zeigt sich dadurch eine bessere Interpretierbarkeit des Modells. Die hierarchischen Fuzzy-Regelmodelle gehen einen Schritt weiter. Diese vereinen gerade diesen Kompromiss, da man auf der einen Seite ein spezielles Modell hat, das

Tabelle 6.1: Zusammenfassung der Klassifikationsfehler (in %) und Anzahl der Regeln der getesteten Datensätze des StatLog-Projekt im Vergleich zum FRL (siehe erste und zweite Spalte). Die anschließenden Spalten zeigen die Ergebnisse des hFRL bei Verwendung verschiedener Klassifikationsmethoden. Die besten Ergebnisse werden jeweils hervorgehoben. In Klammern wird die Anzahl der Regeln notiert.

Datensatz	FRL		hFRL					
			L_i		$L_1[\rightarrow L_i]$	$L_1 \rightarrow L_i$	$L_n[\rightarrow L_i]$	$L_n \rightarrow L_i$
Diabetes	32.29	(151)	25.78	(57)	32.29	26.04	26.04	**25.52**
Aust. Cr.	18.84	(125)	**14.49**	(36)	18.84	16.38	15.36	15.36
Vehicle	32.98	(165)	31.32	(93)	32.98	**29.31**	33.45	32.51
Segment	**3.90**	(96)	4.50	(82)	**3.90**	**3.90**	4.89	4.72
Shuttle	**0.06**	(60)	0.06	(32)	**0.06**	**0.06**	0.15	0.15
SatImage	13.75	(416)	12.95	(182)	13.75	**11.60**	13.95	12.10
DNA	**32.72**	(742)	40.39	(428)	**32.72**	**32.72**	44.94	44.44
Letter	**14.36**	(2369)	**14.36**	(2369)	**14.36**	**14.36**	21.56	20.96

eine gute Klassifikationsleistung besitzt. Auf der anderen Seite ist es möglich, tiefer in die Hierarchie abzusteigen, um kleinere und damit verständlichere Regelmodelle betrachten zu können.

6.2 Ergebnisse auf dem NCI–HIV Datensatz

Nachdem die Experimente des hierarchischen Fuzzy-Regellerners auf den acht Benchmark-Datensätzen erfolgreich durchgeführt wurden, werden sich unsere weiteren Untersuchungen auf einen Datenbestand auf dem Bereich der Bioinformatik stützen. Es gilt zu zeigen, dass sich der hierarchische Regellernansatz praktisch verwenden lässt. Hierbei spielt die Anzahl der Regeln innerhalb der Hierarchie eine wichtige Rolle und sollen während der Experimente genauer betrachtet werden.

Die Daten stammen aus einer Anwendung, bei der der Schutz menschlicher CEM Zellen vor HIV-1 Infektion gemessen wird; eine detaillierte Beschreibung findet sich in Weislow u. a. (1989). Die Daten lassen sich wie folgt charakterisieren: Die Substanzen, die mindestens einen 50%-igen Schutz der CEM Zellen zeigen, werden wiederholt getestet. Die Stoffe, welche im zweiten Test ebenfalls eine mindestens 50%-ige Aktivität zeigen, werden als mittelgradig aktiv (**CM**) bezeichnet. Die Substanzen, bei denen wiederholt ein 100%-iger Schutz festgestellt wurde, werden als bestätigt aktiv (**CA**) markiert. Alle Komponenten, die nicht diese Kriterien erfüllen, werden als bestätigt inaktiv (**CI**) klassifiziert. Der Datensatz mit 41 316 Instanzen besteht aus 56-dimensionale so genannte VolSurf-Deskriptoren, von denen 36 082 für unsere Untersuchungen verwendet werden [1], wobei 325 Muster zu Klasse **CA**, 877 zu **CM** und die restlichen 34 880 zu **CI** gehören. Weiterhin lassen sich 69 Muster chemischen Klassen zuordnen, wobei 13 Beispiele zur Klasse der *Dyes und Polyanions* angehören, die im Folgenden näher betrachtet werden sollen.

Für die Tests wird wiederum die Minimum/Maximum-Norm verwendet. Als Heuristik für die Konfliktlösung wird die bereichbasierte angewendet, die den Volumenverlust bezüglich des Kern- bzw. Einflussbereiches normalisiert. Zusätzlich wird ein Grenzwert von 5 gewählt, um Regeln mit niedriger Relevanz aus dem Modell zu extrahieren. Der Algorithmus erzeugt 6 Hierarchieschichten während des Trainings.

Die Tabelle 6.2 zeigt die Ergebnisse auf den NCI-HIV Daten. Jede Zeile enthält

[1] Aufgrund unbrauchbarer Struktureigenschaften ist es nicht möglich, für alle Moleküle einen Deskriptor zu erzeugen.

Tabelle 6.2: Ergebnisse auf dem NCI–HIV-1 Datensatz. Es wird ein 6-schichtiges hierarchisches Fuzzy-Modell erzeugt. Die einzelnen Schichten sind in der ersten Spalte aufgetragen. Die nachfolgenden Spalten zeigen die Anzahl der verwendeten Eingabedaten und zusätzlich die Anzahl erzeugter Regeln in Klammern, zum einen nach allen und zum anderen nach den einzelnen Klassen gruppiert.

Level	**CA+CM+CI**		**CA**		**CM**		**CI**	
FRL	36082	(2492)	324	(195)	877	(654)	34881	(1643)
L_1	36082	(548)	324	(8)	877	(4)	34881	(536)
L_2	33244	(69)	70	(5)	26	(3)	33148	(61)
L_3	33146	(52)	49	(5)	20	(2)	33077	(45)
L_4	33096	(40)	42	(4)	14	(2)	33040	(34)
L_5	33071	(39)	38	(4)	13	(2)	33020	(33)
L_6	33060	(44)	38	(5)	13	(3)	33009	(36)

die Anzahl der erzeugten Regeln und verwendeten Muster in Klammern; zum einen nach allen Klassen (CA+CM+CI) und zum anderen getrennt nach Klassen. Dieses Experiment zeigt deutlich, wie mit zunehmender Tiefe die Größe der Regelbasis abnimmt. Im FRL-Modell sind 2 492 Regeln notwendig, um 36 082 Daten zu erklären (1.7 Muster pro Regel Klasse CA, 1.3 Muster pro Regel Klasse CM, 21.2 Muster pro

R_1 IF
G IS $\langle -\infty, 0.0, 0.353, 0.370 \rangle$ AND
W8 IS $\langle 0.107, 0.110, 1.0, +\infty \rangle$ AND
Iw2 IS $\langle 0.013, 0.055, 1.0, +\infty \rangle$ AND
Emin3 IS $\langle -\infty, 0.0, 0.691, 0.810 \rangle$ AND
D13 IS $\langle 0.003, 0.009, 1.0, +\infty \rangle$ AND
D23 IS $\langle 0.008, 0.018, 1.0, +\infty \rangle$ AND
D1 IS $\langle 0.075, 0.090, 1.0, +\infty \rangle$ AND
D8 IS $\langle 0.030, 0.040, 1.0, +\infty \rangle$ AND
ID4 IS $\langle 0.038, 0.041, 1.0, +\infty \rangle$ AND
A IS $\langle -\infty, 0.0, 0.473, 0.484 \rangle$ AND
POL IS $\langle 0.0, 0.418, 1.0, +\infty \rangle$ THEN **CA**

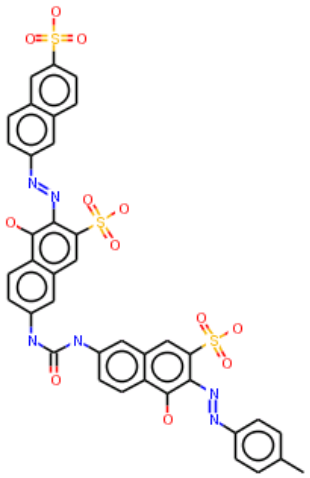


Abbildung 6.9: Linke Seite zeigt eine Regel der Klasse **CA** aus der untersten Hierarchieschicht (Modell L_6). Diese Regel ist auf 11 der 56 Merkmale eingeschränkt und erklärt 15 Eingabebeispiele. Die rechte Seite zeigt zusätzlich ein Beispielmolekül der chemischen Gruppe der *Dyes und Polyanions*.

Regel Klasse CI). Im Level L_1 werden nach Filtern des Modells dagegen nur noch 548 Regeln benötigt. Weiter unten in der Hierarchie werden sukzessive Ausreißerregeln extrahiert, die zu einer Verkleinerung der Regelbasis führen. Hier werden zum Teil chemische Gruppen extrahiert, die nur seltene aber dennoch interessante Beispiele in den Daten beschreiben. Durch das Entfernen zeigen sich in den tieferen Ebenen Details über die Struktur des Systems. Der Experte ist somit in der Lage, das Regelsystem zu interpretieren und einzelne Regeln auszuwerten. Für die weiteren Untersuchungen werden zwei Regeln der Klasse der aktiven Komponenten (**CA**) aus der untersten Hierarchieschicht ausgewählt und näher betrachtet.

Abbildung 6.9 zeigt eine Regel, die 15 Eingabebeispiele erklärt und auf 11 der 56 Attribute eingeschränkt ist; Attribute `G`, `Emin3` und `A` sind nach links, alle anderen sind nach rechts unbeschränkt. Von den 15 erklärten Mustern lassen sich 5 der chemischen Gruppe der *Dyes und Polyanions* zuordnen; für die restlichen Moleküle ist keine chemische Gruppe bekannt. Abbildung 6.10 zeigt eine Regel, die 11 Datenbeispiele erklärt und auf 13 der 56 Attribute eingeschränkt ist; Attribut `D13` ist nach links, alle anderen sind nach rechts unbeschränkt. Von den 13 erklärten Mustern lassen sich ebenfalls fünf (aber andere) der chemischen Gruppe der *Dyes und Polyanions* zuordnen; für die restlichen Moleküle ist keine Gruppe bekannt.

R_2 IF
W8 IS $\langle$ $0.127, 0.134, 1.0, +\infty$ $\rangle$ AND
Iw2 IS $\langle$ $0.004, 0.054, 1.0, +\infty$ $\rangle$ AND
Iw6 IS $\langle$ $0.027, 0.048, 1.0, +\infty$ $\rangle$ AND
Cw8 IS $\langle$ $0.203, 0.257, 1.0, +\infty$ $\rangle$ AND
D12 IS $\langle$ $0.030, 0.053, 1.0, +\infty$ $\rangle$ AND
D13 IS $\langle$ $-\infty, 0.0, 0.207, 0.401$ $\rangle$ AND
D23 IS $\langle$ $0.0, 0.059, 1.0, +\infty$ $\rangle$ AND
ID3 IS $\langle$ $0.015, 0.094, 1.0, +\infty$ $\rangle$ AND
ID4 IS $\langle$ $0.059, 0.099, 1.0, +\infty$ $\rangle$ AND
ID6 IS $\langle$ $0.056, 0.072, 1.0, +\infty$ $\rangle$ AND
ID7 IS $\langle$ $0.003, 0.063, 1.0, +\infty$ $\rangle$ AND
ID8 IS $\langle$ $0.026, 0.058, 1.0, +\infty$ $\rangle$ AND
CP IS $\langle$ $0.001, 0.002, 1.0, +\infty$ $\rangle$ THEN **CA**

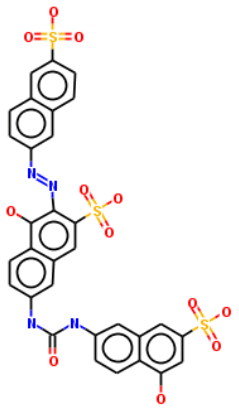


Abbildung 6.10: Die linke Seite zeigt eine Regel der Klasse **CA** aus der untersten Hierarchieschicht (Modell L_6). Diese Regel ist auf 13 der 56 Merkmale eingeschränkt und erklärt 11 Eingabebeispiele. Die rechte Seite zeigt zusätzlich ein Beispielmolekül der chemischen Gruppe der *Dyes und Polyanions*.

Dieses Beispiel aus dem Bereich der Bioinformatik verdeutlicht die praktische Anwendbarkeit des hierarchischen Lernansatzes, um verständlichere Modelle mit einfachen und interpretierbaren Regeln zu erzeugen. Ein normales Fuzzy-Modell ist hier praktisch nicht verwendbar, da zu viele Regeln in diesem flachen System erzeugt werden. Durch die induzierte, hierarchische Struktur des Regelmodells ist es nun möglich, interessante Bereiche in der Regelhierarchie zu untersuchen. In einer Hierarchie von Regelbasen kann ein Anwender sich auf der oberen Schicht einen Überblick über seine Daten verschaffen und später interessante Bereiche tiefer in der Hierarchie erforschen. Diese beschreiben die Daten detailliert und geben dem Anwender einen detaillierten Einblick in seine Datenbasis.

6.3 Zusammenfassung und Ergebnisse

Die Experimente auf den acht Benchmark-Datensätzen sowie dem NCI-HIV Datensatz zeigen gute Ergebnisse des hierarchischen Ansatzes, der – auf Basis des in Kapitel 2 vorgestellten Fuzzy-Regellerners – hier evaluiert wurde.

Die erzeugten hierarchischen Fuzzy-Regelsysteme (hFRL) zeigen teilweise eine signifikante Verringerung der Regelanzahl in den Schichten der Hierarchie und gleichzeitig eine bessere Klassifikation gegenüber dem flachen Fuzzy-Regelmodell (FRL). Gerade diese Eigenschaften sind wichtig, wenn Regelmodelle später interpretiert werden sollen; und nicht ausschließlich für die Klassifikation vorgesehen sind. Bessere Klassifikationsergebnisse mit weniger Regeln können ein Indiz für Ausreißer oder Fehler in den Daten sein, die für die Erklärung des zugrunde liegenden Konzeptes nicht notwendig sind. Sind solche Muster in den Daten interessant, kann die Regelhierarchie verwendet werden, um in den unteren Ebenen diese Besonderheiten zu explorieren.

Die graduelle Struktur der Regelmodellhierarchie kann einem Anwender helfen, ein besseres Verständnis für sein Modell und den zugrunde liegenden Daten zu erhalten. Oftmals werden viele Detailregeln generiert, die aber in der Praxis nicht benötigt werden. Eine Experte verschafft sich gewöhnlich erst einen groben Überblick, bevor interessante (Detail-)Bereiche im Modell genauer exploriert werden. Genau hier setzt unserer hierarchischer Ansatz an, der alle Details mit nur wenigen Regeln pro Ebene in Hierarchie beschreibt, aber gleichzeitig allgemein verständlich auf den oberen Ebenen ist.

Kapitel 7

Zusammenfassung

In dieser Forschungsarbeit wurden Verfahren diskutiert, die automatisch Regeln aus Daten extrahieren. Diese Ansätze leiden häufig darunter, dass sie viele Detailregeln auf großen Datenbeständen generieren und damit Vorhersagemodelle erzeugen, die nur schwer interpretiert werden können. Fuzzy-Regeln können helfen, diesem Problem zu begegnen, da sie Bereiche zwischen Regeln durch ihre Unschärfe besser modellieren können. Dennoch wurde gezeigt, dass auch Fuzzy-Ansätze unter Umständen viele Regeln erzeugen, um komplexe Zusammenhänge in großen, komplizierten Datensätzen zu beschreiben. Aus dieser Motivation heraus wurde eine hierarchische Erweiterung vorgestellt, die es ermöglicht, schichtenbasierte Regelmodelle zu generieren. Diese Hierarchien bestehen aus Regelmodellen auf den unterschiedlichen Niveaus, die das Konzept graduell, d. h., möglichst generell aber gleichzeitig speziell genug approximieren. Dabei beschreiben in den oberen Schichten nur wenige Regeln grob das Konzept hinter den Daten. Weiter unten in der Hierarchie konzentrieren sich Detailregeln auf Artefakte und Ausreißer in den Daten. Sie repräsentieren aber auch alle Details und Feinheiten des Gesamtmodells, die das darunter liegende Konzept zusammen mit Regeln auf den oberen Ebenen beschreiben.

Zur Erzeugung von Fuzzy-Regelsystemen wurde ein Lernverfahren in Kapitel 2 erweitert, welches basierend auf dem Dynamic Decay Adjustment Algorithmus ein konstruktives Training rechteckiger Fuzzy-Regeln durchführt. Diese Methode wurde implementiert und auf acht ausgewählten Benchmark-Datensätzen des StatLog-Projekts in Kapitel 3 angewendet. Die Untersuchungen wurden mit fünf verschiedenen Fuzzy-Normen realisiert – Minimum/Maximum-Norm, Produkt-Norm, $\text{Yager}_{1/2}$-Norm, Łukasiewicz-Norm und die Yager_2-Norm. Diese Normen zeigen unterschied-

liche Ergebnisse in der Generalisierungsleistung des Regelmodells, wobei die nichtparametrisierten Normen – Minimum/Maximum und Produkt – gute Ergebnisse liefern und stabiler in der Generalisierungsfähigkeit gegenüber den anderen Normen sind. Von den parametrisierten Normen liegen $\text{Yager}_{1/2}$-Norm und Łukasiewicz-Norm unterhalb des Mittelwertes; die Yager_2-Norm erzielt ebenfalls gute Ergebnisse in der Klassifikationsleistung. Des Weiteren wurde der Fuzzy-Regellerner, der Prototypen wenn notwendig in ein Modell einfügt oder während des Trainings anpasst, auf verschiedenen Heuristiken zur Konfliktlösung untersucht. Gute Ergebnisse liefert hier die bereichsbasierte Strategie. Die regel- und kernbasierten Konfliktlösungsstrategien weichen zum Teil stark vom durchschnittlichen Klassifikationsfehler ab.

In Kapitel 4 wurden hierarchische Lernansätze angeführt, die ähnlich zu den betrachteten hierarchischen Regellernverfahren, schichtbasierte Modelle erzeugen. Dazu gehören zum einen Entscheidungsbaum-Verfahren, die den Eingaberaum der Attribute rekursiv partitionieren. Jeder Schnitt oder Knoten im Baum verfeinert das Modell auf Basis der Menge der Merkmale. Das erzeugte Modell ist eine verzweigte Liste von Regeln, die zu Beginn den Raum sehr grob aufteilen und später immer feiner verzweigen. Ein zweiter, eher untypischer Ansatz wird in hierarchischen Clustering-Verfahren angewandt. Diese generieren eine Baumstruktur, in der ähnliche Datenpunkte zusammengefasst werden. Durch die erzeugte Struktur werden die Daten in einem hierarchischen Clustermodell ebenenweise zusammengefasst, sodass eine beliebige Anzahl von Clustern aus der Hierarchie heraus extrahiert werden kann.

In Kapitel 5 wurde eine Erweiterung des Fuzzy-Regellerners vorgestellt, der hierarchische Fuzzy-Regelmodelle erzeugt. Die Idee basiert auf einem lokalen Ausreißermodellansatz, der aus einem trainierten Fuzzy-Modell ein Detailmodell extrahiert. Dieses Modell wird anschließend als Filter verwendet, um ein generelleres Modell zu trainieren. Führt man dieses Verfahren weiter fort, erhält man eine mehrschichtige Fuzzy-Modellhierarchie. Die Regeln, die unterhalb eines zuvor definierten Filterschwellwerts liegen, gehen in das Detailmodell ein. Die entstandene Hierarchie von Regelmodellen enthält in den oberen Schichten nur wenige Regeln, die das Konzept grob beschreiben; weiter unten befinden sich Detailregeln, die notwendig sind, um alle Feinheiten der Daten zu modellieren.

Datensätze können verrauschte Einträge enthalten, die durch Fehlmessungen oder fehlerhafte Aufzeichnungen entstanden sein können. Auf solchen Datensätzen liefert das hierarchische Fuzzy-Modell bessere Klassifikationsergebnisse als klassische Re-

gellernverfahren, da diese tiefer liegenden Bereiche in der Hierarchie eine niedrigere Relevanz haben. Dagegen werden, abhängig von der Klassifikationsstrategie, relevante Regeln weiter oben in der Hierarchie bevorzugt. Die weiteren Untersuchungen auf den Benchmark-Datensätzen zeigen, dass die Größe der Regelbasis teilweise stark abnimmt. Dadurch bieten die Modelle in den tiefer liegenden Schichten eine genauere Interpretierbarkeit der Daten, die Modelle bleiben aber dennoch generell auf den oberen Schichten der Hierarchie.

Es wurden zwei Filteransätze betrachtet, die auf Basis eines lokalen Regellernverfahrens hierarchisch-strukturierte Modelle erzeugen. Diese Modelle sind explorierbar und interpretierbar zugleich. Die hierarchischen Regelmodelle generalisieren auf oberen Schichten und bleiben damit auch im Falle komplexer Konzepte verständlich, beschreiben aber auch alle Details der Daten auf den unteren Schichten. Das Gesamtmodell eröffnet damit Raum für die Exploration auf den verschiedenen Abstraktionsebenen der Hierarchie. Die Hierarchie besteht oftmals aus nur wenigen Regeln auf den obersten Ebenen und ist daher ein geeigneter Startpunkt für die Exploration solcher Regelmodellhierarchien. Durch die graduelle Struktur des hierarchischen Regelsystems erhöht sich die Verständlichkeit, weil nicht mehr alle Details in einem komplexen, statischen Modell erklärt werden müssen, sondern die Komplexität des Gesamtmodells über mehrere kleinere Modelle verteilt wird.

Die Evaluierung des vorgestellten hierarchischen Ansatzes in Kapitel 6 bezüglich Explorations- und Klassifikationsfähigkeit auf einigen Benchmark-Datensätzen zeigt zum einen eine Verbesserung der Vorhersagefähigkeit mit nur wenigen Regeln der Hierarchie; zum anderen ergibt sich eine bessere Explorierbarkeit des Regelmodells. Um die praktische Anwendbarkeit zu demonstrieren, wurde der hierarchische Ansatz in Kapitel 6.2 ebenfalls auf einem bekannten Datensatz aus dem Bereich der Bioinformatik getestet. In diesem Versuch liefert der hierarchische Fuzzy-Regellerner gute Ergebnisse. Aufgrund der verringerten Größe der Regelbasis in den unteren Modellen ist es möglich, das Regelmodell auszuwerten. Der Experte hat nun die Aufgabe, sich die erklärten Regeln anzusehen und selbstständig (neue) interessante Zusammenhänge zwischen Regeln und Daten zu erkennen.

Mit der Verwendung von hierarchischen Modellen erzielt man ein besseres Verständnis des zugrunde liegenden Konzepts der Daten. Diese Modelle enthalten weniger Regeln auf den verschiedenen Ebenen und können dadurch besser analysiert und dadurch von einem Experten verstanden werden. Für die praktische Anwendbarkeit

ist es entscheidend, dass sich diese hierarchischen Modelle interaktiv explorieren lassen. Durch die schichtenartige Struktur der Systeme lässt sich jedes Modell getrennt visualisieren, weil die Beziehungen zwischen den Systemen ebenenübergreifend modelliert werden. Für den Anwender ist es wichtig, sein gelerntes Modell zu verstehen und Rückschluss auf die Daten selbstständig durchführen zu können.

Visualisierungsmethoden sind in diesem Anwendungsgebiet interessant, weil diese sich teilweise für Regelmodelle und Regelhierarchien erweitern lassen. Erste Ansätze der Visualisierung solcher Systeme zeigen interessante und viel versprechende Einblicke in Modelle und Daten. Multi-Dimensionale Skalierungsmethoden (Gabriel u. a., 2009) und Parallele Koordinaten (Gabriel u. a., 2005) wurden erweitert, damit hierarchische Fuzzy-Regelsysteme geeignet visualisiert werden können. Durch die hierarchische Struktur ist es nun möglich, das Modell interaktiv zu explorieren, da Regeln innerhalb der Hierarchie verknüpft sind. Damit lassen sich Regeln durch die Hierarchie bis zum Ursprung ihrer Daten verfolgen, um neues, unbekanntes Wissen zu entdecken. Durch die Granulierung der Informationen (Zadeh, 1998) lassen sich hierarchische Modelle zusammen mit den zugehörigen Daten besser verstehen und stellen damit einen interessanten Ansatz für die intelligente, explorative Datenanalyse dar.

Literaturverzeichnis

[Amari 1995] Amari, S.: Statistical and Information-Geometrical Aspects of Neural Learning. In: *Computational Intelligence*, IEEE Press, 1995 (A Dynamic System Perspective), S. 71–82

[Asuncion und Newman 2007] Asuncion, A. ; Newman, D.J.: *UCI Machine Learning Repository*. 2007. – URL `http://mlearn.ics.uci.edu/MLRepository`. – [Online; accessed 2008-06-19]

[Berthold 1997] Berthold, M.R.: *Konstruktives Training Probabilistischer Neuronaler Netze für die Musterklassifikation*. DISKI 155, infix Verlag, 1997

[Berthold 2000] Berthold, M.R.: Learning Fuzzy Models and Potential Outliers. In: *Computational Intelligence in Data Mining*, Springer-Verlag, 2000, S. 111–126

[Berthold 2003] Berthold, M.R.: Mixed Fuzzy Rule Formation. In: *International Journal of Approximate Reasoning (IJAR)* 32 (2003), S. 67–84

[Berthold und Diamond 1998] Berthold, M.R. ; Diamond, J.: Constructive Training of Probabilistic Neural Networks. In: *Neurocomputing* 19 (1998), S. 167–183

[Berthold und Huber 1997] Berthold, M.R. ; Huber, K.-P.: Tolerating Missing Values In A Fuzzy Environment. In: *IFSA World Congress* 1 (1997), S. 359–361

[Bezdek u. a. 1998] Bezdek, J.C. ; Keller, J. ; Krishnapuram, R. ; Pal, N.: *Fuzzy*

Models and Algorithms for Pattern Recognition and Image Processing. Kluwer, Norwell, MA, USA, 1998

[Breiman u. a. 1984] Breiman, L. ; Friedman, J.H. ; Olshen, R.A. ; Stone, C.J.: *Classification and Regression Trees*. Belmont, California, U.S.A. : Wadsworth Publishing Company, 1984 (Statistics/Probability Series)

[Dejean u. a. 2002] Dejean, H. ; Hammerton, J. ; Osborne, M. ; Armstrong, S. ; Daelemans, W.: Learning rules and their exceptions. In: *Journal of Machine Learning Research* 2 (2002), S. 669–693

[Dubois u. a. 1996] Dubois, D. ; Prade, H. ; Yager, R.R.: Information Engineering and Fuzzy-Logic. In: *Proc. 5th IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE'96 New Orleans,LA, USA)* 3 (1996), S. 1525–1531

[Fayyad u. a. 1996] Fayyad, U.M. ; Piatetsky-Shapiro, G. ; Smyth, P. ; Uthurusamy, R. ; eds.: *Advances in Knowledge Discovery and Data Mining*. MIT Press, Cambridge, MA, USA, 1996

[Fukunaga 1990] Fukunaga, K.: *Introduction to Statistical Pattern Recognition*. 2. San Diego, CA : Academic Press, 1990

[Furnkranz 1997] Furnkranz, J.: Pruning Algorithms for Rule Learning. In: *Machine Learning*, 1997, S. 139–171

[Gabriel und Berthold 2003] Gabriel, Thomas R. ; Berthold, M.R.: Constructing Hierarchical Rule Systems. In: Berthold, M.R. (Hrsg.) ; Lenz, H.-J. (Hrsg.) ; Bradley, E. (Hrsg.) ; Kruse, R. (Hrsg.) ; Borgelt, C. (Hrsg.): *Proc. 5th International Symposium on Intelligent Data Analysis (IDA 2003)*, Springer Verlag, 2003 (Lecture Notes in Computer Science (LNCS)), S. 76–87

[Gabriel und Berthold 2005] Gabriel, Thomas R. ; Berthold, M.R.: Missing Values in Fuzzy Rule Induction. In: *IEEE Conference on Systems, Man and Cybernetics*, IEEE Press, 2005, S. 1473–1476

[Gabriel u. a. 2005] Gabriel, Thomas R. ; Pintilie, A.S. ; Berthold, M.R.: Exploring Hierarchical Rule Systems in Parallel Coordinates. In: *Proc. 6th International Symposium on Intelligent Data Analysis (IDA 2005)*, Springer Verlag, 2005 (Lecture Notes in Computer Science (LNCS) 3646), S. 97–108

[Gabriel u. a. 2009] Gabriel, Thomas R. ; Thiel, K. ; Berthold, M.R.: Multi-Dimensional Scaling applied to Hierarchical Rule Systems. In: *International Fuzzy Society Association World Congress*, 2009, S. 944–949

[Gentsch 2001] Gentsch, P.: *Data Mining-Tools: Eine neue Ära für das Data Mining?* `http://www.intelligence-group.com/downloads/DataMining2001.pdf`. 2001. – [Online; accessed 2009-02-10]

[Gras und Kuntz 2006] Gras, R. ; Kuntz, P.: Discovering *R*-rules with a directed hierarchy. In: *Soft Computing* 10 (2006), Nr. 5, S. 453–460

[Higgins und Goodman 1993] Higgins, C.M. ; Goodman, R.M.: Learning Fuzzy Rule-Based Neural Networks for Control. In: *Advances in Neural Information Processing Systems*. California : Morgan Kaufmann, 1993 (5), S. 350–357

[Holden und Anthony 1993] Holden, S.B. ; Anthony, M.: Quantifying Generalization in Linearly Weighted Neural Networks. In: *Complex Systems* 8 (1993), S. 8–91

[Höppner u. a. 1999] Höppner, F. ; Klawonn, F. ; Kruse, R. ; Runkler, T.: *Fuzzy Cluster Analysis*. Chichester, England : J. Wiley & Sons, 1999

[Huber und Berthold 1995] Huber, K.-P. ; Berthold, M.R.: Data Analysis with Rule Generating Learning Algorithms. In: Lasker, George E. (Hrsg.) ; Liu, Xiaohui (Hrsg.): *Advances in Intelligent Data Analysis* Bd. 1, IIAS, 1995, S. 80–84

[Ishibuchi u. a. 1992] Ishibuchi, H. ; Nozaki, K. ; Tanaka, H.: Distributed representation of fuzzy rules and its application to pattern classification. In: *Fuzzy Sets Systems* 52 (1992), Nov, S. 21–32

[Jain und Dubes 1988] Jain, Anil K. ; Dubes, Richard C.: *Algorithms for Clustering Data*. Prentice-Hall, 1988

[Kivinen u. a. 1992] Kivinen, J. ; Mannila, H. ; Ukkonen, E.: Learning hierarchical rule sets. In: *COLT '92: Proceedings of the fifth annual workshop on Computational learning theory*. New York, NY, USA : ACM New York, NY, USA, 1992, S. 37–44

[Kivinen u. a. 1994] Kivinen, J. ; Mannilaand, H. ; Ukkonen, E.: Learning rules with local exceptions. In: *Euro-COLT '93: Proceedings of the first European conference on Computational learning theory*. New York, NY, USA : Oxford University Press, Inc., 1994, S. 35–46

[Klir und Folger 1987] Klir, G.J. ; Folger, T.A.: *Fuzzy sets, uncertainty, and information*. Upper Saddle River, NJ, USA : Prentice-Hall, Inc., 1987

[Klir und Yuan 1996] Klir, G.J. (Hrsg.) ; Yuan, B. (Hrsg.): *Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems : selected papers by Lotfi A. Zadeh*. World Scientific, 1996

[Krogh und Vedelsby 1995] Krogh, A. ; Vedelsby, J.: Neural Network Ensembles, Cross Validation, and Active Learning. In: Tesauro, G. (Hrsg.) ; Touretzky, D. S. (Hrsg.) ; Leen, T. K. (Hrsg.): *Advances in Neural Information Processing Systems*. Cambridge MA : MIT Press, 1995 (7), S. 231–238

[Kruse u. a. 1999] Kruse, R. ; Borgelt, C. ; Nauck, D.: Fuzzy Data Analysis: Challenges and Perspectives. In: *Proc. 8th IEEE Int. Conf. on Fuzzy Systems (FUZZ-IEEE'99 Seoul,Korea)*. Piscataway, NJ, USA : IEEE Press, 1999, S. 1211–1216

[Kruse u. a. 1993] Kruse, R. ; Gebhardt, J. ; Klawonn, F.: *Fuzzy-Systeme*. Teubner-Verlag, 1993

[Kruse u. a. 1994] Kruse, R. ; Gebhardt, J. ; Klawonn, F.: *Foundation of Fuzzy Systems*. Chichster, England : J. Wiley & Son, 1994

[Łukasiewicz 1970] Łukasiewicz, J.: *Selected Works - Studies in Logic and the Foundations of Mathematics*. Amsterdam : North-Holland, 1970

[Mamdani und Assilian 1975] Mamdani, E.H. ; Assilian, S.: An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller. In: *International Journal of Man-Machine Studies* 7 (1975), Nr. 1, S. 1–13

[mayato 2008] mayato: *Data Mining Software 2009*. `http://www.mayato.com/downloads/Summary_mayato_Data-Mining-Study_2009.pdf`. 2008. – [Online; accessed 2009-02-10]

[Michie u. a. 1994] Michie, D. (Hrsg.) ; Spiegelhalter, D.J. (Hrsg.) ; Taylor, C.C. (Hrsg.): *Machine Learning, Neural and Statistical Classification*. Ellis Horwood Limited, 1994

[Murthy 1997] Murthy, S.K.: Automatic Construction of Decision Trees from Data: A Multi-Disciplinary Survey. In: *Data Mining and Knowledge Discovery* 2 (1997), S. 345–389

[Nakhaeizadeh 1999] Nakhaeizadeh, G.: *Data Mining: Theoretische Aspekte und Anwendungen*. Physica-Verlag, Heidelberg, Germany, 1999

[Nauck u. a. 1994] Nauck, D. ; Klawonn, F. ; Kruse, R.: *Neuronale Netze und Fuzzy-Systeme*. Vieweg Verlag Braunschweig Wiesbaden, 1994

[Nauck u. a. 1997] Nauck, D. ; Klawonn, F. ; Kruse, R.: *Foundations of Neuro-Fuzzy Systems*. John Wiley, 1997

[Nauck und Kruse 1998] Nauck, D. ; Kruse, R.: *Handbook of Fuzzy Computation*. Institute of Physics Publishing Ltd., Bristol, United Kingdom, 1998

[Nozaki u. a. 1996] Nozaki, K. ; Ishibuchi, H. ; Tanaka, H.: Adaptive Fuzzy Rule-Based Classification Systems. In: *IEEE Transactions on Fuzzy Systems* 4 (1996), Nr. 3, S. 238–250

[Poggio und Girosi 1989] Poggio, T. ; Girosi, F.: A Theory of Networks for Approximation and Learning / Massachusetts Institute of Technology. Cambridge, MA, USA : MIT, 1989. – Forschungsbericht. AIM-1140

[Quinlan 1993] Quinlan, J.R.: *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993

[Reilly u. a. 1982] Reilly, D.L. ; Cooper, L.N. ; Elbaum, C.: A Neural Model for Category Learning. In: *Biological Cybernetics*, 1982 (45), S. 35–41

[Rivest 1987] Rivest, R.L.: Learning Decision Lists. In: *Machine Learning*, 1987, S. 229–246

[Salzberg 1991] Salzberg, S.: A Nearest Hyperrectangle Learning Method. In: *Machine Learning*, 1991 (6), S. 251–276

[Scheffer 1995] Scheffer, T.: A Generic Algorithm for Learning Rules with Hierarchical Exceptions. In: *SBIA '95: Proceedings of the 12th Brazilian Symposium on Artificial Intelligence*. London, UK : Springer-Verlag, 1995, S. 181–190

[Schweizer und Sklar 1960] Schweizer, B. ; Sklar, A.: Statistical metric spaces. In: *Pacific Journal of Mathematics* 10 (1960), S. 313–334

[Skowron u. a. 2005] Skowron, A. ; Wang, H. ; Wojna, A. ; Bazan, J.: A Hierarchical Approach to Multimodal Classification. In: *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing* Bd. 3642/2005. Springer Berlin : Springer, 2005, S. 119–127

[Specht 1990] Specht, D.F.: Probabilistic Neural Networks. In: *Neural Networks*, 1990 (3), S. 109–118

[Takagi und Sugeno 1985] Takagi, T. ; Sugeno, M.: Fuzzy Identification of Systems and Its Appications to Modeling and Control. In: *IEEE Transactions on Systems, Man, and Cybernetics* SMC-15 (1985), Nr. 1, S. 116–132

[Tsoukalas und Uhrig 1996] Tsoukalas, L.H. ; Uhrig, R.E.: *Fuzzy and Neural Approaches in Engineering*. New York, NY, USA : John Wiley & Sons, Inc., 1996

[Wang und Mendel 1992] Wang, L.-X. ; Mendel, J.M.: Generating Fuzzy Rules by Learning from Examples. In: *IEEE Transactions on Systems, Man, and Cybernetics* 22 (1992), Nr. 6, S. 1313–1427

[Weislow u. a. 1989] Weislow, O.S. ; Kiser, R. ; Fine, D.L. ; Bader, J.P. ; Shoemaker, R.H. ; Boyd, M.R.: New soluble formazan assay for HIV-1 cytopathic effects: application to high flux screening of synthetic and natural products for AIDS antiviral activity. In: *Journal National Cancer Institute* 81 (1989), S. 577–586

[Wettschereck u. a. 1995] Wettschereck, D. ; Dietterich, T.G. ; Sutton, R.: An Experimental Comparison of the Nearest-Neighbor and Nearest-Hyperrectangle Algorithms. In: *Machine Learning*, 1995, S. 5–28

[Yager 1980] Yager, R.R.: An Approach to Inference in Approximate Reasoning. In: *International Journal of Man-Machine Studies* 13 (1980), S. 323–338

[Yager u. a. 1987] Yager, R.R. (Hrsg.) ; Ovchinnikov, S. (Hrsg.) ; Tong, R.M. (Hrsg.) ; Ngugen, H.T. (Hrsg.): *Fuzzy Sets and Applications*. New York : Wiley, 1987

[Zadeh 1965] Zadeh, L.A.: Fuzzy Sets. In: *Information and Control* 8 (1965), S. 338–353

[Zadeh 1998] Zadeh, L.A.: Information Granulation and Its Centrality in Human and Machine Intelligence. In: *Rough Sets and Current Trends in Computing*, 1998, S. 35–36

[Zimmermann 1995] Zimmermann, H.-J.: Prinzipien der Fuzzy Logic. In: *Spektrum der Wissenschaften*, Heidelberg, 1995, S. 90–93