Clusterverfahren zur datenbasierten Generierung interpretierbarer Regeln unter Verwendung lokaler Entscheidungskriterien

D I S S E R T A T I O N

zur Erlangung des akademischen Grades
Doktor der Ingenieurwissenschaften

genehmigt von der
Fakultät für Elektrotechnik und Informationstechnik
der Universität Dortmund

von
Lars Haendel

Fellbach 2003

Tag der mündlichen Prüfung:	26. Juni 2003
Hauptreferent:	Prof. Dr. rer. nat. H. Kiendl
Korreferent	Prof. Dr.-Ing. S. Engell

Vorwort

Die hier vorliegende Arbeit entstand während meiner Tätigkeit als Stipendiat des Graduiertenkollegs *Modellierung und modellbasierte Entwicklung komplexer technischer Systeme* der Deutschen Forschungsgemeinschaft an der Universität Dortmund. Sie wurde angeregt und betreut von Herrn Prof. Dr. rer. nat. H. Kiendl, auf den auch die Grundidee des hier entwickelten SMBC Algorithmus zurückgeht. Ich möchte ihm an dieser Stelle für die stete Förderung meiner Arbeit danken.

Herrn Prof. Dr.-Ing. S. Engell danke ich für die Übernahme des Korreferates und für das meiner Arbeit entgegengebrachte Interesse.

Der Deutschen Forschungsgemeinschaft danke ich für die umfangreiche Förderung des oben genannten Graduiertenkollegs.

Schließlich gilt mein Dank allen Mitarbeiterinnen und Mitarbeitern des Lehrstuhls und allen Stipendiatinnen und Stipendiaten des Graduiertenkollegs, die auf vielfältige Weise zum Gelingen dieser Arbeit beigetragen haben.

Fellbach, August 2003 Lars Haendel

Inhaltsverzeichnis

Kapitel 1

Einleitung

Modelle haben in vielen Fällen des alltäglichen oder technischen Lebens eine Bedeutung. Sie helfen, die Wirklichkeit besser zu verstehen, zu bewerten oder vorherzusehen. Ein Modell ist immer eine Abstraktion der Realität. Für ein zu modellierendes System wird festgelegt, welche (Teil-)Aspekte interessieren und daher betrachtet und modelliert werden sollen. Ziel ist es dann, ein Modell zu erstellen, das bezüglich der interessierenden Aspekte möglichst gut mit der Realität übereinstimmt.

Die Einsatzmöglichkeiten für Modelle sind vielfältiger Natur. So dienen Modelle zur Simulation, zur Kennfeldnachbildung, zur Objekterkennung und -klassifikation, zur Prognose, oder auch zum Wissenserwerb über ein System. Eine Simulation kann beispielsweise im Rahmen eines Reglerentwurfs, oder dessen Optimierung, hilfreich sein, denn sie kann Kosten sparen und Risiken vermeiden, die bei Experimenten am realen System aufgetreten wären. Die Objekterkennung und -klassifikation ist beispielsweise in der Qualitätskontrolle einsetzbar und eine Prognose eröffnet die Möglichkeit zur sinnvollen Anpassung aktueller bzw. zukünftiger Regelstrategien oder Handlungen. Der Wissenserwerb über ein System schließlich erlaubt es, die Auswirkungen von Veränderungen oder Beeinflussungen des Systems vorab abzuschätzen.

Ein möglicher Ansatz zur Erstellung eines Modells ist die sogenannte *mathematisch-physikalische* Modellierung. Dabei werden für die Wirkungszusammenhänge des Systems die entsprechenden beschreibenden Systemgleichungen aufgestellt und gelöst. Unbekannte Systemparameter können dann in Experimenten am realen System identifiziert werden. Die erhaltenen Modelle sind transparent. Sie ermöglichen analytische Betrachtungen, die unter Umständen Rückschlüsse auf das Systemverhalten für ein Kontinuum an möglichen Systemparametern oder Betriebssituationen zulassen. Oft wird diese Vorgehensweise auch als *deduktive* oder *white-box* Modellierung bezeichnet. Ein Nachteil der mathematisch-physikalischen Modellierung ist, daß diese ein tiefgreifendes Wissen über das zu modellierende System und oft auch einen erheblichen Arbeitsaufwand erfordert.

Eine Alternative ist die *datenbasierte* Modellierung. Bei dieser werden zunächst Daten, die möglichst alle relevanten Systemzusammenhänge oder Betriebssituationen abdecken, am realen System erhoben. Aus diesen Beobachtungsdaten wird dann induktiv mittels eines sogenannten *Lernverfahrens* ein Modell erstellt. Vorwissen über das zu modellierende System ist dabei nicht oder nur in stark reduziertem Umfang erforderlich. Dieser Ansatz wird auch als *induktive* oder *black-* bzw. *grey-box* Modellierung bezeichnet. Dabei ist die Einstufung zwischen *black-* und *grey-box* Modellierung subjektiv und spiegelt wider, in welchem Maße das erstellte Modell für einen Menschen verständlich, d.h. interpretierbar ist.

Ein möglicher Ansatz zur datenbasierten Modellierung ist das induktive Lernen von Wenn-Dann-Regeln. Regeln beschreiben jeweils einen Teilaspekt des zu modellierenden Systems und sind somit einzeln sowie als auch in ihrem Zusammenspiel für einen Menschen intuitiv verständlich. Ferner ist es möglich, Regeln wissensbasiert – beispielsweise durch Experteninterviews – aufzustellen. Dieses deduktiv erworbene Wissen über ein System kann mit induktiv erworbenen Wissen in einem einheitlichen Rahmen verarbeitet werden.

Induktion ist ein Teilgebiet des maschinellen Lernens und nicht zu verwechseln mit dem sehr ähnlichen Gebiet des sogenannten *Data-Mining*. Dieses zielt darauf ab, alle gültigen und interessanten Regeln oder Zusammenhänge in unübersichtlichen Datensammlungen und Datenbanken zu finden. Meist werden dabei statistische Methoden im Zuge einer interaktiven, explorativen Datenanalyse eingesetzt.

Struktur der Arbeit Zunächst wird in Kapitel 2 die Aufgabenstellung der datenbasierten Modellierung formal konkretisiert und der übliche Ablauf zur Modellerstellung beschrieben. Daran anschließend werden die Grundlagen regelbasierter Modelle eingeführt und mit dem Fuzzy-ROSA-Algorithmus ein bekanntes Lernverfahren zur automatischen, induktiven Generierung eines regelbasierten Modells skizziert. Nach einem kurzen Überblick über instanz-basierte Algorithmen und weitere Lernverfahren, und nach einem Exkurs in die COD-Problematik[1] und Methoden zur Eingangsgrößenselektion, wird eine Einführung in Clusterverfahren gegeben. Es folgen die in diesem Kontext wichtigen Abstandsfunktionen. Das Kapitel schließt mit einem kurzen Abriß über Möglichkeiten zum Einsatz von Clusterverfahren in der datenbasierten Modellierung und der Benennung der Ziele dieser Arbeit.

Kapitel 3 und 4 stellen die beiden im Rahmen dieser Arbeit entwickelten Clusterverfahren vor. Dies sind das *Positive and Negative Example-Based Clustering* (Pnc 2) und das *Supervised Modell-Based Clustering* (Smbc). Diese beiden Kapitel beginnen jeweils mit der Darstellung der dem jeweiligen Algorithmus zugrundeliegenden Idee, beschreiben dann eine vereinfachte Basis-Variante des Algorithmus und präsentieren dessen Pseudo-Code. Darauf folgen weitere Erläuterungen, Details und Erweiterungen und abschließend eine Einordnung der entwickelten Algorithmen in die bestehende Literatur.

In Kapitel 5 wird zunächst eine Einführung in mögliche Gütemaße zur Bewertung der Prognoseeignung eines Lernverfahrens gegeben und mit der Kreuz-Validierung und der mehrfachen Wiederholung Verfahren zur Ermittlung derselben vorgestellt. Ein Problem dabei ist die Einstellung der freien Parameter eines Lernverfahrens. Erfolgt diese nicht systematisch, kann dies die Validität der erhaltenen Ergebnisse beeinträchtigen. Zur Vermeidung dieser Schwierigkeit wird, basierend auf einer Arbeit von Salzberg, die Vorgehensweise der *harten* Validierung definiert. Es folgt eine Beschreibung der durchgeführten Experimente und Untersuchungen, sowie eine Darstellung und Diskussion der erhaltenen Ergebnisse.

In Kapitel 6 werden die wichtigsten Ergebnisse der Arbeit zusammengefaßt und ein Ausblick auf zukünftige Arbeiten gegeben.

Die Tabellen D.3 und D.4 in Anhang D erläutern die verwendeten speziellen Zähler, Indizes und Begriffe. Es wurde versucht, bevorzugt solche Literatur zu verwenden, welche gebührenfrei online verfügbar ist.

[1] engl.: *Curse of Dimensionality*

Kapitel 2

Grundlagen und Einordnung

In diesem Kapitel wird zunächst die Aufgabenstellung der datenbasierten Modellierung formal konkretisiert und der übliche Ablauf zur Modellerstellung beschrieben. Daran anschließend werden die Grundlagen regelbasierter Modelle eingeführt und mit dem Fuzzy-ROSA-Algorithmus ein bekanntes Lernverfahren zur automatischen, induktiven Generierung eines solchen regelbasierten Modells skizziert. Nach einem kurzen Überblick über instanz-basierte Algorithmen und weitere Lernverfahren und nach einem Exkurs in die COD-Problematik und Methoden zur Eingangsgrößenselektion, wird eine Einführung in Clusterverfahren gegeben. Es folgen die in diesem Kontext wichtigen Abstandsfunktionen. Das Kapitel schließt mit einem kurzen Abriß über Möglichkeiten zum Einsatz von Clusterverfahren in der datenbasierten Modellierung und der Benennung der Ziele dieser Arbeit.

2.1 Datenbasierte Modellierung

Aufgabenstellung Die der datenbasierten Modellierung zugrundeliegende Aufgabenstellung ist in Abbildung 2.1 illustriert. Gegeben sei ein System, für das ein Eingangsvektor $\mathbf{x}$ und eine zugehörige Ausgangsgröße y gemessen bzw. erhoben werden können. Der Eingangsvektor besteht aus einer oder mehreren Eingangsgrößen. Ziel ist es, ein Modell zu erstellen, das für einen gegebenen Eingangsvektor die Ausgangsgröße möglichst genau als Größe $\widehat{y}$ prognostiziert. Ein Tuple $(\mathbf{x}, y)$ von Eingangsvektor und zugehörigem Ausgangsgrößenwert bildet einen Datenpunkt P.

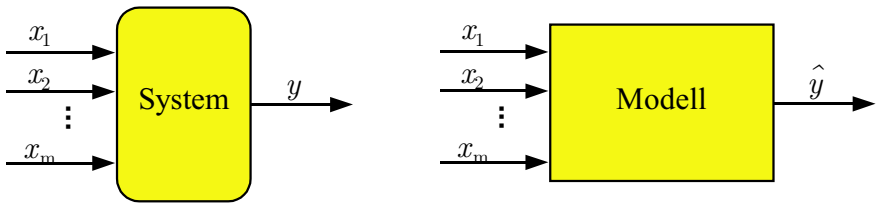


Abbildung 2.1: Aufgabenstellung der datenbasierten Modellierung.

Es wird davon ausgegangen, daß der dem System zugrundeliegende wahre Zusammenhang des Ein/Ausgangsverhaltens durch die sogenannte *true function* $f_{tf}(\mathbf{x})$ und einen nicht näher definierbaren Rauschterm e beschrieben werden kann zu

$$y = f_{tf}(\mathbf{x}) + e. \tag{2.1}$$

Die auftretenden Eingangsvektoren $\mathbf{x}$ gehorchen dabei der Verteilung $\mu_{tf}(\mathbf{x})$. Sowohl die Funktion $f_{tf}(\mathbf{x})$, als auch die Verteilung $\mu_{tf}(\mathbf{x})$ sind im Allgemeinen nicht bekannt. Bei Kenntnis der Funktion $f_{tf}(\mathbf{x})$ wäre die Aufgabenstellung vollständig gelöst. Ziel ist es daher, anhand gemessener Datenpunkte eine möglichst gute Approximation dieser Funktion zu lernen.

Nominale, ordinale und kontinuierliche Größen Bezüglich der Art der gemessenen Größen sind drei verschiedene Typen zu unterscheiden.

- *Nominale Größen* nehmen ausschließlich symbolische Werte an, die nicht in einer Rangfolge bezüglich einer kleiner-größer Relation angeordnet werden können. Beispielsweise kann es sich dabei um die Farbe eines Objektes handeln, die die möglichen Symbole *rot*, *grün* und *blau* annehmen kann. Im Rahmen dieser Arbeit wird prinzipiell davon ausgegangen, daß die verschiedenen Symbole einer nominalen Größe als von eins an aufsteigende natürliche Zahlen kodiert sind.

- *Ordinale Größen* nehmen, ebenso wie nominale Größen, ausschließlich symbolische Werte an, jedoch lassen sich die einzelnen Symbole anhand einer kleiner-größer Relation ordnen. Beispielsweise stellt eine qualitative Temperaturangabe in der Form *kalt*, *normal*, *warm* und *heiß* oder eine in Jahren gemessene Altersangabe eine ordinale Größe dar. Im Rahmen dieser Arbeit werden ordinale Größen mit nur wenigen verschiedenen Symbolen, wie bei der obigen qualitativen Temperaturangabe, als nominale Größen behandelt. Wenn jedoch, wie bei der obigen Altersangabe, sehr viele verschieden Symbole auftreten können, wird eine solche Größe als kontinuierlich behandelt.

- *Kontinuierliche Größen* können beliebige, nur durch die Auflösegenauigkeit und den Wertebereich der verwendeten Meßapparatur begrenzte, reelle Werte annehmen. Beispielsweise stellt die Messung einer Temperatur in Grad Celsius eine kontinuierliche Größe dar.

Ablauf der datenbasierten Modellierung Bei der datenbasierten Erstellung eines Modells für ein gegebenes System wird nach [Sla01], wie in Abbildung 2.2 skizziert, vorgegangen. Es sei angemerkt, daß es in der Praxis selten möglich ist, die dargestellten Schritte streng nacheinander zu bearbeiten. Vielmehr handelt es sich um einen iterativen Vorgang, bei dem immer wieder verschiedene Varianten und Algorithmen ausprobiert werden, und somit der ganze Prozeß, oder Teile desselben, mehrmals wiederholt werden müssen.

- *Datenaufbereitung:* Zunächst wird am realen System eine Stichprobe mit Rohdaten aufgenommen, indem für eine möglichst repräsentative Auswahl von Objekten oder Betriebssituationen Datenpunkte, die aus einem primären Eingangsvektor und einem zugehörigen Ausgangsgrößenwert bestehen, erhoben werden. Da beim Messen am realen System oftmals Fehler auftreten, ist es sinnvoll, augenscheinlich fehlerhafte oder inkonsistente Datenpunkte aus der Stichprobe zu entfernen. Ebenso können einfache Methoden zur Vorverarbeitung der Daten, wie beispielsweise eine Normierung oder Glättung der Werte einer primären Eingangsgröße, zum Einsatz kommen.

- *Merkmalsbildung:* Die Merkmalsbildung bezeichnet die anschließende Überführung der primären in sekundäre Eingangsvektoren mittels geeignet definierter Berechnungsvorschriften. Ziel ist es dabei, anhand der sekundären Eingangsvektoren besser die Ausgangsgröße prognostizieren zu können, als dies anhand der primären Eingangsvektoren möglich gewesen wäre. Der Vorgang der Merkmalsbildung ist optional, d.h. ggf. werden

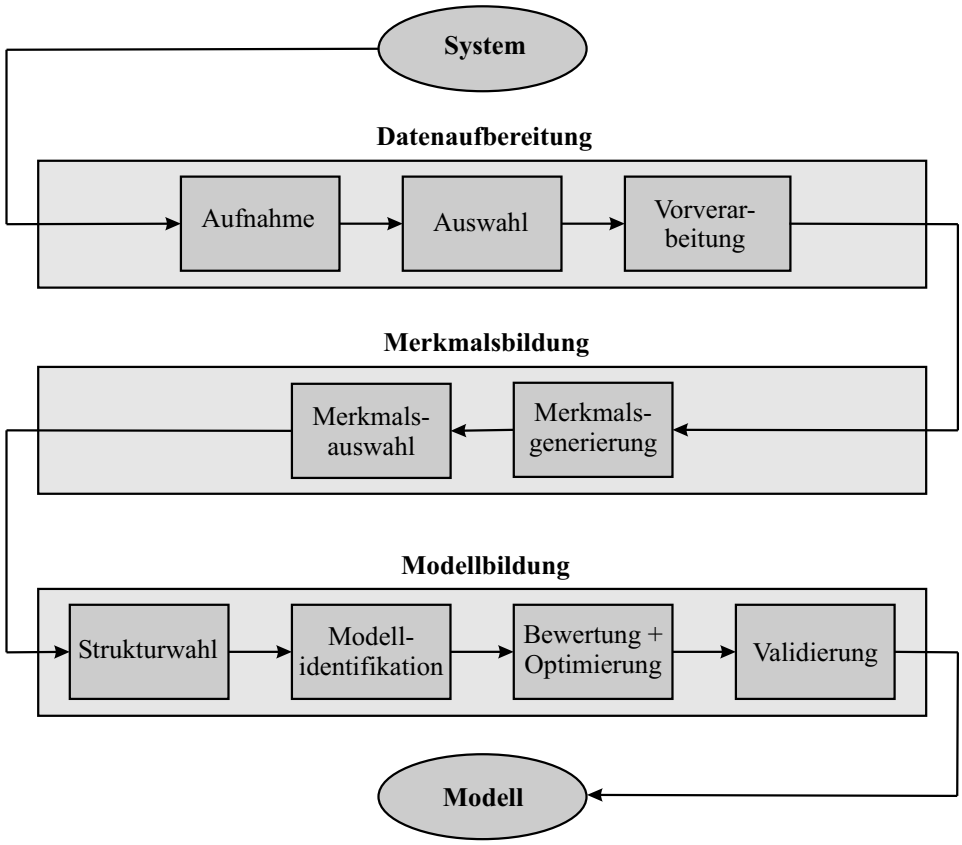


Abbildung 2.2: Vorgehensweise zur datenbasierten Modellierung. *Quelle:* [Sla01]

die gemessenen primären Eingangsvektoren bereits als geeignet angesehen, um darauf basierend ein Modell zu lernen. Zur Illustration betrachte man das folgende Beispiel: Für ein zu prüfendes Objekt werde das während eines Prüflaufes emittierte Geräusch mittels eines Körperschallsensors gemessen und daraus das dazugehörige Frequenzspektrum ermittelt. Nun werde der Frequenzanteil für einige bestimmte, für die Beurteilung des Geräusches interessierende, Frequenzen berechnet. Diese Frequenzanteile stellen Merkmale dar. Um die Anzahl der sekundären Eingangsgrößen zu reduzieren, kann anschließend eine Eingangsgrößenselektion erfolgen. Für einen knappen Überblick über mögliche automatische, datenbasierte Ansätze siehe Kapitel 2.6.

- *Modellbildung:* Die aus den sekundären Eingangsvektoren[1] und der zugehörigen Ausgangsgröße bestehenden Datenpunkte bilden die *Lernstichprobe*. Die in der Abbildung verwendeten Begriffe *Strukturwahl* und *Modellidentifikation* sind hier auch im weiteren Sinne als Wahl eines geeigneten Lernverfahrens und als Einsatz desselben zum Lernen eines Modells zu verstehen. Mittels der Lernstichprobe wird datenbasiert ein Modell erstellt, das ggf. noch mit Hilfe nachgeschalteter Verfahren optimiert wird. Eine anschließende Validierung dient zur Kontrolle der *Generalisierungsfähigkeiten* des gelernten Modells. Hierzu wird die Prognosegüte des gelernten Modells zur Vorhersage des Ausgangsgrößenwertes zu gegebenem Eingangsvektor bezüglich einer *Teststichprobe* ermittelt. Diese Teststichprobe besteht aus weiteren am System gemessenen Datenpunkten. Eine detaillierte Beschreibung verschiedener Maße zur Bewertung der Prognosegüte findet sich in Kapitel 5.1.

[1] Im folgenden wird der Einfachheit halber wieder der Begriff *Eingangsvektor* verwendet.

Regression versus Klassifikation Anhand der Art der Ausgangsgrößenwerte sind zwei Fälle zu unterscheiden, die zu grundlegend voneinander verschiedenen Lernaufgaben führen. Für nominale Ausgangsgrößen handelt es sich um den Fall der Klassifikation[2], für kontinuierliche Ausgangsgrößen dagegen um den Fall der Regression. In dieser Arbeit wird vorwiegend der Fall der Klassifikation betrachtet und der Fall der Regression im wesentlichen im Rahmen der im folgenden beschriebenen *Trivialerweiterung* behandelt: Eine kontinuierliche Ausgangsgröße kann, wie in Kapitel 2.8.2 beschrieben, durch Diskretisierung in eine ordinale Größe überführt werden. Wird diese ordinale Ausgangsgröße dann als nominal betrachtet, ergibt sich ein Klassifikationsproblem, das als ein solches – ggf. mit leicht verändertem Prognosemechanismus des verwendeten Lernverfahrens – gelöst werden kann.

Batch versus inkrementelles Lernen Bezüglich der Arbeitsweise eines Lernverfahrens sind der sogenannte *batch* und der *inkrementelle* Fall zu unterscheiden. Im ersteren Fall stehen alle Datenpunkte der gegebenen Lernstichprobe dem Lernverfahren gleichzeitig zur Verfügung und werden gleichzeitig bearbeitet. Im letzteren Fall ist es – meist aus Gründen des Rechen- und Speicheraufwands – nicht möglich, alle Datenpunkte gleichzeitig zu erfassen oder zu bearbeiten. Dann werden die Datenpunkte iterativ nacheinander betrachtet. Für viele Lernverfahren sind sowohl batch als auch inkrementelle Varianten möglich. Da mit den Batch-Varianten mehr Information gleichzeitig ausgewertet werden kann, lassen sich damit tendenziell bessere Prognosegüten – wenn auch meist um den Preis eines größeren Rechenaufwands – erzielen.

Bestandteile eines Lernverfahrens Der datenbasierten Modellierung liegt die sogenannte *Ähnlichkeitshypothese* zugrunde. Diese geht davon aus, daß im Eingangsraum zusammenhängende Regionen der gleichen Ausgangsklasse bzw. ähnlicher Ausgangsgrößenwerte vorhanden sind. Somit kann die Kenntnis des Ausgangsgrößenwertes zu einer Eingangsraumposition genutzt werden, um auch für umliegende Eingangsraumpositionen Rückschlüsse auf den zugehörigen Ausgangsgrößenwert zu ziehen. Ein datenbasiertes Lernverfahren besteht aus einem Repräsentationsformalismus, einem Suchverfahren und einem Prognosemechanismus [Dom97].

- *Repräsentation:* Durch die Wahl der Repräsentation wird festgelegt, wie die gelernten Zusammenhänge des Ein/Ausgangsverhaltens operational, d.h. anwendbar, repräsentiert werden. Bekannte Varianten zur Repräsentation sind beispielsweise Wenn-Dann-Regeln, diskriminierende Hyperebenen und lineare Funktionsansätze.
- *Suche:* Durch die Suche wird festgelegt, wie zu gegebener Lernstichprobe aus der Menge aller anhand der gewählten Repräsentation möglichen Modelle ein möglichst gut passendes ausgewählt wird.
- *Prognosemechanismus:* Durch die Wahl des Prognosemechanismus wird festgelegt, wie anhand der gelernten Zusammenhänge des Ein/Ausgangsverhaltens zu gegebener Eingangsraumposition ein Prognosewert für die Ausgangsgröße gebildet wird. Teilweise wird dieser Aspekt auch zum Repräsentationsformalismus gerechnet.

Aufgrund der Wahl von Repräsentation, Suche und Prognosemechanismus besitzt jedes Lernverfahren einen mehr oder minder ausgeprägten sogenannten *Bias*: Das Lernverfahren bevorzugt bestimmte Generalisierungen bzw. Annahmen über bestimmte Zusammenhänge in der Lernstichprobe gegenüber anderen und ist damit umso erfolgreicher, je mehr der *Bias* mit den Eigenarten des zu modellierenden Systems übereinstimmt.

[2]Um Mißverständnisse zu vermeiden sei darauf hingewiesen, daß einige Autoren den Prozeß der Klassenbildung selbst als Klassifikation bezeichnen, hier jedoch davon ausgegangen wird, daß dieser Prozeß bereits abgeschlossen ist.

2.2 Regelbasierte Modelle

Die Fuzzy-Logik [Zad65, Zad68] wurde 1965 von ZADEH eingeführt und bildet die Grundlage der Fuzzy-Modellierung. Im folgenden wird eine kurze Einführung in die Fuzzy-Logik und die Funktionsweise darauf aufbauender regel-basierter Fuzzy-Modelle nach MAMDANI [MG81] gegeben. Die Ausführungen sind an [Kie97] angelehnt und die Bezeichnungen orientieren sich weitestgehend an der VDI/VDE-Richtlinie 3550 [VDI02]. Für eine online verfügbare Referenz siehe [KHJ97].

Repräsentation Regelbasierte Fuzzy-Modelle nach MAMDANI bestehen aus einer Menge von linguistischen Regeln der Form

$$\text{WENN } \textit{Prämisse} \text{ DANN } \textit{Konklusion} \tag{2.2}$$

Die Prämisse enthält logisch miteinander verknüpfte linguistische Aussagen über die Eingangsgrößen und die Konklusion enthält eine Aussage über den Wert der Ausgangsgröße. Jede Regel beschreibt in qualitativer Form lokal das Ein/Ausgangsverhalten für das Gebiet des Eingangsgrößenraums, in dem die Prämisse erfüllt ist. Im Gegensatz zu sogenannten *harten* Regeln, bei denen die Prämisse entweder erfüllt oder nicht erfüllt sein und somit nur die Wahrheitswerte 0 und 1 annehmen kann, wird bei einer Fuzzy-Regel das Konzept der Wahrheitswerte einer logischen Aussage auf das Kontinuum $[0, 1]$ erweitert. Ein Wert von 1 bzw. 0 bedeutet wie bisher, daß eine logische Aussage erfüllt bzw. nicht erfüllt ist. Bei Werten zwischen 0 und 1 ist eine logische Aussage teilweise erfüllt. Die Verknüpfung mehrerer linguistischer Aussagen geschieht durch verallgemeinerte logische Operatoren auf den Wahrheitswerten $[0, 1]$. Mögliche Fuzzy-UND-Operatoren sind

$$\begin{array}{lcll} \mu_1 \wedge \mu_2 & = & \min(\mu_1, \mu_2) & \text{(Minimum)} \\ \mu_1 \wedge \mu_2 & = & \mu_1 \mu_2 & \text{(Algebraisches Produkt)} \end{array} \tag{2.3}$$

und mögliche Fuzzy-ODER-Operatoren[3] sind

$$\begin{array}{lcll} \mu_1 \vee \mu_2 & = & \max(\mu_1, \mu_2) & \text{(Maximum)} \\ \mu_1 \vee \mu_2 & = & \mu_1 + \mu_2 - \mu_1 \mu_2 & \text{(Algebraische Summe)} \\ \mu_1 \vee \mu_2 & = & \mu_1 + \mu_2 & \text{(Gewöhnliche Summe) .} \end{array} \tag{2.4}$$

Mittels einer sogenannten Zugehörigkeitsfunktion (ZGF) kann eine kontinuierliche Größe in den Zugehörigkeitsgrad zu einem linguistischen Wert, d.h. in eine beschreibende qualitative linguistische Größe überführt werden. Dieser Vorgang wird als *Fuzzifizierung* bezeichnet. Die typischerweise für Eingangsgrößen verwendeten ZGF haben die Form eines Dreiecks, Trapezes, Rechtecks oder einer Gaußglocke. Für die Ausgangsgröße wird oft ein sogenanntes *Singleton* verwendet, das nur an einer einzigen Stelle einen Wert von 1 annimmt und ansonsten gleich 0 ist. Ein Beispiel für die Wahl der ZGF, mit denen ein kontinuierlicher Temperaturwert in die Zugehörigkeiten zu den linguistischen Größen *kalt*, *normal*, *warm* und *heiß* überführt werden kann, ist in Abbildung 2.3 skizziert.

[3] Die gewöhnliche Summe ist streng genommen kein Fuzzy-ODER-Operator, führt jedoch in Verbindung mit geeigneten nachgeschalteten Defuzzifizierungsmethoden zu besonders einfachen Kennfeldern. Deshalb wird die gewöhnliche Summe oft in der Praxis verwendet.

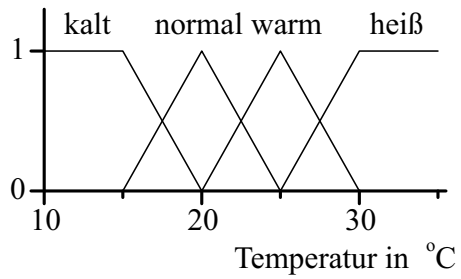


Abbildung 2.3: Beispielhafte Wahl der ZGF, mit denen kontinuierliche Temperaturwerte in die Zugehörigkeiten zu den linguistischen Größen *kalt*, *normal*, *warm* und *heiß* überführt werden können. *Quelle:* [Sla01]

Prognosemechanismus Ein Regelsatzes wird wie folgt zur Prognose des Ausgangsgrößenwertes zu gegebener Eingangsraumposition verwendet. Nach der Fuzzifizierung der Eingangsgrößenwerte wird im Zuge der *Aggregation* für jede Regel der Wahrheits- oder Erfülltheitsgrad der Regelprämisse ermittelt und dieser zur *Aktivierung* mit der Konklusion der jeweiligen Regel UND-verknüpft. Diese aktivierten Regeln werden dann durch die *Akkumulation* ODER-verknüpft und so das *ausgangsseitige Empfehlungsgebirge* erhalten. Für einen Regelsatz mit K Regeln ergibt sich das ausgangsseitige Empfehlungsgebirge formal zu

$$\mu(\mathbf{x}, y) = \bigvee_{t=1}^{K} \left(\mu_t(\mathbf{x}) \wedge \mu_t(y)\right) \ . \tag{2.5}$$

Dabei bezeichnet $\mu_t(\mathbf{x})$ den Erfülltheitsgrad der Prämisse in Abhängigkeit von der Eingangsraumposition und $\mu_t(y)$ bezeichnet den Erfülltheitsgrad der Konklusion der t-ten Regel in Abhängigkeit vom Ausgangsgrößenwert. Aus dem ausgangsseitigen Empfehlungsgebirge wird durch die sogenannte *Defuzzifizierung* der prognostizierte Ausgangsgrößenwert $\widehat{y}$ abgeleitet. Die beiden bekanntesten Defuzzifizierungsmethoden sind die *Schwerpunktmethode* (COG)

$$\widehat{y} = \frac{\int_{-\infty}^{+\infty} y \ \mu(\mathbf{x}, y) \ dy}{\int_{-\infty}^{+\infty} \mu(\mathbf{x}, y) \ dy} \tag{2.6}$$

und die *Maximummethode* (MOM)

$$\widehat{y} \quad \text{mit} \quad \mu(\mathbf{x}, \widehat{y}) = \max \mu(\mathbf{x}, y) \ . \tag{2.7}$$

Wahl der Fuzzy-Operatoren Im Rahmen dieser Arbeit wird als Fuzzy-UND-Operator zur Aggregation und Aktivierung das algebraische Produkt und als Fuzzy-ODER-Operator zur Akkumulation die gewöhnliche Summe verwendet. Als ausgangsseitige ZGF werden stets Singletons gewählt. Anschaulich gesprochen wird bei der Schwerpunktmethode ein Kompromiß aus den verschiedenen empfohlenen Ausgangsgrößenwerten gebildet, während die Maximummethode den am meisten empfohlenen Wert prognostiziert. Bei Klassifikationsaufgaben ist ein Kompromiß zwischen verschiedenen Ausgangsklassen unerwünscht, für Regressionsaufgaben kann sich dagegen eine Mittelwertbildung aus verschiedenen empfohlenen Ausgangsgrößenwerten günstig auf die Prognosegüte auswirken. Daher wird hier im Falle der Klassifikation stets die MOM- und im Falle der Regression stets die COG-Defuzzifizierung eingesetzt. Durch die

gerade beschriebene Wahl der Operatoren und Vorgehensweisen vereinfachen sich die Berechnungsvorschriften für die COG-Defuzzifizierung zu

$$\widehat{y} = \frac{\sum_{t=1}^{K} \mu_t(\mathbf{x})\ y_t}{\sum_{t=1}^{K} \mu_t(\mathbf{x})} \tag{2.8}$$

und jene für die MOM-Defuzzifizierung zu

$$\widehat{y} = y_t \quad \text{mit} \quad t = \arg\max_{t=1}^{K} \mu_t(\mathbf{x})\ . \tag{2.9}$$

Dabei bezeichnet y_t denjenigen Ausgangsgrößenwert, für den das als ausgangsseitige ZGF verwendete Singleton der t-ten Regel den Wert 1 annimmt.

TSK-Modelle Als Erweiterung des Fuzzy-Modells nach Mamdani ist von Takagi, Sugeno und Kang das sogenannte Takagi-Sugeno-Kang-Modell (TSK) [TS85] entwickelt worden. Dessen wesentlicher Unterschied im Vergleich zum Mamdani-Modell besteht darin, daß als Konklusion nicht mehr linguistische Ausdrücke, sondern von den Eingangsvektoren abhängige Funktionen verwendet werden. Damit definiert jede solche TSK-Regel

$$\text{Wenn } \textit{Prämisse} \text{ Dann } y = f(\mathbf{x}) \tag{2.10}$$

ein lokales Modell für den Bereich des Eingangsraumes, in dem die Regelprämisse in einem Grade $\mu_t(\mathbf{x}) > 0$ zutrifft. Der prognostizierte Ausgangsgrößenwert $\widehat{y}$ ergibt sich als gewichteter Mittelwert über die einzelnen empfohlenen Werte analog zu Gl. (2.8), wenn anstelle des festen Wertes y_t jeweils der Funktionswert $f_t(\mathbf{x})$ verwendet wird.

Regelbewertung Es ist möglich, jede Regel mit einem sogenannten *Glaubensgrad* ϱ zu versehen. Dies ist eine Bewertung, die angibt, wie zuverlässig oder wie gut eine Regel ist. Die Glaubensgrade werden vor Durchführung der Aktivierung mit den Erfülltheitsgraden der jeweiligen Regelprämissen unter Verwendung des gewählten Fuzzy-Und-Operators verknüpft. Verschiedene Methoden zur Regelbewertung sind in [Jes00] zu finden. Im folgenden wird mit der Trefferquote exemplarisch die bekannteste Regelbewertung beschrieben. Die Trefferquote ergibt sich als bedingte Auftrittswahrscheinlichkeit $P(c|p)$ der Konklusion c wenn die Prämisse p erfüllt ist zu

$$\varrho = P(c|p)\ . \tag{2.11}$$

Für eine Fuzzy-Regel wird diese Wahrscheinlichkeit im einfachsten Falle anhand der Erfülltheitsgrade $\mu_t(\mathbf{x}_i)$ und $\mu_t(y_i)$ von Prämisse und Konklusion für die verschiedenen Datenpunkte i der Lernstichprobe geschätzt zu

$$\widehat{P}(c|p) = \frac{\sum_{i=1}^{N} \mu_t(\mathbf{x}_i)\ \mu_t(y_i)}{\sum_{i=1}^{N} \mu_t(\mathbf{x}_i)}\ . \tag{2.12}$$

Diese Schätzung aufgrund der relativen Häufigkeiten ist jedoch zu optimistisch, wenn eine Prämisse nur für sehr wenige oder im Extremfall nur für einen einzigen Datenpunkt der Lernstichprobe erfüllt ist. Dann ist nämlich zu erwarten, daß die tatsächliche Trefferquote, d.h. die Trefferquote, die sich ergeben würde, wenn unendlich viele Datenpunkte aufgenommen und ausgewertet werden könnten, niedriger ist. Das häufig verwendete *Laplace-Verhältnis* [Goo65, Nib87] korrigiert den mittels Gl. (2.12) erhaltenen Wert unter Berücksichtigung der Anzahl der möglichen Symbole der Ausgangsgröße S_y zu

$$\widehat{P}(c|p) = \frac{1 + \sum_{i=1}^{N} \mu_t(\mathbf{x}_i)\ \mu_t(y_i)}{S_y + \sum_{i=1}^{N} \mu_t(\mathbf{x}_i)} \ . \tag{2.13}$$

Ein alternativer, im nächsten Abschnitt im Zusammenhang mit dem Fuzzy-ROSA-Algorithmus näher beschriebener Ansatz besteht darin, die berechneten empirischen Häufigkeiten durch die untere Grenze des einseitigen Konfidenzintervalls

$$I^{\alpha} = [P_U^{\alpha}, 1] \tag{2.14}$$

zu ersetzen. Konfidenzintervalle beschreiben den Wertebereich, in dem der tatsächliche Wert einer geschätzten Größe sich mit einer Wahrscheinlichkeit von $(1 - \alpha)$ befinden wird [HEK95]. Der Parameter α wird als Irrtumswahrscheinlichkeit bezeichnet und gibt die Wahrscheinlichkeit dafür an, daß sich der tatsächliche Wert außerhalb des Konfidenzintervalls befindet.

Für Regeln, deren Prämisse für viele Lerndatenpunkte erfüllt ist, nähert sich für beide oben genannte Ansätze der korrigierte Schätzwert asymptotisch an den unkorrigierten Schätzwert an. In [Fri96] wurde in einem Experiment gezeigt, daß die untere Grenze des einseitigen Konfidenzintervalls $I^{95\%}$ im Gegensatz zum Laplace-Verhältnis zu einer eher pessimistischen Abschätzung des tatsächlichen Wertes neigt.

2.2.1 Der Fuzzy-ROSA-Algorithmus

Ein Verfahren zum datenbasierten Lernen von regel-basierten Modellen ist der in [KK89] von Kiendl und Krabs erstmalig vorgestellte ROSA-Algorithmus[4]. In [Kro99] wird das Verfahren zum Fuzzy-ROSA-Algorithmus (Fr) für die Generierung von Mamdani-Fuzzy-Modellen verallgemeinert. Seither ist der Fr-Algorithmus in verschiedenen Richtungen erfolgreich weiterentwickelt worden. So werden in [Jes00] verschiedene Test- und Bewertungsverfahren untersucht, in [Sla01] wird eine systematische Vorgehensweise zur Anwendung entwickelt, in [Kra01] wird der Algorithmus für die Generierung von TSK-Modellen erweitert und in [SNK01] wird ein Verfahren zur gleichzeitigen Selektion relevanter Eingangsgrößen und Optimierung der benötigten ZGF vorgestellt. Im folgenden wird die Grundidee und der prinzipielle Ablauf des Fr-Algorithmus für die Generierung von Mamdani-Fuzzy-Modellen skizziert.

Zunächst werden geeignete eingangs- und ausgangsseitige ZGF definiert. Zur Repräsentation werden nun Fuzzy-Regeln der Form (2.2) verwendet, wobei die Prämisse aus Und-verknüpften linguistischen Aussagen über eine, mehrere oder alle Eingangsgrößen besteht. Die Anzahl der miteinander verknüpften linguistischen Terme einer Prämisse wird als *Verknüpfungstiefe* der Regel bezeichnet. Wenn die Verknüpfungstiefe einer Regel gleich der Anzahl der Eingangsgrößen ist, wird diese Regel als *vollständige* Regel bezeichnet; andernfalls handelt es sich um eine

[4] Abkürzung für: *Rule Oriented Statistical Analysis*

generalisierende Regel. Die Art der Eingangsraumgebiete, die mit derartigen Regelprämissen adressiert werden können, ist in Abbildung 2.4 skizziert.

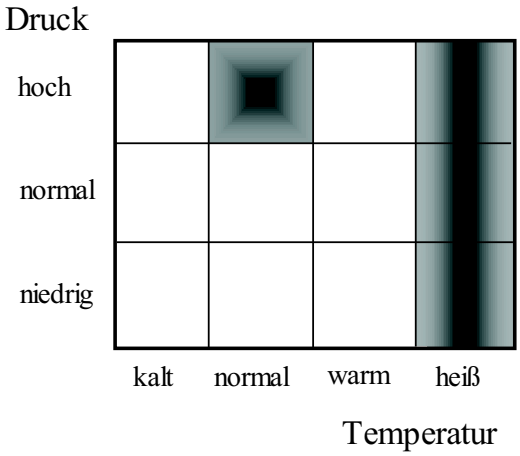


Abbildung 2.4: Zweidimensionales Beispiel für die beim FR-Algorithmus adressierbaren Eingangsraumgebiete.

Die Grundidee des FR-Algorithmus besteht darin, das Problem der Generierung eines kompletten Regelsatzes auf das einfachere Problem der Generierung von einzelnen, lokal vernünftigen Regeln zu reduzieren. D.h. im Gegensatz zu anderen Ansätzen, bei denen der komplette Regelsatz gleichzeitig bewertet und optimiert wird (Pittsburgh-Ansatz), wird der Regelsatz beim FR-Algorithmus sukzessive aus individuell getesteten Regeln aufgebaut (Michigan-Ansatz). Der zu bearbeitende Suchraum besteht somit aus der Menge aller Regeln, die anhand der oben beschriebenen Repräsentation gebildet werden können. Zur Suche in diesem Raum werden zunächst alle potentiellen Regeln der Verknüpfungstiefe 1 als sogenannte *Hypothesen* aufgestellt und mittels eines Regeltests bewertet. Besteht eine Hypothese diesen Test, so wird sie zur Regel erhoben, mit einem Bewertungsindex versehen und in den Regelsatz eingefügt. Anschließend wird bis zu einer vorgegebenen maximalen Verknüpfungstiefe mit der nächsthöheren Verknüpfungstiefe fortgefahren.

Regeltest Beispielhaft werden im folgenden mit der *Trefferquote* und der *konfidenten Trefferquote* zwei Varianten eines Regeltests und der jeweils dazugehörende Bewertungsindex vorgestellt. Für weitere Tests sei auf [Jes00] verwiesen. Eine Hypothese besteht den Regeltest der Trefferquote bzw. den Regeltest der konfidenten Trefferquote und wird mit dem angegebenen Bewertungsindex ϱ versehen, wenn die aufgrund der relativen Häufigkeiten nach Gl. (2.12) berechnete Trefferquote $\widehat{P}(c|p)$ bzw. die nach Gl. (2.14) mittels Konfidenzintervallen nach unten abgeschätzte Trefferquote $\widehat{P}_U^\alpha(c|p)$ einen vorgegebenen Schwellwert θ überschreitet. Eine übliche Wahl des Schwellwertes ist $\theta = 50\%$. Zusätzlich kann als weitere Bedingung gefordert werden, daß die Hypothese einen minimalen *Daten-Support* aufweist, d.h., daß die Fuzzy-Summe derjenigen Datenpunkte der Lernstichprobe die Prämisse und Konklusion erfüllen einen weiteren – oftmals zu 1 gewählten – Schwellwert überschreiten muß.

$$\begin{array}{lcll} \widehat{P}(c|p) & > & \theta & \varrho = \widehat{P}(c|p) \\ \widehat{P}_U^\alpha(c|p) & > & \theta & \varrho = \widehat{P}_U^\alpha(c|p) \end{array} \tag{2.15}$$

Suchraum Der Suchraum läßt sich anhand einer *spezieller-genereller Relation* halb-ordnen. Eine Regel a ist spezieller als eine Regel b, wenn beide Regeln dieselbe Konklusion haben und für jede Eingangsraumposition, die die Prämisse der Regel a erfüllt, auch die Prämisse der Regel b in gleichem oder höherem Maße erfüllt ist. Die Halb-Ordnung der Regeln ermöglicht es zum einen, während der Regelgenerierung, d.h. beim Einfügen neuer Regeln in den Regelsatz, redundante Regeln zu entfernen. Dabei gilt eine Regel genau dann als redundant, wenn im Regelsatz eine generellere Regel existiert, die den gleichen oder einen höheren Bewertungsindex aufweist. Zum anderen ist es möglich, den Suchraum sicher zu beschneiden, denn wenn eine Regel aufgrund zu geringen Daten-Supports den Regeltest nicht bestanden hat, so wird auch jede speziellere Regel den Regeltest ebenfalls nicht bestehen und braucht daher gar nicht mehr als Hypothese aufgestellt zu werden. Dadurch läßt sich der Rechenaufwand bei vollständiger Durchmusterung aller zulässigen Regeln mittels einer üblichen Tiefensuche[5] signifikant reduzieren [Sla01].

2.3 Instanz-basierte Lernverfahren

Die sogenannten *instanz-basierten Lernverfahren*[6] (IBL) speichern alle oder einige, in einem Vorverarbeitungsschritt ausgewählte, Lerndatenpunkten und benutzen diese, um die Ausgangsgröße zu gegebener Eingangsraumposition zu prognostizieren. Kennzeichnend ist dabei, daß – abgesehen von evtl. Vorverarbeitungsschritten – die Analyse der Lerndaten erst zu dem Zeitpunkt erfolgt, zu dem eine Prognose für eine bestimmte Eingangsraumposition benötigt wird. Der bekannteste IBL-Algorithmus ist der *k-Nearest-Neighbor-Algorithmus* (kNN) [CH67, DH73, Das91]. Die Prognose erfolgt bei diesem Algorithmus, indem die Ausgangsgrößenwerte derjenigen k Lerndatenpunkte, die zu der zu prognostizierenden Eingangsraumposition am nächsten liegen, im Sinne eines Mehrheitsentscheides oder einer gewichteten Mittelwertbildung ausgewertet werden. Die nächstliegenden Lerndatenpunkte werden auch als *Nachbarn* bezeichnet. Sie werden anhand einer geeignet gewählten Abstandsfunktion ermittelt. Für den Fall von $k = 1$ wird die Abkürzung des Algorithmus oftmals auch zu NN vereinfacht. Die Auswertung im Sinne eines Mehrheitsentscheides kommt überwiegend im Falle der Klassifikation zum Einsatz. Wenn verschiedene Ausgangsklassen gleich oft unter den nächsten Nachbarn vertreten sind, muß dieser Konflikt geeignet – etwa indem die Ausgangsklasse mit der höchsten a priori Wahrscheinlichkeit prognostiziert wird – aufgelöst werden. Im Falle der Regression dagegen wird meist die Auswertung im Sinne eines gewichteten Mittelwerts verwendet. Der bekannteste Ansatz dazu ist der – hier für den Fall $k = N$ angegebene – *Nadaraya-Watson-Estimator* [Nad64, Wat64]

$$\widehat{y} = \frac{\sum_{i=1}^{N} F(d_i)\ y_i}{\sum_{i=1}^{N} F(d_i)} \ . \tag{2.16}$$

Dabei bezeichnet F eine beliebige eindimensionale Kernelfunktion, die üblicherweise stetig,

[5] Eine Tiefen- oder Breitensuche strukturiert den Suchraum als Baum. Jeder Knoten entspricht einer Regel. Die Nachfolger eines Knotens werden gebildet, indem die nächst spezielleren Regeln aufgestellt werden, wobei geeignet das mehrfache Aufstellen einer Regel vermieden werden muß. Ein Knoten muß dann nicht mehr expandiert werden, wenn die zugehörige Regel einen zu geringen Daten-Support aufweist.

[6] engl: *Instance-Based-Learning* – In dieser Arbeit wird der Begriff IBL analog zu [Aha95b, Dom97] synonym für die Begriffe *Memory-Based-Learning* und *Nearest-Neighbor-Learning* verwendet und im erweiterten Sinne auch für die Begriffe *Exemplar-Based-Generalization*, *Case-Based-Learning* und *Kernel-Based-Learning* gebraucht.

beschränkt und integrierbar gewählt wird und d_i bezeichnet den anhand einer geeigneten Abstandsfunktion berechneten Abstand des i-ten Datenpunktes der N-elementigen Lernstichprobe zu der zu prognostizierenden Eingangsraumposition. Man beachte die Ähnlichkeit von Gl. (2.16) zu Gl. (2.8) und zu den Gln. (4.17) und (4.22).

Die Vor- und Nachteile des kNN-Algorithmus bzw. der IBL-Algorithmen sind [Dom97, WM00a]:

- Vorteile
  - intuitiv und einfach zu verstehende Grundidee
  - einfach und schnell zu implementieren
  - schneller Lernvorgang, da das Lernen im einfachsten Falle nur aus dem Speichern der Lernstichprobe besteht
  - komplexe Entscheidungsgrenzen im Eingangsraum können mit nur wenigen Lerndatenpunkten induziert werden
  - sehr gute Generalisierungsfähigkeiten für viele reale Probleme
- Nachteile
  - vergleichsweise großer Speicherplatzbedarf
  - vergleichsweise langsamer Prognosemechanismus, da zur Prognose alle gespeicherten Lerndatenpunkte berücksichtigt werden müssen
  - das resultierende Modell ist für einen Menschen schwer zu interpretieren
  - Sensibilität gegenüber irrelevanten oder oder mit Rauschen überlagerten Eingangsgrößen

Viele Forschergruppen haben Erweiterungen des kNN- bzw. der IBL-Algorithmen entwickelt. Wichtige und immer noch aktuelle Fragestellungen betreffen die Reduktion der Größe der Modelle durch Vorverarbeitungsmechanismen zur Auswahl der zu speichernden Datenpunkte [Aha90, WM97b, WM00b], die Verwendung fortschrittlicher Abstandsfunktionen [SW92, WM96, WM97a] bzw. die Ermittlung von – teilweise lokal adaptiven – Gewichtsfaktoren für die einzelnen Eingangsgrößen [WAM97, Aha98] sowie die Verminderung der Sensibilität für irrelevante oder verrauschte Eingangsgrößen durch vorhergehende Eingangsgrößenselektion. Für Erläuterungen über Verfahren zur Eingangsgrößenselektion bzw. zu möglichen Abstandsfunktionen siehe Kapitel 2.6 bzw. 2.8.

Ein Ansatz, der im weiteren Sinne den IBL-Algorithmen zugeordnet werden kann, ist die *Exemplar-Based-Generalization* [Sal91, WD95, Dom97], bei der die Datenpunkte der Lernstichprobe durch Zusammenfassen zu sogenannten Exemplaren oder Prototypen generalisiert werden. Hierbei wird allerdings ein wesentlich aufwendigeres Lernverfahren benötigt, so daß das eigentliche Kennzeichen der IBL-Algorithmen nicht mehr gegeben ist. Siehe hierzu auch die Beschreibungen des NGE- und des RISE-Algorithmus in Kapitel 3.4.

2.4 Weitere Lernverfahren

Weitere Lernverfahren zur datenbasierten Modellierung sind künstliche neuronale Netze und Entscheidungsbäume.

Bei den künstlichen neuronalen Netzen [Sar02] wird versucht, die in der Biologie beobachtete Informationsverarbeitung nachzubilden. Wichtige Ansätze sind das *Multi-Layer-Perceptron* (MLP) und *Radiale-Basis-Funktionen-Netze* (RBF). Der letztere Ansatz ist in Kapitel 4.4 näher beschrieben. Neuronale Netze sind universelle Approximatoren, jedoch ist das mit ihnen gelernte Modell aufgrund seiner Komplexität vergleichsweise schwer zu interpretieren. Im Bereich der Neuro-Fuzzy-Systeme [NK97] wird daher versucht, die Lerneigenschaften der neuronalen Netze mit der Interpretierbarkeit der Fuzzy-Systeme zu verbinden. Dazu wird eine spezielle Netzstruktur gewählt, die sich jederzeit – d.h. vor, während und nach dem Lernen – in ein Fuzzy-System überführen läßt.

Entscheidungsbäume [Qui86] sind aus Knoten und Blättern bestehende Graphen. Ausgehend von einem sogenannten *Wurzelknoten* werden die Knoten des Graphen bis zu einem Blatt, d.h. einem Knoten ohne weitere Nachfolgeknoten, durchlaufen. In jedem Knoten wird dabei aufgrund einer Bedingung über die Auswahl des Nachfolgeknotens entschieden. Eine einfache Bedingung kann beispielsweise die Abfrage sein, ob eine bestimmte Eingangsgröße einen bestimmten Wert überschreitet. Ein Blatt schließlich enthält das Klassifikationsergebnis. Entscheidungsbäume sind jederzeit in einen Regelsatz aus – ggf. ineinander geschachtelten – Wenn-Dann-Sonst-Regeln überführbar. Sie sind besonders geeignet, wenn nur einige wenige sehr relevante Eingangsgrößen vorhanden sind.

2.5 COD-Problematik

Der Begriff der *Curse of Dimensionality*[7] (COD) wurde 1961 von Bellman geprägt [Bel61]. Bellman betrachtete die Aufgabe, Wahrscheinlichkeitsdichten in hochdimensionalen Räumen anhand einer gezogenen Stichprobe zu schätzen. Diese Aufgabe kann als das Problem gesehen werden, die Wahrscheinlichkeitsdichte in jeder Zelle eines mehrdimensionalen Gitternetzes zu bestimmen. Für eine bestimmte, feste Anzahl an Gitterlinien pro Dimension, steigt die Anzahl der Zellen exponentiell mit der Anzahl der Dimensionen an. Damit steigt ebenfalls die Stichprobengröße, die benötigt wird, um die gesuchte Wahrscheinlichkeitsdichte mit einer bestimmten statistischen Sicherheit zu schätzen.

Mittlerweile wird der Begriff der COD-Problematik für alle Probleme verwendet, die bei einem Algorithmus mit zunehmender Anzahl an zu bearbeitenden (Eingangs-)Größen auftreten können. Im Bereich der datenbasierten Modellierung ist für jedes Lernverfahren zu untersuchen, welche Auswirkungen sich bezüglich Laufzeit, Speicherplatzbedarf und Qualität der Ergebnisse bzw. benötigter Stichprobengröße ergeben können. So können bei IBL-Algorithmen und ähnlichen Verfahren Probleme mit der verwendeten Abstandsfunktion dahingehend auftreten, daß sich die ermittelten Abstandswerte für den nächsten und den fernsten Nachbarn einer betrachteten Eingangsraumposition immer mehr einander angleichen und damit letztlich bedeutungslos werden [HAK00]. Beim Fr-Algorithmus hingegen kommt es zu einem mit der Verknüpfungstiefe exponentiell größer werdenden Suchraum. Gleichzeitig steigt aber auch die Anzahl der Prämissenterme, die für keinen einzigen Datenpunkt der Lernstichprobe erfüllt sind. Dieser Zusammenhang wird, wie bereits beschrieben, in [Sla01] ausgenutzt, um den Rechenaufwand durch sicheres Beschneiden des Suchraums zu begrenzen. Bei der Berechnung von Kovarianzmatrizen zu gegebenen Stichproben schließlich, steigt die Anzahl der zu bestimmenden Parameter quadratisch mit der Dimensionalität an. Dies bedeutet, daß die Schätzung dieser Parameter bei konstanter Stichprobengröße zunehmend ungenauer wird und wichtige Eigenschaften der Kovarianzmatrix, wie etwa die Invertierbarkeit, ggf. nicht mehr vorhanden sind [HL96].

[7] Fluch der Dimension oder Leere der hochdimensionalen Räume

2.6 Eingangsgrößenselektion

Nicht immer sind alle gemessenen oder zur Verfügung stehenden Eingangsgrößen notwendig, um die zugrundeliegenden Systemzusammenhänge zu lernen. Eine vorhergehende Selektion der relevanten und nicht redundanten Eingangsgrößen wirkt sich oftmals vorteilhaft aus, da

- der Arbeitsaufwand zum Messen und zum Speichern der Daten reduziert wird,
- das sich ergebende Modell oftmals leichter zu interpretieren ist,
- der Rechenaufwand zum Lernen des Modells geringer ist, da die Laufzeit vieler Lernverfahren mit zunehmender Anzahl an Eingangsgrößen ansteigt und
- viele Lernverfahren sensibel gegenüber irrelevanten oder redundanten Eingangsgrößen sind und die Elimination derartiger Eingangsgrößen einen positiven Effekt auf die erzielbare Generalisierungsfähigkeit haben kann.

Die in der Literatur bekannten Selektionsverfahren lassen sich anhand der verwendeten *Bewertungsfunktion* und der verfolgten *Suchstrategie* einteilen. Die Bewertungsfunktion ist eine geeignet gewählte Funktion, die einer bestimmten Auswahl an Eingangsgrößen, einem sogenannten *Eingangsgrößensatz*, einen Gütewert zuordnet. Die Suchstrategie bestimmt dann die Vorgehensweise, wie im Raum aller möglichen Eingangsgrößensätze eine möglichst gute Lösung gefunden und ausgewählt wird.

Bewertungsfunktion Bezüglich der Wahl der Bewertungsfunktion sind zwei verschiedene Ansätze auszumachen [JKP94, KJ97, KW00]. Zum einen der sogenannte *Filteransatz*. Bei diesem wird ein im allgemeinen einfach und schnell zu berechnendes Maß verwendet. Ein solches Maß ist beispielsweise die Transinformation des Eingangsgrößensatzes bezüglich der Ausgangsgröße. Für eine knappe Beschreibung der Transinformation für den Fall der Betrachtung einer einzigen Eingangsgröße siehe Kapitel 2.8.2. Eine formale Darstellung zur Berechnung der Transinformation im mehrdimensionalen Fall findet sich in [KW00]. Zum anderen gibt es den sogenannten *Wrapperansatz* (engl. *wrapper* = einwickeln), bei dem die mittels Kreuz-Validierung oder ähnlichen Ansätzen innerhalb der Lernstichprobe ermittelte Prognosegüte, die der später zu verwendende Lernalgorithmus bezüglich des jeweiligen Eingangsgrößensatzes erreicht, als Bewertung benutzt wird. Der Vorteil dieses Ansatzes besteht darin, daß der *Bias* der Bewertungsfunktion und der *Bias* des hinterher verwendeten Lernverfahrens, eben aufgrund der Identität von Bewertungsfunktion und Lernverfahren, übereinstimmen. Bei einem Filteransatz hingegen kann nicht gewährleistet werden, daß eine als günstig ausgewählte Lösung sich auch als günstig für das verwendete Lernverfahren erweist. Nachteilig beim Wrapperansatz wirkt sich jedoch der teilweise erheblich größere Rechenaufwand aus.

Suche Zur Suche eines bezüglich der Bewertungsfunktion möglichst guten Eingangsgrößensatzes sind in der Literatur verschiedene Ansätze vorgeschlagen worden. Die sogenannte *vollständige Suche* findet garantiert das globale Optimum, ist aber aufgrund der exponentiell mit der Anzahl der Eingangsgrößen anwachsenden Größe des Suchraums in der Praxis nur begrenzt anwendbar. Meist werden daher sogenannte *Greedy-Algorithmen* oder *Genetische-Algorithmen* verwendet [IV94, DM98]. Bekannte Greedy-Algorithmen sind die sogenannte *Rückwärts-* und *Vorwärtsselektion*, bei denen ausgehend vom Satz, der keine bzw. alle Eingangsgrößen enthält,

iterativ immer diejenige Eingangsgröße hinzugefügt bzw. entfernt wird, deren Hinzufügung bzw. Entfernung den positivsten Effekt[8] auf den Wert der Bewertungsfunktion hat.

In [IV94, DM98] werden vollständige, greedy und genetische Suche für einen Wrapperansatz mit einem IBL-Algorithmus bzw. mit einem Entscheidungsbaum experimentell miteinander verglichen. Oft liefert dabei eine greedy Suche in wesentlich kürzerer Zeit als eine vollständige Suche bereits ausreichend gute Lösungen, die sich in der Bewertung kaum von der optimalen Lösung unterscheiden. Bei sehr komplexen Problemen zeigt sich jedoch, daß eine genetische Suche, um den Preis eines höheren Rechenaufwands, die robustere Suchstrategie sein kann.

2.7 Clusterverfahren

Clusterverfahren [And73, Eve93, HKK97, JMF00] partitionieren eine gegebene Menge von Objekten anhand eines geeignet definierten Ähnlichkeitskriteriums in Gruppen, die als Cluster bezeichnet werden. Ziel dabei ist es, die Ähnlichkeit der Objekte innerhalb einer Gruppe zu maximieren und dabei gleichzeitig die Ähnlichkeit zwischen zwei verschiedenen Gruppen zu minimieren. Harte Clusterverfahren ordnen dabei jedes Objekt exklusiv einem bestimmten Cluster zu, während Fuzzy-Clusterverfahren die graduelle Zugehörigkeit eines Objektes zu mehreren Clustern erlauben. Die zu clusternden Objekte können üblicherweise als Datenpunkte, die aus einem m-dimensionalen Datenvektor $\mathbf{x}$ bestehen, beschrieben werden. Die Menge der zu clusternden Datenpunkte stellt die zu bearbeitende Stichprobe dar.

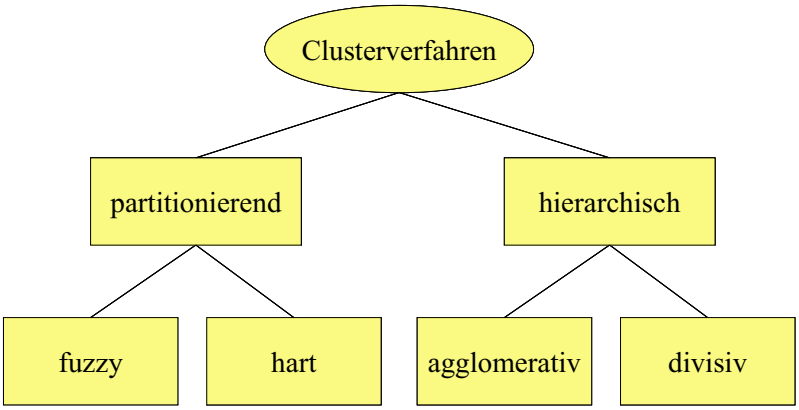


Abbildung 2.5: Verwendete Taxonomie.

Clusterverfahren arbeiten meist *unüberwacht* in dem Sinne, daß eine eventuell vorhandene Ausgangsgröße nicht gesondert berücksichtigt wird. Einige Autoren [Aha95b] definieren Clusterverfahren sogar als unüberwacht arbeitend und ordnen andere Ansätze den maschinellen Lernverfahren zu. Von dieser strengen Sichtweise wird hier jedoch abgesehen und die beiden im Rahmen dieser Arbeit entwickelten Verfahren als Clusterverfahren bezeichnet.

Clusterverfahren lassen sich einteilen in partitionierende und hierarchische Verfahren. Letztere liefern, basierend auf Operationen zum Vereinigen oder zum Teilen von Clustern, eine ganze Serie von Partitionierungen für verschiedene Clusteranzahlen, während erstere nur eine einzige Partitionierung produzieren. Abbildung 2.5 zeigt die hier verwendete Taxonomie. Bevor nun die partitionierenden und hierarchischen Verfahren näher beschrieben werden, soll zunächst ein Eindruck über mögliche Ansätze zur Bewertung der Ähnlichkeit zweier Datenpunkte vermittelt werden.

[8] Dies kann entweder die größtmögliche Verbesserung der Bewertung sein, oder aber auch, wenn sich in jedem Fall eine Verschlechterung ergibt, die kleinstmögliche Verschlechterung.

Ähnlichkeitsmaße Als naheliegendes Ähnlichkeitsmaß kann eine Abstandsfunktion[9] wie beispielsweise der Euklidische Abstand zweier Datenpunkte verwendet werden. Verschiedene Abstandsfunktionen wie die Minkowski-Norm und der allgemeine quadratische Abstand werden in Kapitel 2.8 detailliert beschrieben.

Weitere Ansätze werten beispielsweise für die betrachteten Datenpunkte den Kontext der umliegenden Datenpunkte der zu clusternden Stichprobe aus. Dabei wird eine Abstandsfunktion zur Ermittlung dieses Kontextes benötigt. Beim Jarvis&Patrick-Clusterverfahren [JP73] wird ausgewertet, wieviele nächste Nachbarn zwei Datenpunkte gemeinsam haben. Ähnlich wird bei der sogenannten *Mutual-Neighbor-Distance* (MND)

$$d(\mathbf{x}_a, \mathbf{x}_b) = NN(\mathbf{x}_a, \mathbf{x}_b) + NN(\mathbf{x}_b, \mathbf{x}_a) \tag{2.17}$$

vorgegangen. Dabei besagen die Funktionen $NN(\mathbf{x}_a, \mathbf{x}_b)$ und $NN(\mathbf{x}_b, \mathbf{x}_a)$ der wievielte Nachbar der Datenvektor $\mathbf{x}_a$ bzw. $\mathbf{x}_b$ bezogen auf den Datenvektor $\mathbf{x}_b$ bzw. $\mathbf{x}_a$ ist. Man beachte, daß die MND keine Metrik ist, da die Dreiecksungleichung nicht erfüllt zu sein braucht.

2.7.1 Partitionierende Verfahren

Partitionierende Verfahren optimieren – üblicherweise lokal – die Partitionierung bezüglich einer Gütefunktion. Ein sehr bekanntes und oft eingesetztes einfaches Verfahren ist der k-Means-Algorithmus [McQ67]. Dieser Algorithmus ordnet jeden Datenpunkt exklusiv einem von k generierten Clustern zu. Dabei ist k ein vom Anwender vorzugebender Parameter. Die Ermittlung der Partitionierung erfolgt mittels einer Art iterativer Optimierung. Zunächst werden die k Cluster geeignet initialisiert. Dann wird abwechselnd bis zum Erreichen eines Abbruchkriteriums immer wieder neu die Zuordnung anhand der gegebenen Clusterparameter ermittelt und anschließend die Clusterparameter aufgrund der neuen Zuordnung aktualisiert. Die Abfolge dieser beiden Schritte wird auch als *Lernepoche* bezeichnet. Als Abbruchkriterium kann beispielsweise eine maximale Epochenanzahl oder aber ein Schwellwert für die Veränderung der Clusterparameter zwischen zwei aufeinanderfolgenden Lernepochen verwendet werden. Die Zuordnung der Datenpunkte zu den Clustern kann kompakt in einer sogenannten *Partitionsmatrix* $\mathbf{U}$ dargestellt werden. Ein Element $u_{i,t}$ dieser Matrix enthält die Information über die Zugehörigkeit des i-ten Datenpunktes zum t-ten Cluster. Die Partitionsmatrix hat dementsprechend genausoviele Zeilen wie Datenpunkte in der zu clusternden Stichprobe vorhanden sind und k Spalten. Da der k-Means-Algorithmus eine exklusive Zuordnung vornimmt, können die Elemente der Partitionsmatrix nur die Werte 0 und 1 annehmen. Jedes Cluster wird durch ein sogenanntes *Clusterzentrum* $\mathbf{v}$ gekennzeichnet, dessen einzelne m Komponenten als Schwerpunkt der Datenvektoren der zugeordneten Datenpunkte ermittelt werden zu

$$v_{t\,j} = \frac{\sum_{i=1}^{N} u_{i,t}\; x_{i\,j}}{\sum_{i=1}^{N} u_{i,t}} \; . \tag{2.18}$$

Zu gegebenen Clusterzentren wird die Partitionsmatrix ermittelt, indem jeder Datenpunkt dem nächstliegenden Cluster zugeordnet wird. In Tabelle 2.1 wird der Ablauf, prototypisch für partitionierende Clusterverfahren, zusammengefaßt.

[9] Genaugenommen handelt es sich hierbei um ein *Un*ähnlichkeitsmaß, das passend invertiert werden muß, um als Ähnlichkeitsmaß eingesetzt zu werden.

Tabelle 2.1: Prototypischer Ablauf partitionierender Clusterverfahren.

1.	Initialisiere zufällig die Parameter der k Cluster
2.	Ermittle die Partitionsmatrix zu den gegebenen Clusterparametern
3.	Ermittle die Clusterparameter zu gegebener Partitionsmatrix
4.	Fahre mit dem 2. Schritt bis zum Erreichen eines Abbruchkriteriums fort

Man kann zeigen, daß der Ablauf des k-Means-Algorithmus zur iterativen Minimierung der Zielfunktion

$$J = \sum_{t=1}^{K} \sum_{i=1}^{N} u_{i,t} \; d_{i,t}^2 \qquad \text{mit} \qquad d_{i,t} = ||\mathbf{x}_i - \mathbf{v}_t|| \tag{2.19}$$

führt. Dabei bezeichnet $d_{i,t}$ den aufgrund einer geeigneten Abstandsfunktion ermittelten Abstand des Datenvektors $\mathbf{x}$ zum Zentrum des t-ten Clusters.

Fuzzy-Clusterverfahren Der Fuzzy-C-Means-Algorithmus (FCM) [Bez81] und der Gustafson-Kessel-Algorithmus (GK) [GK79] erweitern den k-Means-Algorithmus für den Fall einer Fuzzy-Partitionierung. Dazu werden die möglichen Werte für die Zuordnung eines Datenpunktes zu einem Cluster von den Werten 0 und 1 auf das Intervall $[0, 1]$ erweitert. Im Rahmen dieser Arbeit wird ausschließlich die sogenannte *probabilistische* Variante, bei der für die Summe der Zuordnungen jedes Datenpunktes i zu den verschiedenen Clustern

$$\sum_{t=1}^{K} u_{i,t} = 1 \tag{2.20}$$

gilt, behandelt.

Der GK-Algorithmus verwendet zusätzlich zum Clusterzentrum sogenannte *Geometrieparameter*, die die Form und Ausdehnung der Datenvektoren der dem Cluster zugeordneten Datenpunkte beschreiben. Dadurch ist der GK-Algorithmus in der Lage, ellipsoide Cluster gleicher Größe zu finden und zu repräsentieren, während der FCM-Algorithmus auf sphärische Cluster beschränkt ist. Der Vollständigkeit halber sei der Gath-Geva-Algorithmus (GG) [GG89] erwähnt, der durch einen weiteren Parameter in der Lage ist, ellipsoide Cluster unterschiedlicher Größe zu finden und zu repräsentieren. In [HKK97] wird im Vergleich der Clusterverfahren FCM, GK, und GG angemerkt, daß die Verfahren mit zunehmend komplexerer Clusterrepräsentation dazu tendieren, während der iterativen Optimierung in einem lokalen Optimum hängen zu bleiben. Daher sind GK- und GG-Algorithmus zwar in der Lage, flexiblere Clusterformen zu identifizieren, jedoch ist die Qualität der gefundenen Partitionierung zunehmend abhängig von einer günstigen Start-Initialisierung. Der genaue Ablauf von FCM- bzw. GK-Algorithmus ist im Rahmen der Beschreibung des hier entwickelten SMBC-Algorithmus in den Kapiteln 4.2.1 bzw. 4.3.1 näher erläutert. So ergibt sich der FCM-Algorithmus aus der Anwendung von Gl. (4.3) und (4.4) zur Ermittlung der Partitionsmatrix und aus Gl. (4.1) zur Ermittlung der Clusterparameter analog zum in Tabelle 2.1 angegeben Ablauf. Der GK-Algorithmus entsteht dann, indem Gl. (4.15) anstelle von (4.3) zur Ermittlung der Abstände von Datenvektoren zu Clustern verwendet wird.

Der FCM-, GK- und GG-Algorithmus können auch im Rahmen der *modell-basierten* Clusterverfahren [FR98] betrachtet werden. Bei diesen Verfahren wird angenommen, daß die zu clusternde Stichprobe anhand einer Überlagerung einzelner Wahrscheinlichkeitsdichtefunktionen gezogen

wurde. Dabei entsprechen diese Wahrscheinlichkeitsdichtefunktionen den Clustern. Die oben für den k-Means-Algorithmus beschriebene alternierende Optimierung der Clusterparameter ist im Rahmen der modell-basierten Clusterverfahren identisch mit einem Expectation-Maximization-Algorithmus (EM).

2.7.2 Hierarchische Verfahren

Die hierarchischen Verfahren lassen sich unterteilen in die sogenannten *agglomerativen* und *divisiven* Verfahren. Bei den agglomerativen Verfahren werden, ausgehend von einer Partitionierung, in der jeder Datenpunkt ein eigenes Cluster darstellt, schrittweise immer zwei, anhand eines Ähnlichkeitskriteriums ausgewählte, Cluster miteinander vereinigt, bis nur noch ein einziges Cluster vorhanden ist oder bis ein Abbruchkriterium erfüllt ist. Bei divisiven Verfahren wird im Gegensatz dazu, ausgehend von einer Partitionierung, in der alle Datenpunkte in einem einzigen Cluster zusammengefaßt sind, schrittweise immer ein Cluster geteilt, bis jedes Cluster nur noch aus einem einzigen zugeordneten Datenpunkt besteht oder bis ein Abbruchkriterium erfüllt ist. Die meisten hierarchischen Verfahren arbeiten agglomerativ, da bei divisiven Verfahren zusätzlich zur Auswahl eines zu teilenden Clusters auch festgelegt werden muß, wie ein Cluster geteilt werden soll, d.h. wie die in dem zu teilenden Cluster enthaltenen Datenpunkte auf die beiden neuen Cluster aufgeteilt werden sollen. Im folgenden werden verschiedene agglomerative Verfahren kurz vorgestellt. Für divisive Verfahren sei auf [SBBG02] verwiesen.

Die beiden bekanntesten agglomerativen Verfahren sind der *Single-Linkage-Algorithmus* [SS73] und der *Complete-Linkage-Algorithmus* [Kin67]. Bei diesen Verfahren werden in jedem Schritt die Cluster für die Vereinigung wie folgt ausgewählt. Für jede mögliche Kombination zweier Cluster C_a und C_b der aktuellen Partitionierung wird der minimale bzw. der maximale Abstand über alle möglichen Paare zweier Datenpunkte aus den beiden betrachteten Clustern zu

$$d(C_a, C_b) = \min_{w \in C_a \; z \in C_b} d(\mathbf{x}_w, \mathbf{x}_z) \qquad \text{bzw.} \qquad d(C_a, C_b) = \max_{w \in C_a \; z \in C_b} d(\mathbf{x}_w, \mathbf{x}_z) \tag{2.21}$$

ermittelt. Dabei indiziert $w \in C_a$ bzw. $z \in C_b$ die Menge aller dem Cluster C_a bzw. C_b zugeordneter Datenpunkte. Ausgewählt wird nun das Clusterpaar C_a und C_b, für das der Abstand $d(C_a, C_b)$ minimal wird. Der Single-Linkage-Algorithmus tendiert dazu, Cluster zu produzieren, deren zugeordnete Datenpunkte im Raum langgezogene Ketten bilden [Nag68], wohingegen der Complete-Linkage-Algorithmus sehr kompakte Cluster bevorzugt [FBY92].

Weitere Varianten sind der *Average-Distance-Algorithmus* [Mur83] und *Ward's-Algorithmus* [War63]. Das erstere Verfahren arbeitet ähnlich wie der Single-Linkage- bzw. Complete-Linkage-Algorithmus, ermittelt jedoch den Abstand zweier Cluster als durchschnittlichen Abstand über alle möglichen Paare zweier Datenpunkte aus den beiden betrachteten Clustern zu

$$d(C_a, C_b) = \frac{1}{|C_a||C_b|} \sum_{w \in C_a} \sum_{z \in C_b} d(\mathbf{x}_w, \mathbf{x}_z) \; . \tag{2.22}$$

Dabei bezeichnet $|C_a|$ bzw. $|C_b|$ die Anzahl der dem jeweiligen Cluster zugeordneten Datenpunkte. Bei letzterem Verfahren wird in jedem Schritt für die Vereinigung das Clusterpaar ausgewählt, dessen Vereinigung zu der geringsten Erhöhung der, über alle Cluster aufsummierten, quadratischen Abstände der Datenpunkte zu einem, analog zum k-Means-Algorithmus nach Gl. (2.18) berechneten, Schwerpunkt führt. Ward's-Algorithmus bevorzugt ebenso wie der k-Means-Algorithmus sphärische Cluster.

2.8 Abstandsfunktionen

Abstandsfunktionen werden zur Bestimmung des Abstandes zweier Datenpunkte P_a und P_b verwendet. Im Rahmen dieser Arbeit werden die Abstände von Datenpunkten zueinander prinzipiell im Eingangsraum ermittelt. Daher werden in den folgenden Beschreibungen die Begriffe *Eingangsvektor* bzw. *Eingangsgröße* verwendet. Für den Einsatz der dargestellten Abstandsfunktionen im Kontext unüberwachter Clusterverfahren sind diese Begriffe durch *Datenvektor* bzw. *Größe* zu ersetzen. Die bekannteste Abstandsfunktion ist die *Minkowski-Metrik* zu

$$d = \sqrt[\rho]{\sum_{j=1}^{m} d_j^{\rho}} \ . \tag{2.23}$$

Für kontinuierliche Eingangsgrößen werden die darin vorkommenden komponentenweisen Abstände d_j zu

$$d_j = |x_{aj} - x_{bj}| \tag{2.24}$$

berechnet. Dabei bezeichnet x_{aj} bzw. x_{bj} die Komponente j des Eingangsvektors $\mathbf{x}_a$ bzw. $\mathbf{x}_b$. Für bestimmte Werte des Parameters ρ entspricht die Minkowski-Metrik den in Tabelle 2.2 angegebenen bekannten Abstandsfunktionen.

Tabelle 2.2: Übliche Wahl des Parameters ρ bei der Minkowski-Metrik.

Komponentenweiser Abstand	$\rho = 1$	$d = \sum_{j=1}^{m} d_j$
Euklidischer Abstand	$\rho = 2$	$d = \sqrt{\sum_{j=1}^{m} d_j^2}$
Chebychev Abstand	$\rho = \infty$	$d = \max_{j=1}^{m} \lvert d_j \rvert$

Üblicherweise wird der Parameterwert $\rho \geq 1$ gewählt. Praktisch sind jedoch auch Werte von $0 < \rho < 1$ möglich. In [AHK01] wird die derart gebildete *Fractional-Distance-Metric* eingeführt und experimentell gezeigt, daß diese bei Verwendung zusammen mit einem kNN-Algorithmus in hochdimensionalen Eingangsräumen günstiger sein kann.

Eine weitere bekannte Abstandsfunktion ist der sogenannte *allgemeine quadratische Abstand* zu

$$d = (\mathbf{x}_a - \mathbf{x}_b)^T \ \mathbf{V} \ (\mathbf{x}_a - \mathbf{x}_b) \ , \tag{2.25}$$

bei dem $\mathbf{V}$ eine problemspezifisch gewählte positiv definite quadratische Matrix der Größe m ist. Wenn $\mathbf{V}$ aus der Kovarianzmatrix

$$\mathbf{Q} = \frac{1}{N-1} \sum_{i=1}^{N} (\mathbf{x}_i - \overline{\mathbf{x}})^T (\mathbf{x}_i - \overline{\mathbf{x}}) \tag{2.26}$$

der N Eingangsvektoren $\mathbf{x}_i$ der betrachteten Stichprobe zu

$$\mathbf{V} = \sqrt[m]{\det \mathbf{Q}}\ \mathbf{Q}^{-1} \tag{2.27}$$

berechnet wird, ergibt sich die sogenannte *Mahalanobis-Distanz*.

Komponentenweise Abstände bei nominalen Eingangsgrößen Die einfachste Art, den komponentenweisen Abstand d_j zweier nominaler Eingangsgrößenwerte x_{aj} und x_{bj} festzulegen, ist die sogenannte *Overlap-Metric*. Bei dieser ist der komponentenweise Abstand entweder 0 bei Gleichheit oder 1 bei Ungleichheit der beiden betrachteten Eingangsgrößenwerte.

$$d_j = \begin{cases} 0 & x_{aj} = x_{bj} \\ 1 & sonst \end{cases} \tag{2.28}$$

Um mit einer feineren Abstufung die Abstände nominaler Eingangsgrößen – auch unter Einbeziehung der Ausgangsgröße – ermitteln zu können, wird in [SW92] die sogenannte *Value-Difference-Metric* (VDM) eingeführt. Die Idee dieses Ansatzes ist es, daß zwei Symbole $w = x_{aj}$ und $z = x_{bj}$ einer nominalen Größe x_j als umso näher zueinander gelten sollen, je mehr sich die bedingten Auftrittswahrscheinlichkeiten $P(y = s|x_j = w)$ und $P(y = s|x_j = z)$ für die verschiedenen Ausgangsklassen s gleichen. Die Vorschrift

$$d_j = \sum_{s=1}^{S_y} |P(y = s|x_j = w) - P(y = s|x_j = z)| = \sum_{s=1}^{S_y} \left| \frac{N_{w,s}}{N_w} - \frac{N_{z,s}}{N_z} \right| \tag{2.29}$$

ist eine vereinfachte Variante der VDM. Die benötigten Auftrittswahrscheinlichkeiten werden mittels der relativen Häufigkeiten bezüglich der Datenpunkte der Lernstichprobe geschätzt. N_w bzw. N_z ist dann die Anzahl aller Datenpunkte, für die die Eingangsgröße x_j das Symbol w bzw. z aufweist und $N_{w,s}$ bzw. $N_{z,s}$ ist dementsprechend die Anzahl aller Datenpunkte, für die zusätzlich noch die Ausgangsgröße das Symbol s aufweist. Dabei wird davon ausgegangen, daß die S_y möglichen Symbole der Ausgangsgröße als von 1 an aufsteigende natürliche Zahlen kodiert sind.

Für mehrere nominale oder gemischt kontinuierliche und nominale Eingangsgrößen können die komponentenweisen Abstände d_j nach Gl. (2.28) oder (2.29) für nominale und nach Gl. (2.24) für kontinuierliche Eingangsgrößen berechnet und dann mittels Gl. (2.23) zum resultierenden Abstand zusammengefaßt werden. Die dadurch entstehenden Abstandsfunktionen werden für den Fall von $\rho = 2$ in [WM97a] auch als *Heterogeneous-Euclidean-Overlap-Metric* (HEOM) und als *Heterogeneous-Value-Difference-Metric* (HVDM) bezeichnet[10].

Fehlende Eingangsgrößenwerte Bei praktischen Lernaufgaben kommt es oft vor, daß für einige Eingangsvektoren Werte fehlen. Ursache hierfür sind z.B. Meßfehler. Aus der Literatur sind zahlreiche Ansätze bekannt, nach denen Lernverfahren in solchen Fällen fehlende Werte verarbeiten können. Beim Regelinduktionsverfahren RISE [Dom97] beispielsweise wird ein fehlender Wert als zusätzlicher symbolischer Wert einer Größe betrachtet. Die Repräsentation der

[10] In neueren Veröffentlichungen [WM97a, WM00a] wird über den erfolgreichen Einsatz der VDM und der daraus entwickelten *Interpolated-Value-Difference-Metric* (IVDM) im Bereich der datenbasierten Modellierung berichtet. Erste Versuche mit den im Rahmen dieser Arbeit entwickelten Clusterverfahren zeigten jedoch, daß die Verwendung der VDM oder IVDM zu fast gleichen Prognosegüten führt, wie eine sinnvoll normierte und gewichtete Metrik nach Gl. (3.15). In Anbetracht des wesentlich höheren Aufwands zur Implementierung wurde daher der Einsatz der VDM und IVDM verworfen und auf weitergehende systematische Untersuchungen verzichtet.

Regeln wird zu diesem Zwecke für jede Eingangsgröße um ein zusätzliches *Flag*, das angibt, ob fehlende Werte abgedeckt sind oder nicht, erweitert. Man beachte, daß diese Vorgehensweise unabhängig vom Typ der Eingangsgröße möglich ist. Eine andere, u.a. beim NGE-Algorithmus [Sal91] verfolgte, Vorgehensweise ist die sogenannte *Ignore Strategie* [Aha90], nach der ein fehlender Wert in einer Abstandsfunktion der Form (2.23) einfach ignoriert wird, indem der entsprechende Einzelabstand $d_j = 0$ gesetzt wird. Der Gesamtabstand wird anschließend durch die Anzahl der bekannten Werte dividiert. Alternativ ist es auch möglich, die Stichprobe vorzubereiten, indem fehlende Werte – beispielsweise durch ein eigens dafür generiertes Modell – geeignet geschätzt werden. In [Qui89, LWTB97] findet sich ein Überblick über verschiedene Strategien, die im Zusammenhang mit der Induktion von Entscheidungsbäumen verwendet wurden.

2.8.1 Normierung

Kontinuierliche Eingangsgrößen Normalerweise wird angestrebt, daß alle Eingangsgrößen in gleicher Weise in den resultierenden Wert der Abstandsfunktion eingehen.[11]. Bei den hier vorgestellten Abstandsfunktionen treten jedoch im Falle kontinuierlicher Eingangsgrößen Probleme auf, wenn die zugehörigen Wertebereiche stark unterschiedlich sind, da dann ggf. die Werte einer einzigen oder weniger Eingangsgrößen den resultierenden Abstandswert im wesentlichen bestimmen können. So z.B. wenn der Wertebereich einer Eingangsgröße im Intervall $[0, 1]$ und der einer anderen im Intervall $[0, 100]$ liegt. Daher sollten die einzelnen Eingangsgrößen vorher mittels Normierungsfaktoren δ_j derart normiert werden, daß die auftretenden Werte der komponentenweisen Abstände meistens innerhalb des Intervals $[0, 1]$ liegen. Für Abstandsfunktionen nach Gl. (2.25) empfiehlt sich eine vorherige Transformation jeder Komponente der Eingangsvektoren der Stichproben zu

$$\widetilde{x}_j = \frac{x_j - x_{min\,j}}{\delta_j} \ . \tag{2.30}$$

Dabei bezeichnet $x_{min\,j}$ den minimalen Wert der Eingangsgröße x_j. Bei Verwendung der Minkowski-Metrik treten die komponentenweisen Abstände explizit auf, weshalb eine Integration der Normierung in die Abstandsfunktion gemäß

$$d = \sqrt[\rho]{\sum_{j=1}^{m} \left(\frac{d_j}{\delta_j} \right)^{\rho}} \tag{2.31}$$

möglich ist. Für kontinuierliche Größen können die Normierungsfaktoren δ_j sehr einfach an die, bezüglich der Lernstichprobe ermittelte, Spannweite bzw. Standardabweichung der jeweiligen Eingangsgröße gekoppelt werden. Eine formale Definition dieser beiden einfachen statistischen Kennzahlen ergibt sich aus Tabelle 2.3. Bei Normierung auf die Spannweite nach

$$\delta_j = x_{range\,j} \tag{2.32}$$

liegen die möglichen komponentenweise Abstände d_j zweier Punkte der Lernstichprobe wie gewünscht im Intervall $[0, 1]$. Die Eingangsgrößen einer Teststichprobe können zwar durchaus andere Extrema aufweisen, jedoch kann in einem solchen Falle ein sich ergebender komponentenweiser Abstand $d_j > 1$ auch als gerechtfertigt angesehen werden. Ein Problem bei Verwendung

[11] Sollten einige Eingangsgrößen als wichtiger eingestuft werden als andere, wird dem, wie im Kapitel 2.8.2 beschrieben, mittels zusätzlicher Gewichtsfaktoren Rechnung getragen.

Tabelle 2.3: Einfache statistische Kennzahlen einer Größe x.

Arithmetischer Mittelwert	$\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$
Standardabweichung	$\sigma = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2$
Maximalwert	$x_{max} = \max_{i=1}^{N} x_i$
Minimalwert	$x_{min} = \min_{i=1}^{N} x_i$
Spannweite	$x_{range} = x_{max} - x_{min}$

der Spannweite ist jedoch, daß diese durch eventuelle Ausreißer, wie sie beispielsweise durch Meßfehler entstehen können, stark gestört sein kann. Bei verrauschten Daten ist es daher sinnvoll, zur Ermittlung der Spannweite einen gewissen Prozentsatz von beispielsweise $\alpha = 5\%$ der kleinsten und größten Werte zu ignorieren, um so robustere Schätzwerte zu erhalten[12]. Bei Normierung mittels der vierfachen Standardabweichung zu

$$\delta_j = 4\sigma_j \tag{2.33}$$

liegen die komponentenweisen Abstände zwar nicht immer, aber doch meist innerhalb des gewünschten Intervals $[0, 1]$. Bei einer normalverteilten Größe x_j liegen 95% aller Werte innerhalb des Intervals $[-2\sigma_j, +2\sigma_j]$, womit Abstände zweier Werte bei Normierung mit 4σ meist kleiner als 1 sein werden.

Gemischt kontinuierliche und nominale Eingangsgrößen Wenn gemischt kontinuierliche und nominale Eingangsgrößen vorhanden sind, stellt die geschickte Normierung von nach Gl. (2.28) ermittelten Abständen ein Problem dar[13]. Zwar liegen die möglichen komponentenweisen Abstände innerhalb des Intervals $[0, 1]$, jedoch ist der sich im arithmetischen Mittel ergebende Wert meist wesentlich größer, als derjenige, der sich für die normalisierten Abstände kontinuierlicher Eingangsgrößen ergibt. Dies führt zu einer überproportionalen Gewichtung nominaler im Vergleich zu kontinuierlichen Eingangsgrößen. Im Rahmen dieser Arbeit werden zwei mögliche Varianten zur gleichmäßigen Normierung kontinuierlicher und nominaler Eingangsgrößen betrachtet.

Die erste und einfachste Vorgehensweise besteht darin, für kontinuierliche Eingangsgrößen weiterhin den Normierungsfaktor nach Gl. (2.32) oder (2.33) an die Spannweite oder Standardabweichung zu koppeln und für nominale Eingangsgrößen einen Normierungsfaktor von $\delta = \delta_{Symb} > 1$ zu wählen.

[12] Dies entspricht der Verwendung des Abstandes des $(1-\alpha)$-Quantils zum α-Quantil der betrachteten Eingangsgröße.

[13] Für Angaben über eine günstige Normierung für nach Gl. (2.29) ermittelte Abstände nominaler Eingangsgrößen siehe [WM97a].

Bei der zweiten Variante werden die Normierungsfaktoren für alle, d.h. für nominale sowie für kontinuierliche Eingangsgrößen nach der in [HC99] beschriebene Normierung auf den durchschnittlichen komponentenweisen Abstand d_{avr} ermittelt. Die Idee dieses Ansatzes besteht darin, die komponentenweisen Abstände jeder Eingangsgröße derart zu normieren, daß sich im arithmetischen Mittel 1 ergibt. Dazu wird anhand der Datenpunkte der jeweils betrachteten Lernstichprobe für jede Eingangsgröße der durchschnittliche komponentenweise Abstand zweier Datenpunkte zueinander bestimmt. In Gl. (2.34) wird der durchschnittliche komponentenweise Abstand der Eingangsgröße x_j als arithmetischer Mittelwert über den komponentenweisen Abstand aller möglichen Kombinationen zweier Eingangsvektoren gemäß

$$\delta_j = d_{avr} = \frac{\sum_{i=1}^{N} \sum_{z=i+1}^{N} d_j(x_{i,j}, x_{z,j})}{\frac{1}{2} N(N-1)} \tag{2.34}$$

ermittelt. Dabei steigt der Rechenaufwand quadratisch mit der Anzahl der Lerndatenpunkte an. Es ist jedoch nicht notwendig, alle möglichen Kombinationen auszuwerten, um einen hinreichend robusten Schätzwert für den durchschnittlichen Abstand zu ermitteln. So könnte zur Begrenzung des Rechenaufwandes nur eine Teilmenge der Datenpunkte betrachtet werden.

2.8.2 Gewichtsfaktoren der Eingangsgrößen

Mittels sogenannter *Gewichtsfaktoren* ist es möglich, den Einfluß besonders relevant erscheinender Eingangsgrößen auf den Wert der Abstandsfunktion zu erhöhen bzw. den Einfluß irrelevant erscheinender Eingangsgrößen zu vermindern. Dabei wird hier streng unterschieden zwischen der oben beschriebenen Normierung und der Gewichtung. Ziel der Normierung ist es, zunächst einen möglichst gleichmäßigen Einfluß aller Eingangsgrößen zu gewährleisten, um dann mittels Gewichtsfaktoren die Relevanz oder Irrelevanz der Eingangsgrößen zu berücksichtigen.

In [WAM97, Aha98] sind zahlreiche Methoden zur Berechnung von Gewichtsfaktoren im Zusammenhang mit dem Einsatz von IBL-Algorithmen beschrieben. Im folgenden wird die auch in [WD95] zusammen mit dem NGE-Algorithmus verwendete Methode mittels der Transinformation [Rez94] der jeweiligen Eingangsgröße zur Ausgangsgröße vorgestellt.

Die Transinformation $I(x, y)$ ist ein Maß dafür, wie stark die *Unsicherheit* über den Wert einer diskreten Zufallsgröße y abnimmt, wenn eine andere diskrete Größe x bekannt ist. Das Maß, mit dem die Unsicherheit gemessen wird, ist die Entropie $E(y)$ [Sha48].

$$E(y) = -\sum_{s=1}^{S_y} P(y = s) \log P(y = s) \tag{2.35}$$

$$I(x, y) = I(y, x) = \sum_{w=1}^{S_x} \sum_{s=1}^{S_y} P(y = s \wedge x = w) \log \frac{P(y = s \wedge x = w)}{P(y = s) P(x = w)} \tag{2.36}$$

Dabei bezeichnen S_x und S_y die Anzahl der möglichen diskreten Ereignisse (Erhebung der Werte) der Zufallsgrößen x bzw. y und $P(x = w)$ und $P(y = s)$ bezeichnen die Auftrittswahrscheinlichkeiten der Ereignisse w und s. Es gilt $\log \frac{0}{0} = 0$. Wenn eine Eingangsgröße nicht dazu beiträgt, die Unsicherheit über die Ausgangsgröße zu verringern, so ist die Transinformation 0. Wenn die Kenntnis einer Eingangsgröße dagegen die Ausgangsgröße vollständig erklärt, so ist die Transinformation gleich der Entropie der Ausgangsgröße.

Die Transinformation ist schnell zu berechnen und die sich ergebenden Werte sind vergleichsweise robust, d.h. es ergeben sich für verschiedene Stichproben ähnliche Zahlenwerte.

Basierend auf der Transinformation werden die Gewichtsfaktoren unter Verwendung der Lernstichprobe wie folgt berechnet: Für jede der m Eingangsgrößen x_j wird die Transinformation $I(x_j, y)$ bezüglich der Ausgangsgröße bestimmt. Kontinuierliche Eingangsgrößen werden für die Berechnung der Transinformation vorher, wie im folgenden Abschnitt beschrieben, durch Vorgabe mehrerer Intervalle diskretisiert. Die erhaltenen Werte der Transinformation werden derart normiert, daß sich im Mittel über alle Komponenten der Wert 1 ergibt.

$$\omega_j = \frac{m\, I(x_j, y)}{\sum_{w=1}^{m} I(x_w, y)} \tag{2.37}$$

Diskretisierung kontinuierlicher Größen Kontinuierliche Größen können durch Dikretisierungsverfahren in ordinale Größen überführt werden, indem der Wertebereich der kontinuierlichen Größe in mehrere Intervalle unterteilt wird. Ein kontinuierlicher Wert wird dann durch die Nummer des Intervalls, in dem er liegt, ersetzt[14]. Die beiden einfachsten und am häufigsten eingesetzten Verfahren sind die sogenannte *äquidistante* und die *äquifrequente* Diskretisierung. Beide Verfahren arbeiten univariat und unüberwacht, d.h. ohne eine evtl. vorhandene Ausgangsgröße zu berücksichtigen. Ferner wird die Vorgabe einer festen Anzahl an Diskretisierungsintervallen N_{Bins} benötigt.

- *Äquidistante Diskretisierung:* Der Wertebereich einer Größe x wird – unter Verwendung der bezüglich der Lernstichprobe ermittelten Spannweite – äquidistant in N_{Bins} Intervalle der Breite

  $$w = \frac{x_{range}}{N_{Bins}} \tag{2.38}$$

  eingeteilt. Ähnlich wie bei der oben beschriebenen Normierung ist es auch hier ratsam, bei Ermittlung der Spannweite einen geringen Prozentsatz der kleinsten und größten Werte zu ignorieren um auch bei evtl. in den Daten enthaltenen Ausreißern ein stabiles Ergebnis zu erhalten. Die Grenzen der beiden äußersten Intervalle werden nach links und rechts ins Unendliche erweitert.

- *Äquifrequente Diskretisierung:* Die gleichmäßige Aufteilung des Wertebereichs einer Größe kann ggf. zu einer in dem Sinne ungünstigen Diskretisierung führen, daß die Anzahl der in jedem Intervall liegenden Werte stark streut. Die äquifrequente Diskretisierung umgeht dieses Problem, indem die Intervallgrenzen so positioniert werden, daß in jedem Intervall jeweils gleich viele Werte

  $$N' = floor\left(\frac{N}{N_{Bins}}\right) \tag{2.39}$$

  liegen. Der Operator $floor(x)$ liefert den ganzzahligen Anteil einer Gleitkommazahl. N ist die Lernstichprobengröße. Wie bei der äquidistanten Diskretisierung werden die Grenzen der beiden äußersten Intervalle nach links und rechts ins Unendliche erweitert.

[14] In dieser Arbeit werden diese Intervalle prinzipiell beginnend mit 1 durchnumeriert.

In der Literatur sind zahlreiche weitere Verfahren zur Diskretisierung einer Größe bekannt. Besonders interessant sind im hier vorliegenden Kontext jene Verfahren, die die Diskretisierung einer Eingangsgröße überwacht, d.h. unter Berücksichtigung der zugehörigen Ausgangsgrößenwerte ermitteln. Im folgenden wird ein in [FI89] erstmalig vorgestelltes auf der Entropie basierendes Verfahren beschrieben. Für einen Überblick über mögliche andere Verfahren siehe [HLTM99, DKS95].

Das Verfahren in [FI89] generiert eine Diskretisierung, die die intervallweise nach Gl. (2.35) berechnete und gemittelte Entropie der Ausgangsgröße y minimiert. Eine Vorgabe der Anzahl der Intervalle ist nicht notwendig. Der Ablauf gestaltet sich wie folgt: Die Eingangsgröße wird derart in zwei Intervalle geteilt, daß die sich bezüglich dieser Diskretisierung ergebende Entropie minimal wird. Anschließend wird rekursiv auf den beiden erhaltenen Intervallen fortgefahren, solange das *Minimum Description Length Principle* (MDLP) nicht verletzt wird. Dieses Prinzip wägt die aus der Unterteilung eines Intervalls resultierende Verringerung der Entropie gegen die zunehmende Komplexität der Diskretisierung ab. Das MDLP wurde ursprünglich im Fachgebiet der Nachrichtentechnik in Zusammenhang mit den Übertragungskosten für eine Botschaft von einem Sender zu einem Empfänger vorgestellt.

Dieses Verfahren scheint zum Einsatz für eine anschließende Berechnung der Transinformation besonders geeignet zu sein. In [HLTM99] wird es in einer Vergleichsstudie verschiedener Verfahren als *first method of choice* bezeichnet.[15]

2.9 Einsatz von Clusterverfahren zur datenbasierten Modellierung

Das Ziel der datenbasierten Regelgenerierung besteht darin, anhand der gegebenen Lernstichprobe zusammenhängende Eingangsraumgebiete, in denen die Ausgangsgrößenwerte möglichst homogen sind, d.h. die gleichen oder ähnliche Werte aufweisen, zu identifizieren. Clusterverfahren können in diesem Kontext auf unterschiedliche Weise genutzt werden. So ist es zum einen möglich, jedes Cluster als eine eigene Regel zu betrachten. Zum anderen kann das Clusterverfahren als eine Art Vorverarbeitungsmechanismus eingesetzt werden, um so für ein nachgeschaltetes Lernverfahren den Rechenaufwand zu reduzieren oder eine günstige Start-Initialisierung vorzugeben.

Einsatz zur direkten Regelgenerierung Im folgenden wird skizziert, wie Clusterverfahren zur direkten Regelgenerierung eingesetzt werden können. Wenn das verwendete Clusterverfahren unüberwacht arbeitet, kann dieses entweder nur im Eingangsraum oder aber im Ein/Ausgangsraum eingesetzt werden. Im letzteren Fall bedeutet dies, daß die Ausgangsgröße als zusätzliche Eingangsgröße behandelt wird. Für einen knappen Überblick siehe [KK97].

Jedes Cluster entspricht direkt einer Regel der Form

$$\textsc{Wenn}\ \textit{Eingangsvektor liegt im Cluster}\ \textsc{Dann}\ y = c\ . \tag{2.40}$$

Für die Ermittlung des Erfülltheitsgrads der Regelprämisse ist, je nach Art des verwendeten Clusterverfahrens, eine geeignete Vorgehensweise zu definieren. Bei partitionierenden Verfahren

[15] Erste experimentelle Versuche mit den im Rahmen dieser Arbeit entwickelten Clusterverfahren verliefen jedoch wenig erfolgversprechend, da die Prognosegüte nicht gesteigert werden konnte. Deshalb wurde auf weitere systematische Untersuchungen verzichtet.

werden die Cluster meist durch Prototypen repräsentiert und der Erfülltheitsgrad der Prämisse kann anhand des Abstandes des Eingangsvektors zu dem Cluster mittels einer geeigneten, beispielsweise gaußförmigen, Kernelfunktion festgelegt werden. Siehe hierzu die Beschreibung zur Überführung der Cluster des hier entwickelten SMBC-Algorithmus in Kapitel 4.2.4. Bei hierarchischen Verfahren hingegen existiert nicht zwangsläufig eine explizite Clusterrepräsentation, sondern nur eine exklusive Zuordnung der einzelnen Datenpunkte zu den Clustern. Anhand dieser kann beispielsweise, wie beim hier entwickelten PNC 2-Algorithmus in Kapitel 3.2.1 beschrieben, der die Eingangsvektoren der zugeordneten Datenpunkte umschließende Hyperquader zur Bildung der Regelprämisse verwendet werden.

Die Konklusion erfordert die Festlegung eines Clusterausgangsgrößenwerts c. Möglich ist beispielsweise, analog zur Ermittlung des Clusterzentrums beim k-Means- oder FCM-Algorithmus, diesen als Schwerpunkt der Ausgangsgrößenwerte der dem Cluster zugeordneten Datenpunkte zu bestimmen. Oder aber der Clusterausgangsgrößenwert ist, wie beim PNC 2-Algorithmus, aufgrund des prinzipiellen Ablaufes des Verfahrens von Anfang an festgelegt.

Einsatz zur Vorverarbeitung Beim Einsatz von Clusterverfahren zur Vorverarbeitung für ein nachgeschaltetes Lernverfahren sind im wesentlichen die drei im folgenden aufgeführten Ansätze zu unterscheiden.

- *Kombination mit Evolutionären- oder Genetischen-Algorithmen:* Das Clusterverfahren wird eingesetzt, um einen ersten Regelsatz zu generieren, der dann mittels Evolutionärer- oder Genetischer-Algorithmen optimiert wird. Insofern dient das Clusterverfahren zur günstigen Start-Initialisierung des nachgeschalteten Optimierverfahrens. Konkrete Beispiele für diesen Ansatz sind in [BGC97, GSJ97, RSA00] zu finden.

- *Kombination mit künstlichen neuronalen Netzen:* Das Clusterverfahren wird, wie bei der Kombination mit Evolutionären- oder Genetischen-Algorithmen, zur Generierung eines ersten Regelsatzes eingesetzt. Dieser wird dann in ein künstliches neuronales Netz überführt. Dabei wird eine spezielle Netzstruktur verwendet, so daß das Netz jederzeit äquivalent zu einem Regelsatz ist [NK97]. Mittels eines Lernverfahrens für künstliche neuronale Netze wird nun der Regelsatz optimiert. Konkrete Arbeiten, die diesen Ansatz verfolgen, sind in [Fri97, RPP99] zu finden.

- *Einsatz zur Erzeugung von TSK-Modellen:* Das Clusterverfahren wird eingesetzt, um die Lernstichprobe zu partitionieren und damit geeignete Eingangsraumgebiete bzw. Regelprämissen, für deren Geltungsbereich dann ein lokales Modell aufgestellt wird, zu identifizieren. Eine einführende Beschreibung dieses Ansatzes ist in [Bab98, Bab02] zu finden. Für ein konkretes Verfahren dieser Art siehe [HK00].

2.10 Ziele der Arbeit

Ziel dieser Arbeit ist es, Clusterverfahren zu entwickeln, die zur direkten datenbasierten Generierung von Regeln der Form (2.40) eingesetzt werden können. Dafür sollen unüberwacht arbeitende Clusterverfahren oder Strategien erweitert oder weiter entwickelt werden, um der von der Clusteranalyse verschiedenen Zielvorgabe, der Identifikation zusammenhängender Eingangsraumgebiete mit möglichst homogenen Ausgangsgrößenwerten, gerecht zu werden. Dabei ist, neben einer hohen Prognosegüte der generierten Regelsätze, den folgenden Randbedingungen Rechnung zu tragen.

Interpretierbare Regeln Um einen möglichst interpretierbaren Regelsatz zu erhalten soll jede einzelne Regel für sich einen sinnvollen Teilaspekt der zu modellierenden Systemzusammenhänge beschreiben, d.h. lokal vernünftig sein. Dafür ist es notwendig, aufgrund von lokalen Entscheidungskriterien über die Bildung eines Clusters zu entscheiden bzw. dessen Bildung zu beeinflussen. Man beachte den Unterschied zu den in Kapitel 2.9 skizzierten Ansätzen, bei denen Clusterverfahren zusammen mit Evolutionären- oder Genetischen-Algorithmen oder neuronalen Netzen, die den generierten Regelsatz unter Verwendung eines globalen Kriteriums optimieren, eingesetzt werden. Eine anschließende Verwendung derartiger Strategien ist zwar prinzipiell möglich, soll hier aber nur am Rande betrachtet werden.

Hochdimensionale Eingangsräume Die entwickelten Clusterverfahren sollen auch in hochdimensionalen Räumen mit 40 und mehr Eingangsgrößen eingesetzt werden können. Dafür ist zu analysieren, welche konkreten Auswirkungen die COD-Problematik auf die entwickelten Clusterverfahren hat. Anschließend sind geeignete Mechanismen zur Umgehung oder Bewältigung der COD-Problematik zu entwickeln.

Vollautomatische Standard-Vorgehensweise Viele Lernverfahren besitzen eine Reihe freier, vom Anwender einzustellender Parameter, mit denen die Strategieelemente des Verfahrens an die jeweilige Lernaufgabe angepaßt werden müssen bzw. können. Die sinnvolle Einstellung dieser Parameter ist meist nur bei einem tiefgreifenden Verständnis der Wirkungsweise des jeweiligen Lernverfahrens möglich. Von großem Nutzen für den Anwender ist daher die Angabe einer Standard-Vorgehensweise, nach der die Parameter des Verfahrens vollautomatisch und ohne tieferes Hintergrundwissen eingestellt werden können.

Kapitel 3

Der PNC 2-Algorithmus

In diesem Kapitel wird das im Rahmen dieser Arbeit entwickelte *Positive and Negative Example-Based Clustering* (PNC 2) beschrieben. Das Kapitel beginnt mit der Darstellung der zugrundeliegenden Idee, beschreibt dann eine vereinfachte Basis-Variante des Algorithmus und präsentiert dessen Pseudo-Code. Darauf folgen Details und Erweiterungen und abschließend eine Einordnung in die bestehende Literatur.

Beim PNC 2-Algorithmus handelt es sich um den Nachfolger des in [Hae01b, Hae01a] erstmalig vorgestellten PNC-Algorithmus. Im wesentlich unterscheidet sich der PNC 2 von seinem Vorgänger darin, daß für nominale Eingangsgrößen eine sinnvolle Verarbeitungsweise realisiert wurde[1], nun eine kontext-sensitive Eingangsgrößenselektion nach Abschluß des Clustervorgangs durchgeführt werden kann, und der Prognosemechanismus für den Fall der Klassifikation verbessert wurde.

3.1 Grundidee

Der PNC 2 ist ein überwacht arbeitendes hierarchisches agglomeratives Clusterverfahren. Derartige Clusterverfahren betrachten zunächst jeden Datenpunkt als *elementares Cluster* und vereinigen dann schrittweise jeweils zwei Cluster miteinander, bis ein Abbruchkriterium erreicht ist oder aber alle Datenpunkte in einem einzigen Cluster vereinigt wurden. Die meisten hierarchischen Verfahren wählen, wie in Kapitel 2.7.2 beschrieben, die jeweils zu vereinigenden Cluster anhand eines im (Eingangs-)Raum ausgewerteten Abstands- oder Punktdichtekriteriums aus. Daher berücksichtigen diese Verfahren zwar die Struktur der Daten im (Eingangs-)Raum, nicht aber, ob die zusammengefaßten Datenpunkte gleiche oder wenigstens ähnliche Ausgangsgrößenwerte haben. Dagegen führt der PNC 2-Algorithmus vor der Vereinigung einen Test durch, der das entstehende generalisierte Cluster nach Regelgesichtspunkten bewertet.

Die Cluster werden durch im Eingangsraum definierte achsparallele *Hyperquader*[2] und einen zugeordneten Ausgangsgrößenwert repräsentiert. Die Hyperquader schließen die Eingangsvektoren aller im Cluster vereinigten Datenpunkte ein. Ein Cluster kann direkt als eine WENN-DANN-Regel interpretiert werden: Der Hyperquader entspricht der Regelprämisse und der zugeordnete Ausgangsgrößenwert bildet die Konklusion.

[1] Beim PNC-Algorithmus wurden nominale Eingangsgrößen ohne weitere Maßnahmen als kontinuierliche Größen behandelt. Unter anderem die deutlich schlechteren Prognosegüten, die der PNC-Algorithmus beim Benchmark DNA erzielte, sind darauf zurückzuführen.

[2] Der Begriff *Hyperquader* paßt eigentlich nur bei kontinuierlichen Eingangsgrößen. Darüber wird an dieser Stelle jedoch hinweggesehen und auf die Darstellung der Repräsentation für nominale Eingangsgrößen in Kapitel 3.3.1 verwiesen.

Der agglomerative Clustervorgang gestaltet sich wie folgt: Zunächst wird jeder Datenpunkte einzeln für sich als elementares Cluster betrachtet. Dabei werden die Ausgangsgrößenwerte dieser elementaren Clusters vom jeweiligen Datenpunkt übernommen. Für eine Vereinigung kommen nur Cluster mit demselben Ausgangsgrößenwert in Betracht. Daher werden die Cluster in Gruppen gleicher Ausgangsgrößenwerte, innerhalb derer dann versucht wird möglichst viele Cluster miteinander zu vereinigen, eingeteilt. Dafür werden immer die beiden einander am nächsten liegenden und bisher noch nicht betrachteten Cluster herausgegriffen und mittels des sogenannten *Vereinigungstests* wird überprüft, ob sie miteinander vereinigt werden dürfen. Dieser Test prüft, ob das durch die Vereinigung entstehende *generalisierte* Cluster eine zutreffende Regel darstellt, die eine bestimmte minimale Trefferquote – ermittelt auf der Lernstichprobe – aufweist. Der Clustervorgang innerhalb einer Gruppe ist beendet, sobald keine nicht getesteten Clusterpaare mehr existieren.

Zur Prognose des Ausgangsgrößenwertes zu gegebener Eingangsraumposition werden die Ausgangsgrößenwerte der im Eingangsraum nächstliegenden Cluster verwendet.

3.2 Basisalgorithmus

Im folgenden wird der vereinfachte aber in niederdimensionalen Eingangsräumen bereits voll funktionsfähige Basisalgorithmus beschrieben. Mehrere Einschränkungen sind vorgenommen worden, um den PNC 2 kompakt und verständlich in seinen wesentlichen Grundzügen darzustellen. So wird zunächst ausschließlich der Fall kontinuierlicher Eingangsgrößen betrachtet und der Aspekt der Normierung und Gewichtung vernachlässigt. Die Darstellung beschränkt sich ferner auf den Fall der Klassifikation mit einem vereinfachten Prognosemechanismus. Die Vorgehensweise für Regressionsaufgaben, spezielle Mechanismen zur Bewältigung der COD-Problematik sowie Details und Erweiterungen sind in Kapitel 3.3 ausführlich beschrieben.

3.2.1 Repräsentation

Cluster Ein Datenpunkt sei, wie in Kapitel 2.1 definiert, durch einen m-dimensionalen Eingangsvektor $\mathbf{x}$ und durch eine zugeordnete Ausgangsklasse y gegeben. Ein Cluster wird durch eine zugeordnete Ausgangsklasse c und durch einen im Eingangsraum definierten Hyperquader H repräsentiert. Der Hyperquader wird durch seine linke untere und rechte obere Ecke $\mathbf{l}$ bzw. $\mathbf{r}$ im m-dimensionalen Eingangsraum beschrieben. Ein Datenpunkt kann in ein elementares Cluster überführt werden, indem die Ausgangsklasse des Datenpunktes übernommen und die Eingangsraumposition des Datenpunktes als linke und rechte Ecke des Hyperquaders verwendet werden. Ein derart gebildeter Hyperquader wird im folgenden auch als *trivialer* Hyperquader bezeichnet.

$$\mathbf{l} = \mathbf{x} \qquad \mathbf{r} = \mathbf{x} \qquad c = y \tag{3.1}$$

Abstandsmaß Der Abstand zweier Cluster C_a und C_b ist der Abstand der beiden Hyperquader H_a und H_b, der als Summe der komponentenweisen Abstände, gemessen von den Außenkanten der Hyperquader aus, gebildet wird. Dies entspricht einer Minkowski-Norm nach Gl. (2.23) mit Wahl des Metrikparameters zu $\rho = 1$. Bezüglich einer einzelnen Eingangsgröße j stellt der Hyperquader ein Intervall dar, welches durch die linke und rechte Grenze l_j und r_j bestimmt ist. Der komponentenweise Abstand zweier Hyperquader d_j bezüglich der Eingangsgröße j ist genau dann 0, wenn sich die beiden Intervalle überlappen. Dementsprechend ist der

gesamte Abstand zweier Hyperquader d genau dann 0, wenn sich die beiden Hyperquader in jeder Eingangsgröße überlappen.

$$d_j = \begin{cases} l_{aj} - r_{bj} & l_{aj} > r_{bj} \\ l_{bj} - r_{aj} & r_{aj} < l_{bj} \\ 0 & \text{sonst} \end{cases} \tag{3.2}$$

Da der Eingangsvektor eines Datenpunktes in einen trivialen Hyperquader überführt werden kann, wird Gl. (3.2) auch verwendet, um den Abstand eines Datenpunktes zu einem Cluster zu ermitteln. Man beachte, daß die Ausgangsgrößenwerte des Datenpunktes und des Clusters irrelevant sind, da die Abstände ausschließlich im Eingangsraum berechnet werden. Ein Abstand von $d = 0$ bedeutet dann, daß der Datenpunkt *innerhalb* des Hyperquaders und damit innerhalb des Clusters liegt: Der Datenpunkt wird von dem Hyperquader und damit von dem Cluster *abgedeckt*.

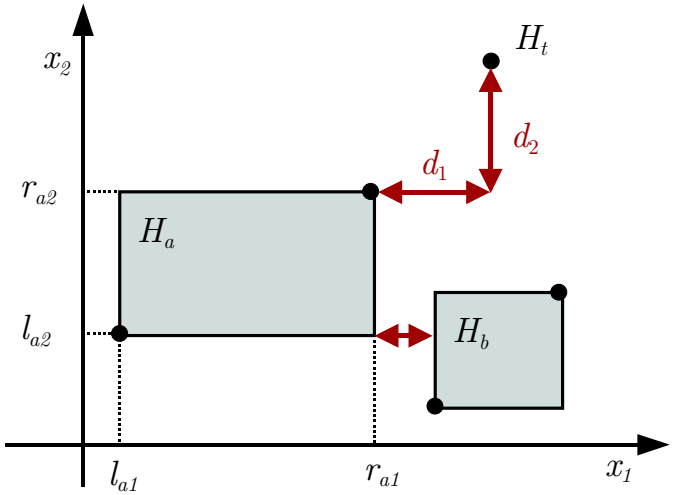


Abbildung 3.1: Repräsentation und Abstände der Hyperquader im zweidimensionalen Eingangsraum.

Überführung in Regeln Ein Cluster $C = (H, c)$ ist äquivalent zu einer WENN-DANN-Regel nach Gl. (3.3) und kann direkt in eine solche überführt werden, indem der vom Hyperquader H abgedeckte Eingangsraumbereich als Regelprämisse und die zugeordnete Ausgangsklasse c als Konklusion verwendet werden.

$$\text{WENN } \mathbf{x} \in H \text{ DANN } y = c \tag{3.3}$$

Ein Eingangsvektor $\mathbf{x}$ liegt innerhalb des Hyperquaders, wenn der Abstand $d(\mathbf{x}, H) = 0$ wird. Dementsprechend wird der Erfülltheitsgrad der Prämisse mittels der ZGF

$$\mu(\mathbf{x} \in H) = \begin{cases} 1 & d(\mathbf{x}, H) = 0 \\ 0 & \textit{sonst} \end{cases} \tag{3.4}$$

ermittelt. Durch Projektion der Zugehörigkeitsfunktion auf die einzelnen Eingangsgrößen ist es möglich, die Regel auf die üblichere Form

$$\text{WENN } \; x_1 = \mu_1 \; \text{ UND } \; x_2 = \mu_2 \; \text{ UND } \; \ldots \; \text{ UND } \; x_m = \mu_m \; \text{ DANN } y = c \tag{3.5}$$

mit

$$\mu_j = \begin{cases} 1 & d_j = 0 \\ 0 & sonst \end{cases} \tag{3.6}$$

zu bringen.

Die hier beschriebenen Regeln verwenden ausschließlich rechteckförmige Zugehörigkeitsfunktionen, d.h. eine Eingangsraumposition ist entweder vollständig inner- oder aber vollständig außerhalb eines Hyperquaders. In Kapitel 3.3.5 wird erläutert, wie die Cluster alternativ auch in Fuzzy-Regeln überführt werden können.

Vereinigung Zwei Cluster mit derselben Ausgangsklasse werden miteinander zu einem generalisierten Cluster vereinigt, indem die Ausgangsklasse übernommen und ein generalisierter Hyperquader H_g aus den beiden Hyperquadern H_a und H_b der zu vereinigenden Cluster wie folgt gebildet wird: Der Hyperquader eines Clusters soll die Eingangsvektoren aller im Cluster vereinigten Datenpunkte erfassen. Dementsprechend werden die neuen linken und rechten Grenzen komponentenweise als Minimum bzw. Maximum der beiden linken und rechten Grenzen der zu generalisierenden Hyperquader gebildet. Dies ist im Rahmen der gewählten Repräsentation die speziellst mögliche Generalisierung.

$$\mathbf{l}_g = \min(\mathbf{l}_a, \mathbf{l}_b) \qquad \mathbf{r}_g = \max(\mathbf{r}_a, \mathbf{r}_b) \qquad c_g = c_a = c_b \tag{3.7}$$

Für jedes Cluster sei die Anzahl der abgedeckten *positiven* und *negativen* Beispiele wie folgt definiert. Es handelt sich dabei um eine, aufgrund der rechteckförmigen ZGF, vereinfachte Betrachtungsweise analog zur Darstellung der Bewertung von Fuzzy-Regeln in Kapitel 2.2.

- Alle Datenpunkte, die innerhalb des Hyperquaders eines Clusters liegen und die denselben Ausgangsgrößenwert wie das Cluster haben, d.h. alle Datenpunkte, für die sowohl die Regelprämisse als auch die Konklusion der äquivalenten Regel gilt, werden als positive Beispiele p eines Clusters betrachtet. Als rechentechnisch einfacher zu aktualisierende, aber ggf. etwas zu pessimistische Schätzung dieser Größe, wird die Anzahl der im Cluster jeweils vereinigten Datenpunkte verwendet. Diese Kenngröße wird im folgenden auch als *Clustermasse* bezeichnet.
- Alle Datenpunkte, die innerhalb des Hyperquaders eines Clusters liegen und einen anderen Ausgangsgrößenwert als das Cluster haben, d.h. alle Datenpunkte für die zwar die Regelprämisse, nicht aber die Konklusion der äquivalenten Regel gilt, werden als negative Beispiele n eines Clusters betrachtet.

3.2.2 Suche

Genereller Ablauf Zur Initialisierung werden zunächst alle Datenpunkte in elementare Cluster überführt. Nur Cluster mit demselben Ausgangsgrößenwert kommen für eine potentielle Vereinigung in Betracht. Daher werden die Cluster anhand der Ausgangsgrößenwerte in Gruppen eingeteilt. In jeder Gruppe wird nun versucht, schrittweise möglichst viele Cluster miteinander zu vereinigen.

Ein einzelner Clusterschritt gestaltet sich wie folgt: Die beiden einander nächstliegenden und bisher noch nicht betrachteten Cluster werden ausgewählt, generalisiert und mittels des sogenannten *Vereinigungstests* wird geprüft, ob das dann entstehende generalisierte Cluster eine

gültige Regel darstellt. Bei bestandenem Test werden die beiden alten Cluster durch das neue, generalisierte Cluster ersetzt.

Die Suche ist beendet, wenn in jeder Gruppe alle möglichen Paare der verbliebenen Cluster negativ getestet worden sind. Um die Clusteranzahl weiter zu reduzieren, werden alle Cluster eliminiert, deren Masse einen Schwellwert p_{min} nicht überschreitet. Dabei ist p_{min} ein vorzugebender Parameter, der im folgenden auch als *minimale Clustermasse* bezeichnet wird. Die Anzahl der Cluster vor bzw. nach der Reduktion wird als K bzw. K_{Red} bezeichnet.

Der doppelte Vereinigungstest Das generalisierte Cluster soll eine möglichst gute Regel darstellen. Als Kriterium für die Güte einer Regel wird die Trefferquote verwendet, deren Schätzwert als das Verhältnis der positiven zu der Gesamtzahl der von der Regel abgedeckten Beispiele zu

$$\varrho = \frac{p}{p+n} \qquad \text{(Gewöhnliche Trefferquote)} \tag{3.8}$$

bzw.

$$\varrho = \frac{1+p}{S_y+p+n} \qquad \text{(Laplace korrigierte Trefferquote)} \tag{3.9}$$

ermittelt wird. Es handelt sich dabei um, aufgrund der rechteckförmigen ZGF nach Gl. (3.6), vereinfachte Versionen der Gl. (2.12) und (2.13).

Die maximale Anzahl der negativen Beispiele n_{max}, die das generalisierte Cluster aufweisen darf, wird aus den Anzahlen der positiven und negativen Beispiele der beiden zu testenden Cluster a und b berechnet zu

$$n_{max} = \min(\, p_a\eta + n_b,\; p_b\eta + n_a \,) \qquad 0 \leq \eta \leq 1 \;. \tag{3.10}$$

Der Parameter η steuert, wieviele negative Beispiele relativ zur Anzahl der positiven Beispiele der Cluster a und b vorhanden sein dürfen. Für $\eta = 0$ ist kein einziges (zusätzliches) negatives Beispiel zulässig. Eine Wahl von $\eta > 0$ erscheint bei verrauschten oder in sich widersprüchlichen Lerndaten sinnvoll.

Die der Gl. (3.10) zugrundeliegenden Überlegungen sollen im folgenden erläutert werden: Man betrachte die beiden Hyperquader der Cluster a und b und den daraus entstandenen generalisierten Hyperquader des Clusters g, die jeweils die Eingangsraumgebiete H_a, H_b und H_g abdecken. Dem generalisierten Cluster äquivalent ist analog zu Gl. (3.3) die Regel

$$\textsc{Wenn}\ \mathbf{x} \in H_g\ \textsc{Dann}\ y = c_g. \tag{3.11}$$

Getestet und bewertet wird jetzt allerdings nicht diese, sondern die beiden folgenden Regeln

$$\textsc{Wenn}\ \mathbf{x} \in H_g\ \textsc{Und}\ \mathbf{x} \notin H_a\ \textsc{Dann}\ y = c_g \tag{3.12}$$

und

$$\textsc{Wenn}\ \mathbf{x} \in H_g\ \textsc{Und}\ \mathbf{x} \notin H_b\ \textsc{Dann}\ y = c_g \;. \tag{3.13}$$

Der Parameter η bedingt einen Schwellwert für die minimale Trefferquote ϱ_{min}, den beide Regeln überschreiten müssen, damit der Vereinigungstest bestanden wird.

$$\varrho_{min} = \frac{1}{1+\eta} \tag{3.14}$$

Diese Vorgehensweise verhindert ein unerwünschtes *Ausufern* von Hyperquadern mit größerer Masse bei Generalisierung mit Hyperquadern kleinerer Masse. Zur Illustration sei das in Abbildung 3.2 skizzierte Szenario betrachtet, bei welchem zwar die dem generalisierten Cluster äquivalente Regel durchaus eine minimale Trefferquote überschreitet, jedoch eine Vereinigung dennoch unerwünscht ist. Dargestellt sind die zwei Hyperquader H_a und H_b der beiden Cluster a und b. Die Massen der Cluster unterscheiden sich stark. Im Eingangsraum liegen zwischen den Hyperquadern der beiden Cluster einige Datenpunkte mit anderer Ausgangsklasse, die sämtlich negative Beispiele des generalisierten Clusters sind. Wenn nun der Parameters η zu 1 gewählt wird, bedingt dies eine minimale Trefferquote von $\varrho_{min} = 50\%$. Das generalisierte Cluster wäre demnach eine gültige Regel, eine Vereinigung ist in diesem Falle jedoch unerwünscht, da – von dem Cluster mit der größeren Masse aus betrachtet – für den von der generalisierten Prämisse zusätzlich abgedeckten Eingangsraumbereich eine Aussage getroffen wird, die augenscheinlich unzutreffend ist.

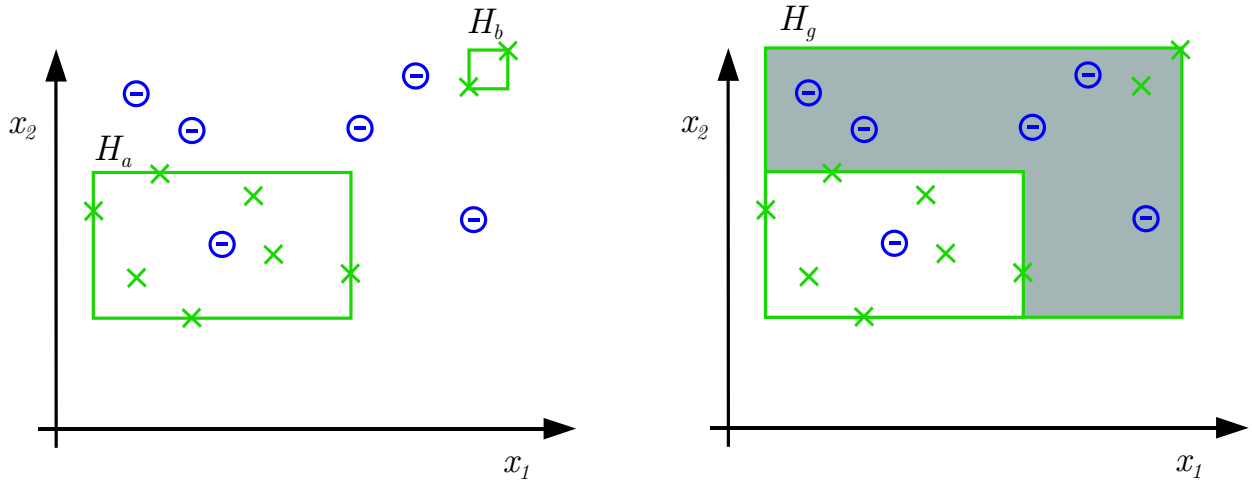


Abbildung 3.2: Illustration der Wirkungsweise des doppelten Vereinigungstests. Rechts ist grau der – vom Hyperquader H_a aus betrachtet – zusätzlich abgedeckte Eingangsraumbereich skizziert.

3.2.3 Prognosemechanismus

Zur Prognose des Ausgangsgrößenwertes zu gegebener Eingangsraumposition wird der Ausgangsgrößenwert des im Eingangsraum nächstliegenden Hyperquaders verwendet. Wenn die Eingangsraumposition innerhalb eines Hyperquaders liegt, so ist ihr Abstand zu diesem $d = 0$ und dementsprechend verhält sich dieser Mechanismus wie ein regelbasiertes System. Falls keine einzige Regel erfüllt sein sollte, d.h. wenn die Eingangsraumposition außerhalb aller Hyperquaders liegt, so verhält sich der Mechanismus wie ein *Nearest-Neighbor-Ansatz* mit dem Unterschied, daß die Abstände nicht zwischen Eingangsraumpositionen und Lerndatenpunkten, sondern zwischen Eingangsraumpositionen und Clustern ermittelt werden. Die Cluster können insofern als *generalisierte* Datenpunkte betrachtet werden.

Der Vollständigkeit halber sei angemerkt, daß der hier beschriebene Prognosemechanismus auch mit einem – wie in Kapitel 3.3.5 beschrieben – aus den Clustern erzeugten Fuzzy-Regelsatz realisiert werden kann, wenn zur Defuzzifizierung die Maximummethode, als UND-Operator das Produkt und als ODER-Operator die gewöhnliche Summe verwendet werden.

3.2.4 Pseudo-Code

In Abbildung 3.3 ist der Ablauf und in Algorithmus 1 der Pseudo-Code des PNC 2-Basisalgorithmus angegeben. Dabei wird davon ausgegangen, daß die möglichen Symbole der Ausgangsgröße als von 1 an durchnumerierte natürliche Zahlen kodiert sind. Der Operator "." bezeichnet dabei das Zugreifen auf eine Variable aus einem Variablenverbund. So wird beispielsweise bei einem durch das Tuple $(\mathbf{l}, \mathbf{r}, c, p, n)$ repräsentierten Cluster a das Element c durch den Ausdruck $a.c$ adressiert.

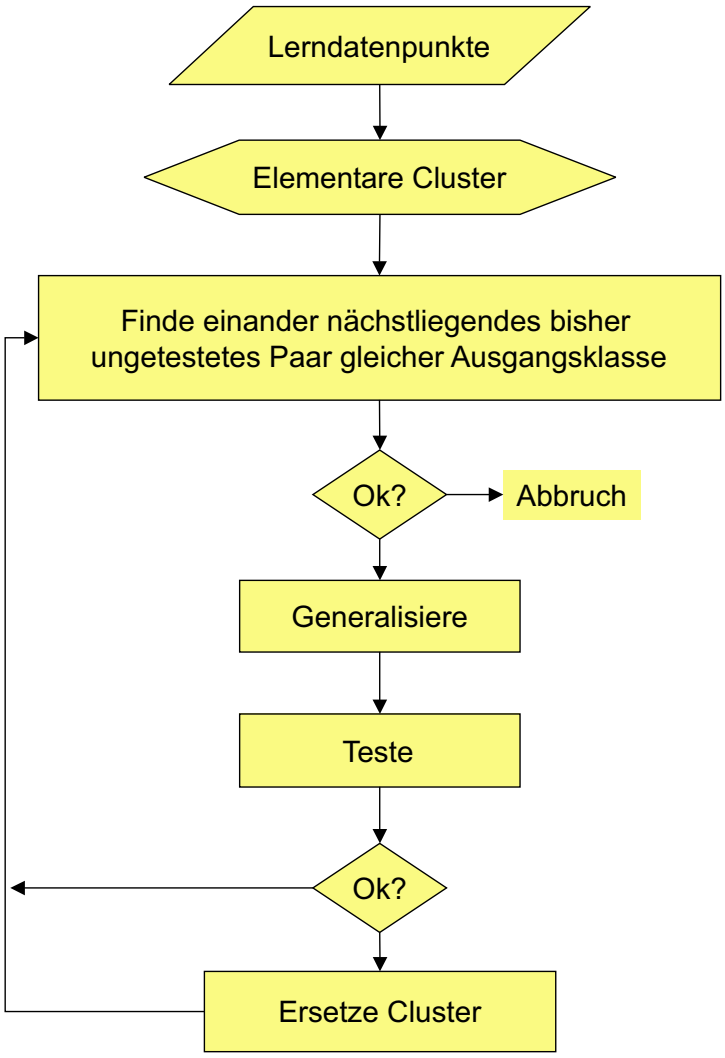


Abbildung 3.3: Prinzipieller Ablauf des PNC 2-Basisalgorithmus.

Algorithm 1: PNC 2($\mathcal{P}$)

comment: Learn clusters $\mathcal{C}$ from given data tuples $\mathcal{P}$. The global p_{min} is the minimum cluster mass and global η is the maximal percentage of negative examples relative to the cluster's mass.

global p_{min}, η

procedure CREATE($\mathbf{x}, y$)
comment: create new elementary cluster a from given data tuple $P = (\mathbf{x}, y)$
$a.\mathbf{l} = \mathbf{x}$
$a.\mathbf{r} = \mathbf{x}$
$a.c = y$
$a.p = 1$
$a.n = 0$
return (a)

procedure GENERALIZE(a, b)
comment: create new cluster g as generalization of the two given clusters a and b
$g.\mathbf{l} = min(a.\mathbf{l}, b.\mathbf{l})$
$g.\mathbf{r} = max(a.\mathbf{r}, b.\mathbf{r})$ **comment:** component wise minima and maxima
$g.p = a.p + b.p$
$g.c = a.c$
return (g)

procedure DISTANCE(a, b)
comment: calculate distance d of clusters a and b to each other
$d = 0$
for each *input* j
do **if** $a.l_j > b.r_j$
then $d = d + a.l_j - b.r_j$
else if $a.r_j < b.l_j$
then $d = d + a.l_j - b.r_j$
return (d)

procedure ISCOVERED($\mathbf{x}, a$)
comment: determine if data tuple's input vector $\mathbf{x}$ is covered by cluster a
b = CREATE($\mathbf{x}, 0$) **comment:** output class is arbitrary
return (DISTANCE(a, b) == 0)

procedure COUNTNEGATIVEEXAMPLES($a, \mathcal{P}$)
comment: count number n of negative examples covered by cluster a
$n = 0$
for each *data tuple* $P_i = (\mathbf{x}_i, y_i)$
do **if** $y_i \neq a.c$ **and** ISCOVERED($\mathbf{x}_i, a$)
then $n = n + 1$
return (n)

procedure MERGETEST(n, a, b)
comment: decide upon the number of positive and negative examples if cluster is a valid rule
$n_{max} = min(a.p\ \eta + b.n, b.p\ \eta + a.n)$
return ($n < n_{max}$)

procedure PREDICT($\mathbf{x}, \mathcal{C}$)
comment: predict output class $\widehat{y}$ given the input vector $\mathbf{x}$
find cluster b *closest to the input vector* $\mathbf{x}$
$\widehat{y} = b.c$
return ($\widehat{y}$)

Algorithm 1: PNC 2($\mathcal{P}$)

main
for each *data tuple* $P_i = (\mathbf{x}_i, y_i)$
do $C_i = \text{CREATE}(\mathbf{x}_i, y_i)$ **comment:** create elementary clusters

for each *output class* s
do { *take the pair of clusters* a *and* b *with output class* s *which are closest to each other and have not been taken before*
continue loop if no pair is found
$g = \text{GENERALIZE}(a, b)$
$g.n = \text{COUNTNEGATIVEEXAMPLES}(g, \mathcal{P})$
if $\text{MERGETEST}(g.n, a, b)$
then *replace the clusters* a *and* b *by* g

for each *cluster* C_t
do { **if** $C_t.p \leq p_{min}$ **comment:** if cluster mass is below threshold
then *delete cluster* t

return ($\mathcal{C}$)

3.3 Details und Erweiterungen

3.3.1 Abstandsfunktion und nominale Eingangsgrößen

Der beschriebene Basisalgorithmus ist nur für kontinuierliche Eingangsgrößen einsetzbar. Für den Fall, daß einige oder alle Eingangsgrößen nominal sein sollten, sind Änderungen an der Repräsentation vorzunehmen. Für die Hyperquader der Cluster wird bei nominalen Eingangsgrößen statt einer linken und rechten Grenze ein *Bitstring* verwendet, der angibt, welche Symbole als vom Hyperquader bezüglich der jeweiligen Eingangsgröße abgedeckt gelten. Der Bitstring hat genauso viele Stellen, wie die nominale Eingangsgröße unterschiedliche Symbole annehmen kann. Jede Stelle korrespondiert mit einem der möglichen Symbole und kann entweder den Wert 0 oder den Wert 1 haben, wobei letzteres bedeutet, daß das entsprechende Symbol vom Bitstring abgedeckt wird. Es wird davon ausgegangen, daß nominale Eingangsgrößen als von 1 an ansteigende natürliche Zahlen kodiert sind. Beispielsweise entsprechen die drei möglichen Symbole a, b und c einer nominalen Eingangsgröße den Werten 1, 2 und 3. Bei Generalisierung zweier Hyperquader wird für eine nominale Eingangsgröße der Bitstring mit einer Art ODER-Operator derart generalisiert, daß ein Symbol als abgedeckt gilt, wenn es von einem der beiden zu generalisierenden Bitstrings abgedeckt ist.

Zur Ermittlung der Abstände zwischen Datenpunkten und Hyperquadern wird analog zu Gl. (2.31) eine normierte und gewichtete heterogene Minkowski-Overlap-Metric zu

$$d = \sqrt[\rho]{\sum_{j=1}^{m} \omega_j \left| \frac{d_j}{\delta_j} \right|^\rho} \tag{3.15}$$

verwendet. Dabei ergeben sich die komponentenweisen Abstände d_j für kontinuierliche Größen nach Formel (3.2) und für nominale Größen zu

$$d_j = \begin{cases} 1 & \text{Symbol } x_j \text{ ist im Bitstring } \mathbf{b}_j \text{ aktiviert} \\ 0 & \text{sonst} \end{cases} . \tag{3.16}$$

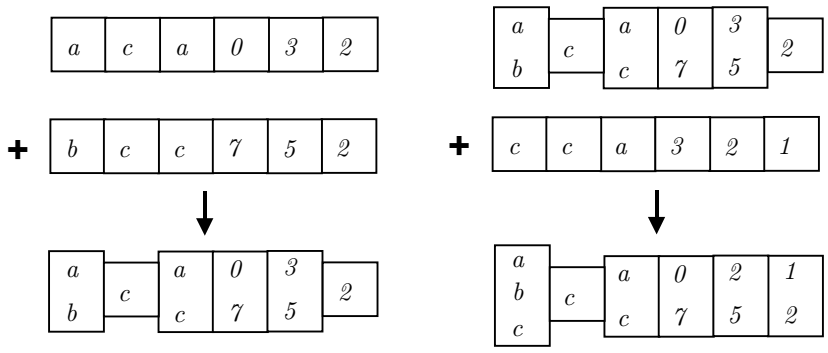


Abbildung 3.4: Zwei Beispiele zur Illustration der symbolischen Repräsentation und Generalisierung. Die ersten 3 Eingangsgrößen sind nominal und können die Symbole a, b und c annehmen, die letzten 3 Eingangsgrößen sind kontinuierlich.

3.3.2 COD-Problematik

Wie in Kapitel 2.5 beschrieben, kann sich die COD-Problematik in unterschiedlicher Art und Weise auf einen Algorithmus auswirken. Dieser Abschnitt erläutert, welcher ungünstige Effekt beim PNC 2-Algorithmus mit zunehmender Anzahl an Eingangsgrößen auftreten und wie dem entgegengewirkt werden kann.

Zunächst sei das *negative Gebiet* definiert als dasjenige Gebiet des Eingangsraumes, welches von der Prämisse einer Regel abgedeckt wird, und in dem die zugrundeliegende wahre Funktion $f_{tf}(\mathbf{x})$ einen anderen Ausgangsgrößenwert annimmt, als die Konklusion der Regel empfiehlt. Sinnvollerweise sollte eine Regel kein negatives Gebiet abdecken. Da $f_{tf}(\mathbf{x})$ jedoch i. allg. nicht bekannt ist, kann obige Forderung nicht exakt erfüllt werden. Stattdessen wird im Vereinigungstest die Anzahl der negativen Beispiele einer Regel betrachtet und davon ausgegangen, daß, wenn eine Regel ein negatives Gebiet abdeckt, diese Regel auch negative Beispiele aufweisen wird. Genau diese Annahme kann aber ein entscheidendes Problem darstellen, wie folgendes Szenario illustriert:

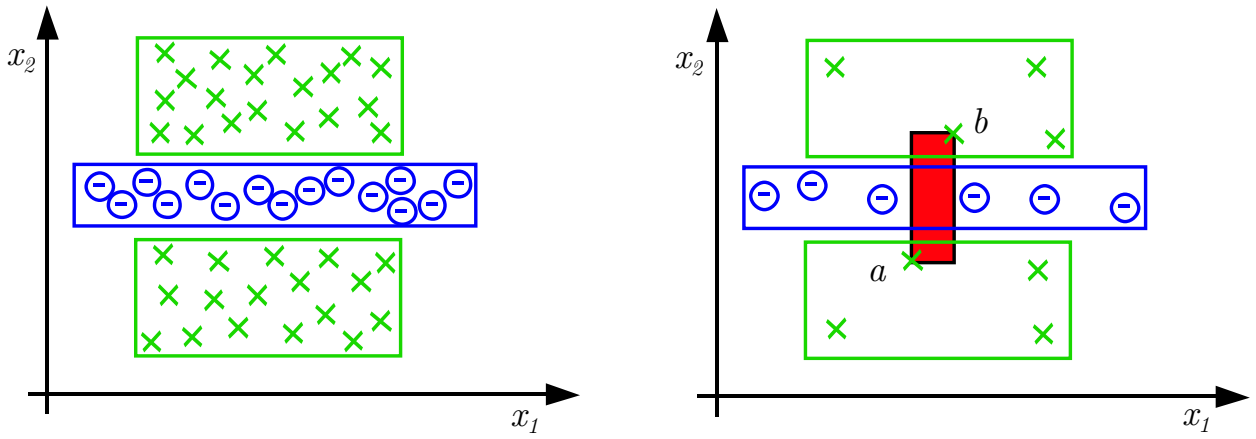


Abbildung 3.5: Illustration zur COD-Problematik.

Es werde das in Abbildung 3.5 skizzierte Problem mit zwei Ausgangsklassen in einem zweidimensionalen Eingangsraum betrachtet. Der wahre zugrunde liegende Zusammenhang ist, wie durch die Quader skizziert, gegeben. Zweimal werden zufällig Lernstichproben mit etwa 50 (links) bzw. etwa 15 (rechts) Datenpunkten generiert. Für beide Lernstichproben soll nun versucht werden, mit dem PNC 2-Algorithmus ein Modell zu lernen. Da $y_{tf}(\mathbf{x})$ bekannt ist, können

die gelernten Cluster danach beurteilt werden, wie gut sie die wahren Zusammenhänge wiedergeben. Im ersten Fall wird dies problemlos möglich sein, bei der kleineren Lernstichprobe wird jedoch das Problem deutlich: Die beiden gekennzeichneten Eingangsvektoren der Datenpunkte a und b liegen einander näher als die Eingangsvektoren anderer Datenpunkte derselben Ausgangsklasse. Daher wird der Versuch unternommen, sie zu generalisieren. Da innerhalb des generalisierten Hyperquaders kein weiterer Datenpunkt liegt, besteht das generalisierte Cluster den Test und deckt damit ein negatives Gebiet ab.

Der Vereinigungstest liefert die gewünschten Ergebnisse nur dann, wenn innerhalb des negativen Gebietes, das eine zu testende generalisierte Regel abdeckt, auch ein Lerndatenpunkt und damit ein negatives Beispiel liegt. Es ist leicht einsichtig, daß die Wahrscheinlichkeit, das ein abgedecktes negatives Gebiet mindestens einen Lerndatenpunkt enthält, von der Anzahl der Lerndatenpunkte und dem Volumen des zu testenden Gebiets derart abhängt, daß diese Wahrscheinlichkeit mit zunehmender Anzahl an Lerndatenpunkten und zunehmendem Volumen des zu testenden Gebiets größer wird.

Die COD-Problematik wirkt sich beim PNC 2-Algorithmus nun derart aus, daß – vorallem in der Anfangsphase des Clustervorgangs – die Volumen der zu testenden Hyperquader mit zunehmender Anzahl an Eingangsgrößen zu klein werden. Folgendes Experiment verdeutlicht diesen Effekt: Betrachtet werden 100 gleichverteilte m-dimensionale Eingangsvektoren. Die Ausgangsgröße ist an dieser Stelle nicht relevant und wird daher auch nicht betrachtet. Es werde nun das durchschnittliche Volumen der aus der Vereinigung von 10 zufällig ausgewählten Eingangsvektoren gebildeten Hyperquader über die Anzahl der Eingangsgrößen aufgetragen.

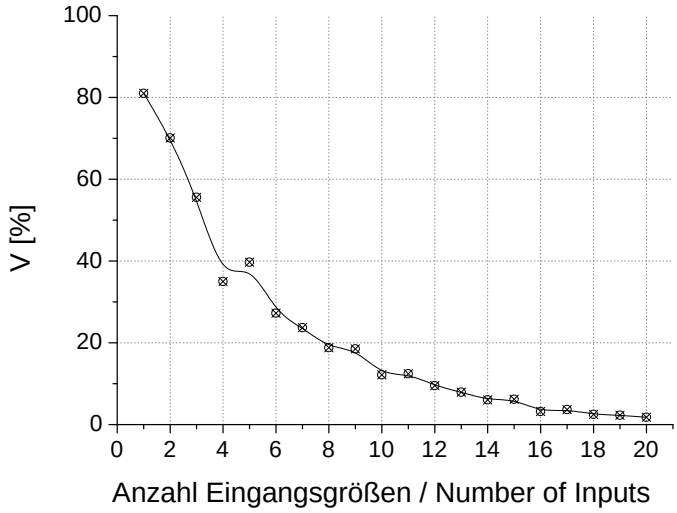


Abbildung 3.6: Volumen der Hyperquader V – normiert auf Volumen des Eingangsraumes.

Wie zu sehen ist, fällt das Volumen annähernd exponentiell ab. Damit steigt also die Wahrscheinlichkeit, daß von einem generalisierten Cluster ein negatives Gebiet unerkannt abgedeckt wird mit zunehmender Anzahl an Eingangsgrößen an. Mit fortschreitendem Clustervorgang werden zwar die Volumen der Hyperquader größer und damit wird dieser Effekt weniger stark auftreten. Dennoch ist nicht davon auszugehen, daß das resultierende Modell eine zufriedenstellende Prognosegüte erzielt, wenn bereits in der Anfangsphase Cluster entstehen, deren äquivalente Regeln negative Gebiete abdecken. Dies läßt sich auch an mehreren höherdimensionalen Benchmarkbeispielen gut beobachten: Die Lerndatenpunkte werden zu sehr wenigen Clustern vereinigt und der äquivalente Regelsatz erzielt nur mäßige bis ausgesprochen schlechte Prognosegüten.

Daher wird folgender Mechanismus eingeführt, um die Vorgehensweise bei der Entscheidung, ob ein Datenpunkt innerhalb eines Hyperquaders liegt, zu modifizieren. Bisher war es ausreichend, wenn ein Datenpunkt bezüglich einer einzigen Eingangsgröße außerhalb lag, um als außerhalb des Hyperquaders zu gelten. Nun wird gefordert, daß die Summe der Gewichtsfaktoren ω_j derjenigen Eingangsgrößen, die außerhalb des Hyperquaders liegen, einen Schwellwert ω_{COD} überschreitet.

$$\sum \omega_j \, sgn(d_j) > \omega_{COD} \tag{3.17}$$

Ansonsten gilt der betreffende Datenpunkt während – und nur während – des Vereinigungstests als innerhalb des Hyperquaders und wird als negatives Beispiel gezählt. Wenn keine Gewichtsfaktoren für die einzelnen Eingangsgrößen verwendet werden sollen, werden alle Eingangsgrößen für den Vereinigungstest mit einem Gewicht von 1 versehen.

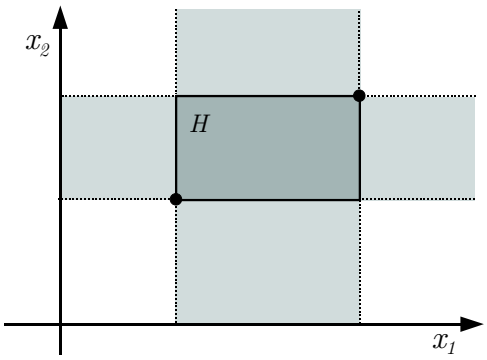


Abbildung 3.7: Extension (grau) der Hyperquader während des Vereinigungstests. Beide Eingangsgrößen haben jeweils einen Gewichtsfaktor von $\omega_1 = \omega_2 = 1$.

Im Abbildung 3.7 ist das von einem generalisierten Hyperquader während des Vereinigungstests bei Wahl des Parameters $\omega_{COD} = 1$ abgedeckte Gebiet für den Fall eines zweidimensionalen Eingangsraumes dargestellt. Wie zu sehen ist, bewirkt der vorgeschlagene Mechanismus eine Vergrößerung der Hyperquader.

3.3.3 Auswahl der zu testenden Cluster

Alle Kombinationen von zwei Clustern mit der gleichen Ausgangsklasse kommen für eine potentielle Generalisierung in Betracht. Dementsprechend werden die Cluster nach ihrer Ausgangsklasse in *Gruppen* eingeteilt, die nacheinander bearbeitet werden. Immer die beiden am nächsten beieinander liegenden und noch nicht negativ getesteten Cluster werden für den Vereinigungstest ausgewählt.

Die Abstände jedes Clusters zu jedem anderen Cluster einer Gruppe werden in einer *Adjazenzmatrix* gespeichert. Adjazenzmatrizen werden beispielsweise in Autoatlanten zur Auflistung der Entfernungen von Städten untereinander verwendet. Es sind Dreiecksmatrizen der Dimension $K-1$, wobei K die aktuelle Anzahl der in der Gruppe enthaltenen Cluster bezeichnet. Die Adjazenzmatrizen werden bei Generalisierung zweier Cluster um eine Dimension verkleinert und bezüglich der Abstände zwischen dem generalisierten und den bestehenden Cluster aktualisiert. Es ist ferner zu beachten, daß ein generalisiertes Cluster wieder mit allen anderen Clustern getestet werden muß. Dies gilt nicht, wenn der Parameter $\eta = 0$ gewählt wird. In diesem Falle braucht das generalisierte Cluster mit keinem Cluster mehr getestet zu werden, mit dem eines der beiden generalisierten Cluster bereits negativ getestet wurde.

Ein Problem stellt die quadratisch mit der Anzahl der Cluster und damit mit der Anzahl der Lerndatenpunkte wachsende Größe der Adjazenzmatrizen dar. Um den PNC 2-Algorithmus dennoch für Aufgaben mit einer großen Anzahl an Datenpunkten verwenden zu können, werden solche Aufgaben in mehrere kleinere Einheiten zerlegt. Diese kleineren Einheiten werden mit wesentlich geringerem Aufwand gelöst und die einzelnen Ergebnisse werden anschließend zu einer Gesamtlösung zusammengesetzt.

Konkret bedeutet dies, daß bei der Initialisierung, d.h. bei der Transformation aller Lerndatenpunkte in Cluster, die maximale Anzahl an Clustern in einer Gruppe auf eine handhabbare Anzahl $N_{G\,Max}$ begrenzt wird. Wenn die Anzahl der Cluster einer Gruppe diese maximal zulässige Anzahl übersteigt, werden die Cluster in zwei oder mehrere etwa gleichgroße *Sub-Gruppen* aufgeteilt. Diese werden einzeln bearbeitet und bilden die einzelnen Lösungen. Um die Gesamtlösung zu erhalten, werden die Cluster der Sub-Gruppen untereinander noch auf potentiell mögliche Vereinigungen getestet.

3.3.4 Kontext-sensitive Eingangsgrößenselektion

Die in Kapitel 2.6 dargestellten Verfahren zur Eingangsgrößenselektion arbeiten global, d.h. eine Eingangsgröße wird entweder als relevant angesehen und verwendet oder aber eliminiert. Teilweise sind Eingangsgrößen jedoch nur innerhalb eines gewissen Kontextes, d.h. innerhalb eines bestimmten Eingangsraumgebietes, relevant. Eine solche Größe dürfte nicht eliminiert werden, da sie teilweise benötigt wird. Motiviert durch einen entsprechenden Ansatz beim, in Kapitel 3.4 beschriebenen, RISE-Algorithmus, wird für den PNC 2 das folgende Verfahren zur kontext-sensitiven Eingangsgrößenselektion (CSFS) definiert.

Nach Abschluß des Clustervorgangs liefert der PNC 2 die Clusterpopulation $\mathcal{C}$. Jedes Cluster C werde nun einzeln für sich betrachtet. Die einzelnen Eingangsgrößen werden anhand des *Abdeckungsgrades* der Hyperquader bezüglich der jeweiligen Eingangsgröße

$$\xi_j = \begin{cases} \frac{|r_j - l_j|}{x_{range\,j}} & \textit{kontinuierliche Eingangsgrössen} \\ \frac{\sum_{w=1}^{S_j} b_{j\,w}}{S_j} & \textit{nominale Eingangsgrössen} \end{cases} \tag{3.18}$$

in absteigender Reihenfolge sortiert. S_j bezeichnet dabei die Anzahl möglicher Symbole einer nominalen Eingangsgröße. Für kontinuierliche Größen wird der vom Hyperquader bezüglich der jeweiligen Eingangsgröße abgedeckte Bereich auf die Spannweite, und damit auf den maximal möglichen Wertebereich, bezogen. Für symbolische Größen wird betrachtet, wieviel Prozent der möglichen Symbole der jeweiligen Eingangsgröße im Bitstring aktiviert sind.

Die einzelnen Hyperquaderkomponenten werden jetzt iterativ in Reihenfolge abnehmender Abdeckungsgrade durchgegangen und es wird versucht, die jeweilige Hyperquaderkomponente maximal zu generalisieren. Dies bedeutet bei nominalen Eingangsgrößen, daß im Bitstring alle Symbole aktiviert werden. Bei kontinuierlichen Eingangsgrößen wird – jeweils getrennt nacheinander – zunächst die linke und dann die rechte Grenze der Hyperquaderkomponente auf $l_j = -\infty$ und $r_j = \infty$ gesetzt. Der Versuch wird akzeptiert, wenn dadurch die Anzahl der vom Cluster abgedeckten negativen Beispiele, die wie während des Vereinigungstests ermittelt werden, nicht erhöht wird. Ansonsten wird der ursprüngliche Zustand der Hyperquaderkomponente wiederhergestellt. Man beachte, daß das Verfahren iterativ arbeitet, d.h. bei Generalisierung einer weiteren Eingangsgröße wird mit dem aus den vorhergehenden Generalisierungsversuchen entstandenen Hyperquader gearbeitet.

Vor Anwendung der CSFS hat jeder Hyperquader die, im Bezug zum benötigten Speicherplatzbedarf angegebene, Größe m. Nach Anwendung der CSFS wird die Größe eines Hyperquaders zu

$$\sum_{j=1}^{m} h_j \tag{3.19}$$

ermittelt, wobei h_j für kontinuierliche Eingangsgrößen zu

$$h_j = \begin{cases} 1 & l_j \neq -\infty \wedge r_j \neq \infty \\ 0 & l_j = -\infty \wedge r_j = \infty \\ 0.5 & sonst \end{cases} \tag{3.20}$$

und für nominale Eingangsgrößen zu

$$h_j = \begin{cases} 0 & \mathbf{b}_j = \mathbf{1} \\ 1 & sonst \end{cases} \tag{3.21}$$

festgelegt ist.

Die Ordnung dieses Verfahrens zur lokalen Eingangsgrößenselektion ist $O(m^2 N_L K)$, wobei N_L die Anzahl der Lerndatenpunkte, m die Anzahl der Eingangsgrößen und K die Anzahl der Cluster bezeichnet. Der Ansatz skaliert linear mit der Anzahl der Cluster, da jedes Cluster unabhängig von den anderen betrachtet wird. Für jedes Cluster muß nacheinander für jede Eingangsgröße die Anzahl der negativen Beispiele bei Generalisierung ermittelt werden. Dies wiederum skaliert linear mit der Anzahl der Eingangsgrößen und der Anzahl der zu berücksichtigen Lerndatenpunkte.

3.3.5 Prognosemechanismus

Klassifikation Der im Basisalgorithmus beschriebene Prognosemechanismus stellt eine Art *1-Nearest-Neighbor* Prognose dar, wobei die gelernten Cluster als *generalisierte* Datenpunkte anzusehen sind. In vielen Untersuchungen ist gezeigt worden, daß ein *k-Nearest-Neighbor* Ansatz bessere Prognosegüten erzielen kann. Bei einem solchen Ansatz werden zur Prognose die k im Eingangsraum nächstliegenden Lerndatenpunkte bestimmt und der bei diesen Punkten am häufigsten vorkommende Ausgangsgrößenwert als Prognosewert verwendet. Dies führt zu einer Glättung der Entscheidungsgrenzen, was z.B. bei verrauschten Daten günstig sein kann.

Für Klassifikationsaufgaben kann ein ähnlicher Mechanismus für den PNC 2 realisiert werden. Jedoch empfiehlt es sich, nicht mit einer feststehenden Anzahl an nächstliegenden Clustern zu arbeiten. Dies wäre aufgrund der teilweise recht starken Kompressionseigenschaften des PNC 2, der die Lerndatenpunkte für einige Lernaufgaben bis auf wenige Cluster zusammenfaßt, unter Umständen fatal, wie folgendes Szenario mit den beiden Ausgangsklassen a und b illustriert. Die Lerndatenpunkte seien zu 5 Clustern zusammengefaßt worden, wobei ein Cluster die Ausgangsklasse a und die restlichen 4 Cluster die Ausgangsklasse b aufweisen. Eine Auswertung der Ausgangsklassen der jeweils 3 nächstliegenden Cluster würde für jede Eingangsraumposition immer zur Prognose der Klasse b führen.

Daher wird der im folgenden beschriebene Prognosemechanismus verwendet, der zu der jeweils gegebenen Eingangsraumposition individuell einen Nachbarschaftsbereich ermittelt: Zunächst wird der Abstand d_{min} des zur Eingangsraumposition $\mathbf{x}$ am nächsten liegenden Clusters bestimmt. Der Nachbarschaftsbereich sei dann festgelegt durch den Nachbarschaftsabstand

$$d_{ref} = w_{Kernel}\, d_{min} + w_{Kernel\,Min}. \tag{3.22}$$

Dabei sind w_{Kernel} und $w_{Kernel\,Min}$ einstellbare Parameter. Eine Wahl zu $w_{Kernel} = 1$ und $w_{Kernel\,Min} = 0$ führt zu einer reinen *1-Nearest-Neighbor* Prognose. Die Einstellung des Parameters $w_{Kernel\,Min}$ wird prozentual an die durchschnittliche Länge der Diagonalen der Hyperquader nach Gl. (3.26) gekoppelt und bedingt eine minimale Größe des Nachbarschaftsbereichs. Alle Cluster mit einem geringeren oder gleichen Abstand zur Eingangsraumposition $\mathbf{x}$ als d_{ref} gelten als innerhalb des Nachbarschaftsbereichs. Die Ausgangsgrößenwerte dieser Cluster werden im Sinne eines gewichteten Mehrheitsentscheides ausgewertet. Dabei wird ein Ausgangsgrößenwert desto stärker gewichtet, umso näher der Abstandswert des zugehörigen Clusters dem Wert des minimalen Abstandes d_{min} ist. Für jedes mögliche Symbol s der Ausgangsgröße wird ermittelt, in welchem Grade

$$\mu_s = \sum_{t=1}^{K} e \left(1 - \frac{d_t - d_{min}}{d_{ref} - d_{min}}\right) \qquad \text{mit} \qquad e = \begin{cases} 1 & d_t < d_{ref} \wedge s = c_t \\ 0 & sonst \end{cases} \tag{3.23}$$

es empfohlen ist[3]. Dabei bezeichnet K die Anzahl der Cluster und d_t bzw. c_t den Abstand bzw. Ausgangsgrößenwert des t-ten Clusters. Damit wird der Einfluß eines Clusters auf den Prognosewert linear mit zunehmendem Abstand geringer. Der Einfluß ist maximal für ein Cluster mit Abstand d_{min} und er wird 0 für ein Cluster mit Abstand d_{ref}, d.h. für ein Cluster an der Grenze des Nachbarschaftsbereichs. Als Ausgangsgrößenwert $\widehat{y}$ wird der am meisten empfohlene Wert nach

$$\widehat{y} = \arg \max_{s=1}^{S_y} (\mu_s) \tag{3.24}$$

prognostiziert.

Regression Der PNC 2-Algorithmus ist problemlos für den Fall der Regression erweiterbar. Dazu wird in einem Vorverarbeitungsschritt die kontinuierliche Ausgangsgröße durch ein geeignetes Diskretisierungsverfahren, wie im Kapitel 2.8.2 beschrieben, in mehrere Intervalle eingeteilt und damit ein Klassifikationsproblem erhalten. Dieses kann dann wie gehabt bearbeitet werden. Im Rahmen dieser Arbeit wird hierzu die äquidistante Diskretisierung verwendet, da diese zum einen ein sehr einfaches Verfahren ist, und zum anderen zu einer gleichmäßigen Aufteilung des Wertebereichs der Ausgangsgröße führt. Um den durch die Diskretisierung der Ausgangsgröße entstandenen Informationsverlust teilweise wieder auszugleichen, werden am Ende des Clustervorgangs die Ausgangsgrößenwerte der Cluster als Schwerpunkt der Ausgangsgrößenwerte der im jeweiligen Cluster vereinigten Datenpunkte neu berechnet.

Die Cluster werden in einen Fuzzy-Regelsatz überführt[4]. Als Fuzzy-UND-Operator wird die Multiplikation und als Fuzzy-ODER-Operator die gewöhnliche Summe verwendet. Die Defuzzifizierung erfolgt nach der COG-Methode analog zu Gleichung (2.8). Die Regeln nehmen dieselbe Form wie im Basisalgorithmus nach Gl. (3.3) an, jedoch wird der Erfülltheitsgrad der Prämisse nun zu

[3] Sollte aufgrund der gewählten Parameter $d_{ref} = d_{min}$ gelten, so wird der dann in Gl. (3.23) auftretende Fall $\frac{0}{0} = 0$ gesetzt.

[4] Um Mißverständnisse zu vermeiden sei darauf hingewiesen, daß die Trefferquote für die Fuzzy-Regeln nicht neu berechnet, sondern von den Clustern übernommen wird. Sämtliche beschriebenen Mechanismen, wie die Reduktion derjenigen Cluster mit zu kleiner Masse oder die kontext-sensitive Eingangsgrößenselektion, finden vorher statt.

$$\mu = \exp - \left(\frac{d}{\sigma_{ZGF}} \right)^{\upsilon} \tag{3.25}$$

ermittelt. Mittels der Parameters υ und σ_{ZGF} wird dabei die Form und die Breite der Flanken der ZGF bestimmt. Übliche Werte für υ sind 1 und 2. Bezüglich des Parameters σ_{ZGF} beachte man, daß es sich um einen global für alle Regeln geltenden Parameter und nicht etwa um eine aus den Datenpunkten eines Cluster berechnete Standardabweichung handelt. Um den Parameter interpretierbar einstellen zu können, ist es sinnvoll, den Wert an den maximal möglichen Abstand im Eingangsraum, oder aber an die wie folgt berechnete durchschnittliche Länge der Diagonalen eines Hyperquaders zu koppeln. Diese Länge wird unter Verwendung der Abstandsfunktion nach Gl. (3.15) als Abstand der linken unteren zur rechten oberen Ecke des Hyperquaders ermittelt. Die einzelnen komponentenweisen Abstände d_j ergeben sich für kontinuierliche Eingangsgrößen als normale Differenz der rechten und linken Ecke und für symbolische Eingangsgrößen als Anteil der abgedeckten zur Gesamtzahl der möglichen Symbole.

$$d_j = \begin{cases} r_j - l_j & \textit{kontinuierliche Eingangsgrössen} \\ \dfrac{\sum_{w=1}^{S_j} b_{w\,j}}{S_j} & \textit{nominale Eingangsgrössen} \end{cases} \tag{3.26}$$

S_j ist die Gesamtzahl der möglichen Symbole der Eingangsgröße j und mit $\sum_{w=1}^{S_j} b_{w\,j}$ wird dann die Anzahl der Symbole bestimmt, die vom Hyperquader bezüglich der betrachteten Eingangsgröße j vom Bitstring $\mathbf{b}_j$ abgedeckt sind.

Wenn der Parameter der Minkowski-Metrik ρ mit υ übereinstimmt ist es ferner möglich, die Zugehörigkeitsfunktion auf die einzelnen Eingangsgrößen zu projizieren und die üblichere Form einer Fuzzy-Regel

$$\textsc{Wenn}\ \ x_1 = \mu_1\ \ \textsc{Und}\ \ x_2 = \mu_2\ \ \textsc{Und}\ \ \ldots\ \ \textsc{Und}\ \ x_m = \mu_m\ \ \textsc{Dann}\ y = c \tag{3.27}$$

mit

$$\mu_j = \exp - \omega_j \left(\frac{d_j}{\delta_j\ \sigma_{ZGF}} \right)^{\rho} \tag{3.28}$$

zu erhalten.

Die Repräsentation der mit dem PNC 2-Algorithmus erhaltenen Regressions-Modelle als Fuzzy-Regeln stellt eine eher ungünstige Art der Repräsentation dar, da für kontinuierliche Eingangsgrößen jede Regel eigene Zugehörigkeitsfunktionen benötigt. Dem kann begegnet werden, indem die Eingangsgrößen vorher in ordinale oder eventuell auch nominale Größen überführt werden. Dies kann entweder durch ein geeignetes Diskretisierungsverfahren automatisch oder aber auch, analog zur üblichen Definition von Zugehörigkeitsfunktionen, wissensbasiert von Hand geschehen.

3.3.6 Skalierbarkeit

Von großer Bedeutung für die Abschätzbarkeit der praktischen Anwendbarkeit eines Lernverfahrens für eine gegebene Lernaufgabe ist die sogenannte *Ordnung*. Diese gibt an, wie sich die Laufzeit bzw. der Speicherplatzbedarf von einem Algorithmus mit zunehmender Größe des zu

bearbeitenden Problems erhöht. Die Größe des Problems ist im Kontext dieser Arbeit gekennzeichnet durch die Anzahl der Eingangsgrößen m sowie durch die Lernstichprobegröße N_L.

Der Vereinigungstest besitzt die Ordnung $O(N_L\, m)$, da jeder Datenpunkt mit anderem Ausgangsgrößenwert betrachtet werden muß und der Aufwand, um festzustellen, ob ein Datenpunkt innerhalb eines Hyperquaders liegt, linear mit der Anzahl der Eingangsgrößen skaliert. Die Anzahl der durchzuführenden Vereinigungstests in jeder Sub-Gruppe gleicher Ausgangsgrößenwerte ergibt sich im *worst case*[5] aus der Kombination jedes Clusters mit jedem anderen, womit sich insgesamt für den Algorithmus die Ordnung $O(N_L^3\, m)$ ergibt. Damit erhöht sich die Laufzeit des PNC 2-Algorithmus im *worst case* kubisch mit der Anzahl der Lerndatenpunkte. Wie die in Kapitel 5.3.4 durchgeführten Experimente jedoch nahelegen, steigt die durchschnittliche Laufzeit an praktisch auftretenden Lernaufgaben nur etwa quadratisch mit der Anzahl der Lerndatenpunkte an.

3.3.7 Pseudo-Code

Im folgenden ist der Pseudo-Code des PNC 2-Algorithmus für gemischt kontinuierliche und nominale Eingangsgrößen sowie der erweiterte Prognosemechanismus für Klassifikationsaufgaben angegeben. Auf die Angabe der Routinen für den Fall der Regression, für den Einsatz einer Trefferquote zur Regelbewertung, für die Aufstellung und effiziente Bearbeitung der Adjazenzmatrizen und für die Eingangsgrößenselektion, ist aus Platzgründen verzichtet worden.

Zur Repräsentation wird bei symbolischen Eingangsgrößen statt der linken unteren und der rechten oberen Ecke ein Bitstring verwendet, der speichert, welche möglichen Symbole abgedeckt sind. Es wird davon ausgegangen, daß die Symbole als von 1 an ansteigende natürliche Zahlen kodiert sind. Zur Abstandsberechnung wird eine gewichtete und normierte Minkowski-Overlap-Metric nach Gl. (3.15) verwendet. Für die Berechnung der dabei benötigten Gewichts- und Normierungsfaktoren der Eingangsgrößen sei auf die Kapitel 2.8.1 und 2.8.2 verwiesen.

Algorithm 2: PNC2($\mathcal{P}$)

comment: The globals $\boldsymbol{\omega}$ and $\boldsymbol{\delta}$ contain feature weights and normalization factors for each input. The global ρ defines the Minkowski metric used and the globals w_{Kernel} and $w_{Kernel\,Min}$ control the neighborhood for the *nearest cluster* prediction. The program **main** and the procedures COUNTNEGATIVEEXAMPLES and MERGETEST are omitted here as they are same as in algorithm 1.

global $\boldsymbol{\omega}, \boldsymbol{\delta}, \rho, w_{Kernel}, w_{Kernel\,Min}, \omega_{COD}$

procedure CREATE(P)
comment: create new elementary cluster a from given data tuple $P = (\mathbf{x}, y)$
for each *input* j
do { **if** *input* j *is symbolic*
then *set bit string* $a.\mathbf{b}_j$ *to cover symbol* x_j
else { $a.l_j = x_j$; $a.r_j = x_j$ }

$a.c = y$
$a.p = 1$
$a.n = 0$
return (a)

[5] Im *worst case* scheitert jeder Vereinigungstest, wodurch die Clusterpopulation am Ende des Clustervorgangs identisch ist mit derjenigen nach der Initialisierung. Das Auftreten dieses Falles ist in der Praxis nahezu auszuschließen.

Algorithm 2: PNC($\mathcal{P}$)

procedure GENERALIZE(a, b)
comment: create new cluster g as generalization of the two given clusters a and b
for each *input* j
do **if** *input* j *is symbolic*
then *set bit string* $g.\mathbf{b}_j$ *to cover all symbols covered by bit strings* $a.\mathbf{b}_j$ *or* $b.\mathbf{b}_j$
else $g.l_j = min(a.l_j, b.l_j)$, $g.r_j = max(a.r_j, b.r_j)$

$g.p = a.p + b.p$
$g.c = a.c$
return (g)

procedure DISTANCE(a, b)
comment: calculate distance d of the two clusters a and b to each other
for each *input* j
do **if** *input* j *is symbolic*
then *set* d_j *to* 1 *if at least one bit is set in* **both** *bit strings* $a.\mathbf{b}_j$ *and* $b.\mathbf{b}_j$
else **if** $a.l_j > b.r_j$ **then** $d_j = d + a.l_j - b.r_j$
else if $a.r_j < b.l_j$ **then** $d_j = d + b.l_j - a.r_j$

$$d = \sqrt[\rho]{\sum \omega_j \left(\frac{d_j}{\delta_j}\right)^{\rho}}$$
return (d)

procedure ISCOVERED($\mathbf{x}, a$)
comment: determine if data tuple's input vector $\mathbf{x}$ is covered by cluster a
$\omega_{sum} = 0$
for each *input* j
do **if** *input* j *is symbolic*
then $\omega_{sum} = \omega_{sum} + \omega_j$ *if symbol* x_j *is* **not** *covered by bit string* $a.\mathbf{b}_j$
else **if** $x_j < a.l$ **or** $x_j > a.r$ **then** $\omega_{sum} = \omega_{sum} + \omega_j$

return ($\omega_{sum} < \omega_{COD}$)

procedure PREDICT($\mathbf{x}, \mathcal{C}$)
comment: predict output class $\widehat{y}$ given the input data vector $\mathbf{x}$
for each *cluster* C_t
do $d_t =$ *distance of input vector* $\mathbf{x}$ *to cluster* C_t
$d_{min} = \min_{t=1}^{K}(d_t)$
$d_{ref} = d_{min}\, w_{Kernel} + w_{Kernel\,Min}$

$\boldsymbol{\mu} = \mathbf{0}$
for each *cluster* C_t
do **if** $d_t < \delta_{ref}$ **then** $s = C_t.c$, $\mu_s = \mu_w + 1 - \frac{d_t - d_{min}}{d_{ref} - d_{min}}$

$\widehat{y} = \arg\max_{s=1}^{S_y}(\mu_s)$ **comment:** predict most suggested output class
return (y)

3.4 Einordnung

In diesem Kapitel wird der PNC 2-Algorithmus mit aus der Literatur bekannten Verfahren und Ansätzen verglichen. Zunächst werden mit dem NGE- und dem RISE-Algorithmus die beiden dem PNC 2 am ähnlichsten Verfahren betrachtet.

Vergleich mit dem NGE-Algorithmus

Der *Nearest-Generalized-Exemplar-Algorithmus* (NGE) [Sal91, Wet94, Aha95a] wurde von SALZBERG entwickelt. Zur Repräsentation werden sogenannte *Exemplare*, bestehend aus einem im Eingangsraum definierten Hyperquader und einer zugeordneten Ausgangsklasse, verwendet. Im wesentlichen wie in Kapitel 3.2 beschrieben, können Datenpunkte in triviale Exemplare überführt, zwei Exemplare generalisiert oder Abstände zwischen Datenpunkten und Exemplaren ermittelt werden. Zur Prognose der Ausgangsklasse zu gegebener Eingangsraumposition wird jeweils die Ausgangsklasse des im Eingangsraum nächstliegenden Exemplars verwendet. Die Repräsentation und der Prognosemechanismus sind damit weitgehend identisch zu denjenigen, welche im PNC 2-Basisalgorithmus verwendet werden.

Der Lernvorgang gestaltet sich dagegen grundlegend anders. Eine vorher festgelegte Anzahl an Lerndatenpunkten wird zufällig ausgewählt und in triviale Exemplare überführt. Diese trivialen Exemplare bilden die Startpopulation. Inkrementell wird jetzt jeweils einer der verbliebenen Lerndatenpunkte ausgewählt, in ein triviales Exemplar E_t überführt und es wird das im Eingangsraum nächstliegende Exemplar E_p aus der aktuellen Population ermittelt. Stimmen die Ausgangsklassen der beiden Exemplare überein, so wird das Exemplar E_g als Generalisierung der beiden Exemplare E_t und E_p erzeugt und in der Population gegen das Exemplar E_p ausgetauscht. Ansonsten wird das zweitnächste Exemplar bestimmt und analog verfahren. Sollten auch bei diesem zweiten Versuch die Ausgangsklassen nicht übereinstimmen, wird das triviale Exemplar E_t in die Population eingefügt. Damit ist ein Lernschritt beendet, der jeweilige Lerndatenpunkt bearbeitet und es wird mit einem weiteren, noch nicht bearbeiteten Lerndatenpunkt, fortgefahren. Das Verfahren ist beendet, wenn alle Lerndatenpunkte bearbeitet wurden.

Den grundlegenden Unterschied im Vergleich zum PNC 2-Algorithmus stellt der inkrementelle Charakter des Lernvorgangs dar, durch den zum einen das Ergebnis abhängig von der Reihenfolge der Datenpunkte ist. Zum anderen wird bei der Generalisierung zweier Exemplare nicht berücksichtigt, ob das hierbei gebildete Exemplar Lerndatenpunkte mit anderer Ausgangsklasse abdeckt.

In [WD95] werden zahlreiche Varianten des ursprünglichen NGE-Algorithmus vorgestellt. Zwei Varianten, die den ursprünglich inkrementellen Algorithmus in eine Batch-Version überführen, sind hier von Interesse: Dies ist zum einen der *Overlapping-Batch-NGE-Algorithmus* (OBNGE) und zum anderen der *Batch-NGE-Algorithmus* (BNGE). Bei diesen Algorithmen wird eine Vereinigung zweier Hyperquader nur dann durchgeführt, wenn von dem dann entstehenden Hyperquader keine negativen Beispiele abgedeckt werden bzw. keine Überlappungen mit bestehenden Hyperquadern auftreten. Damit wird – insbesondere beim OBNGE-Algorithmus – ein im Ansatz ähnlicher Weg wie beim PNC 2-Algorithmus verfolgt. Jedoch fehlen im OBNGE-Algorithmus Mechanismen zur Begegnung der COD-Problematik, so daß die Prognosegüten in höherdimensionalen Räumen ausgesprochen schlecht sind. Wie in Kapitel 5.3.6 experimentell gezeigt wird, ist der PNC 2 bezüglich der Prognosegüte und Größe der generierten Modelle bei allen betrachteten Benchmarkbeispielen jeder NGE-Variante überlegen.

Vergleich mit dem RISE-Algorithmus

Der von DOMINGOS entwickelte RISE-Algorithmus[6] [Dom95, Dom96, Dom97] verwendet dieselbe Repräsentation und einen ähnlichen Prognosemechanismus wie der NGE- bzw. der PNC 2-Algorithmus. Unterschiedlich gestaltet sich jedoch der Lernvorgang: Zur Initialisierung werden alle Lerndatenpunkte – analog zur Vorgehensweise beim PNC 2-Algorithmus – als Regeln betrachtet. Anschließend wird der Regelsatz bis zum Erreichen eines Abbruchkriteriums zyklisch durchgegangen. Dabei werden die Regeln der Reihe nach betrachtet und derart zu generalisieren versucht, daß die generalisierte Regel das nächstliegende positive Lernbeispiel, d.h. den nächstliegenden Lerndatenpunkt mit gleicher Ausgangsklasse, mit abdeckt. Die Generalisierung wird akzeptiert, wenn sie nicht zu einer Verschlechterung der Prognosegüte des Gesamtregelsatzes führt. Die Prognosegüte wird dabei mittels *Leave-One-Out Cross-Validation* bezüglich der Lernstichprobe ermittelt. Da hier keine Datenpunkte miteinander vereinigt werden, sondern vielmehr Regeln unabhängig voneinander derart generalisiert werden, daß sie nahe liegende Datenpunkte mit abdecken, können einander einschließende oder identische Regeln entstehen. Diese müssen detektiert und gelöscht werden.

Der wesentliche Unterschied im Vergleich zum PNC 2-Algorithmus ist in der verwendeten Suchstrategie zu sehen. Der PNC 2 verwendet ein lokales Entscheidungskriterium und fährt fort, solange potentiell zwei Cluster miteinander vereinigt werden können. Insofern kann die Suchstrategie des PNC 2 auch als *global* bezeichnet werden. Der RISE-Algorithmus dagegen verwendet mit der Prognosegüte des aktuellen Regelsatzes ein globales Entscheidungskriterium, jedoch wird nur das jeweils am nächsten zu einer Regel liegende positive Lernbeispiel als Kandidat für eine mögliche Generalisierung der Regel betrachtet, womit die verwendete Suchstrategie als *lokal* bezeichnet werden kann.

Es sind zwei Mechanismen vorgesehen, um die Anzahl der Regeln bzw. die Anzahl der Prämissenterme einer Regel nachträglich zu reduzieren. So werden zum einen alle Regeln gelöscht, die kein einziges positives Lernbeispiel *gewinnen*. Die Regeln werden dazu mit ihrer laplacekorrigierten Trefferquote bewertet. Eine Regel *gewinnt* ein Beispiel, wenn es die dem Beispiel nächstliegende Regel ist. Haben mehrere Regeln den gleichen Abstand – was sehr oft vorkommen kann, wenn sich Regelprämissen überlappen – gewinnt die Regel mit der höchsten Bewertung. Dieser Reduktionsmechanismus führt bei zahlreichen in [Dom97] betrachteten Benchmarkproblemen zu einer Verkleinerung des Regelsatzes um etwa 90%. Allerdings nimmt die Prognosegüte dabei leicht ab. Der andere Ansatz zielt auf eine Reduktion der Prämissenterme ab. Dazu werden bei jeder Prämisse diejenigen Eingangsgrößen gelöscht, für die bei negativen Lernbeispielen keine anderen als die ohnehin abgedeckten Werte auftreten, d.h. für die jedes negative Lernbeispiel bezüglich der betrachteten Eingangsgröße innerhalb des Hyperquaders liegt. Dieser Ansatz ähnelt bezüglich der zugrundeliegenden Idee der kontext-sensitiven Eingangsgrößenselektion beim PNC 2-Algorithmus, erzielt im Gegensatz dazu jedoch laut DOMINGOS keine nennenswerten Kompressionsraten.

In Kapitel 5.3.6 wird der PNC 2 experimentell mit dem RISE-Algorithmus verglichen.

Vergleich mit dem FR-Algorithmus

Der in Kapitel 2.2.1 beschriebene Fuzzy-ROSA-Algorithmus unterscheidet sich bezüglich aller drei Komponenten eines datenbasierten Lernverfahrens, d.h. bezüglich Repräsentation, Suche und Prognosemechanismus, vom PNC 2-Algorithmus. Im folgenden werden diese Unterschiede

[6] Abkürzung für: *Rule Induction from a Set of Exemplars*

erläutert und diskutiert. In Kapitel 5.3.6 wird ergänzend hierzu der PNC 2 experimentell mit dem FR-Algorithmus verglichen.

Repräsentation Der PNC 2-Algorithmus verwendet ausdruckstärkere Prämissen. Der FR-Algorithmus benötigt vorab die Definition von Zugehörigkeitsfunktionen mittels derer kontinuierliche Ein- und Ausgangsgrößen in symbolische Größen überführt werden[7]. Damit sind die Eingangsraumgebiete, die in einer Regelprämisse adressiert werden können, fest vorgegeben. Dies hat andererseits jedoch den Vorteil, daß in allen Regelprämissen dieselben Zugehörigkeitsfunktionen verwendet werden und nicht wie beim PNC 2-Algorithmus, wenn der gelernte PNC 2-Regelsatz in die Form eines Fuzzy-Regelsatzes überführt werden soll, für jede Regelprämisse eigene Zugehörigkeitsfunktionen erzeugt werden müssen. Bei nominalen Eingangsgrößen ermöglicht der PNC 2, wie im folgenden Beispiel illustriert, standardmäßig die ODER-Verknüpfung mehrerer Symbole einer Größe, wohingegen beim FR-Algorithmus in solchen Fällen mehrere Regeln generiert werden müssen.

$$\text{WENN } (x_1 = \textit{gelb} \text{ ODER } x_1 = \textit{rot}) \text{ UND } x_2 = \textit{klein} \text{ DANN } \ldots \tag{3.29}$$

Suche Der FR-Algorithmus wird, vor allem bei Lernaufgaben mit größerer Anzahl an Eingangsgrößen, Regeln geringerer Verknüpfungstiefe generieren als der PNC 2-Algorithmus. Die Ursache hierfür ist die beim FR-Algorithmus verwendete Suchstrategie, welche in einer Art *top-down* Vorgehensweise zunächst Regeln geringer Verknüpfungstiefe, d.h. generalisierende Regeln, sucht. Üblicherweise ist es beim FR-Algorithmus ausreichend, sich auf Regeln mit einer Verknüpfungstiefe von ein bis drei linguistischen Prämissentermen zu beschränken. Der Begriff *ausreichend* wird hier gebraucht, um auszudrücken, daß das Zulassen von Regeln höherer Verknüpfungstiefe meist keine Verbesserung der Prognosegüte zur Folge hat. Auch lassen sich aufgrund der COD-Problematik oft keine Regeln höherer Verknüpfungstiefe finden, da die Größe der adressierten Eingangsraumgebiete mit zunehmender Verknüpfungstiefe exponentiell kleiner wird, und daher dann keine Datenpunkte zur statistischen Absicherung vorhanden sind.

Der PNC 2 hingegen verfolgt einen *bottom-up* Ansatz und fängt mit den speziellsten bildbaren Regeln, also auch mit jeweils voller Verknüpfungstiefe, an. Im Laufe des Lernvorgangs, und vor allem im Zuge der abschließenden kontext-sensitiven Eingangsgrößenselektion, wird durch Generalisierung von Eingangsgrößen die Verknüpfungstiefe reduziert. Eine untere Grenze der Verknüpfungstiefe ist durch den Mechanismus zur Begegnung der COD-Problematik gegeben: Die minimale Summe der Gewichte der verknüpften Eingangsgrößen muß größer sein als der eingestellte Wert des Parameters ω_{COD}. Da dieser Parameter mit zunehmender Anzahl an Eingangsgrößen größer zu wählen ist, kann es hier zu einem Problem kommen, wenn in sehr hochdimensionalen Eingangsräumen die tatsächlichen Zusammenhänge nur die Verknüpfung einiger weniger Eingangsgrößen erfordern.

Aufgrund der gerade erörterten Unterschiede von Repräsentation und Suche ist zu erwarten, daß der *Bias* des FR-Algorithmus vorteilhaft ist, wenn die wahren Zusammenhänge in niederdimensionalen Unterräumen von ein bis drei Eingangsgrößen zu finden sind. Sollte dagegen die Verknüpfung von mehr Eingangsgrößen notwendig sein, sollte der *Bias* des PNC 2-Algorithmus günstiger sein.

[7] Es wird an dieser Stelle vernachlässigt, daß üblicherweise überlappende ZGF verwendet werden, so daß ein kontinuierlicher Wert zu mehr als einer linguistischen Größe gehören kann

Prognosemechanismus Der FR-Algorithmus generiert einen Fuzzy-Regelsatz. Dieser ermöglicht eine Prognose für Eingangsraumpositionen, in denen mindestens eine Regelprämisse aktiviert ist. In allen anderen Bereichen des Eingangsraums ist keine Aussage möglich bzw. es kann lediglich ein vorher eingestellter Default-Wert zur Prognose verwendet werden. Der PNC 2 trifft für alle Bereiche des Eingangsraumes eine Aussage, indem er, falls keine Regel aktiviert ist, den Prognosewert aus den Ausgangsklassen der nächstliegenden Regeln ableitet. Es ist jedoch zu bedenken, daß beim FR-Algorithmus der Fall, daß eine Prognose für eine nicht abgedeckte Eingangsraumposition gemacht werden muß, bei praktischen Lernaufgaben aufgrund der stärker generalisierenden Regeln seltener auftritt.

Reduktionsmethoden Ein weiterer Aspekt ist die Verwendung von Methoden zur nachträglichen Reduktion der Regelanzahl. Beispielsweise enstehen beim FR-Algorithmus redundante Regeln, wenn für ein und denselben Zusammenhang im Ein/Ausgangsraum mehrere Projektionen existieren, in denen sich aus den Lerndatenpunkten gültige Regeln ableiten lassen. Hier kann die *datenbasierte Konfliktreduktion*, bei der alle Regeln in der Reihenfolge absteigender Bewertung durchgegangen und neu bewertet werden, zum Einsatz kommen. Bei dieser Neubewertung wird jeder Lerndatenpunkt mit einem sogenannten *Kreditfaktor* gewichtet, der zu eins initialisiert und dann jedesmal reduziert wird, wenn der zugehörige Datenpunkt von der gerade betrachteten Regel abgedeckt wird. Damit werden Lerndatenpunkte, nachdem sie zum Aufstellen einer oder mehrerer Regel verwendet wurden, allmählich aus der Lernstichprobe ausgeblendet und letztendlich vollkommen entfernt. Eine andere Möglichkeit ist die *optimierende Konfliktreduktion* (OCR), bei der der gesamte Regelsatz mittels eines Evolutionären-Algorithmus' bezüglich der auf der Lernstichprobe erzielten Prognosegüte durch das Ein- und Ausschalten von einzelnen Regeln optimiert wird.

Beim PNC 2-Algorithmus sind zur Zeit keine derartigen nachgeschalteten Reduktionsmechanismen realisiert.

Kapitel 4

Der SMBC-Algorithmus

In diesem Kapitel wird das in [KH02] erstmalig vorgestellte *Supervised Modell Based Clustering* (SMBC) in weiter ausgestalteter und überarbeiterter Form beschrieben. Beginnend mit der Darstellung der zugrundeliegenden Idee wird dann eine vereinfachte Basis-Variante des Algorithmus beschrieben und dessen Pseudo-Code präsentiert. Darauf folgen Details und Erweiterungen und abschließend eine Einordnung in die bestehenden Literatur.

Der im vohergehenden Kapitel vorgestellte PNC 2-Algorithmus folgt dem Paradigma der hierarchischen Clusterverfahren. Mit dem SMBC-Algorithmus wird nun ein auf dem Paradigma der partitionierenden Clusterverfahren basierender Ansatz zur Regelgenerierung vorgestellt.

4.1 Grundidee

Der FCM- und der GK-Algorithmus arbeiten *unüberwacht*, d.h. ohne besondere Berücksichtigung einer Ausgangsgröße. Im Kontext der datenbasierten Regelgenerierung ist es jedoch erwünscht, Cluster zu erhalten, die sich durch möglichst gleiche oder homogene Ausgangsgrößenwerte auszeichnen und es erscheint sinnvoll, dieses bereits während des Clustervorgangs zu berücksichtigen. Ferner ist die Anzahl der zu verwendenden Cluster selten im vorhinein bekannt und sollte sich deshalb während des Clusterprozesses selbständig einstellen. Daher werden der FCM- bzw. GK-Algorithmus durch die neu entwickelten Strategieelemente wie folgt erweitert und das daraus resultierende Verfahren wird fortan als SMBC-Algorithmus bezeichnet.

- Erweiterung der Repräsentation der Cluster um einen Ausgangsgrößenwert
- Modifikation der Zuteilung von Datenpunkten, die bezüglich des Ausgangsgrößenwertes *unpassend* zu einem Cluster sind
- automatische Ermittlung einer passenden Clusteranzahl
  - durch Einfügen neuer Cluster in Gebiete des Eingangsraums, in denen inhomogene Cluster, d.h. Cluster, denen Datenpunkte mit stark unterschiedlichen Ausgangsgrößenwerten zugeteilt wurden, liegen
  - durch Übertragung und Einsatz der Relevanzkonzepte des Fuzzy-ROSA-Algorithmus zur Bewertung von Clustern. Löschen von Clustern mit besonders schlechter Bewertung und Vereinigen von Clustern, wenn dies anhand der resultierenden Relevanzbewertung sinnvoll erscheint.

4.2 Basisalgorithmus

Im folgenden wird die auf dem FCM-Algorithmus basierende Basis-Variante des SMBC-Algorithmus beschrieben. Die Cluster werden dabei zunächst nur durch ein Clusterzentrum repräsentiert. Die Vorgehensweise zur Erweiterung des GK-Algorithmus wird in Kapitel 4.3.1 detailliert beschrieben.

4.2.1 Clusterprototypen und Partitionsmatrix

Ein Cluster wird beschrieben durch ein im Eingangsraum definiertes sogenanntes Clusterzentrum $\mathbf{v}$ und durch einen zugeordneten Ausgangsgrößenwert c. Die Zuteilung jedes Datenpunktes der zu clusternden Stichprobe zu jedem Cluster sei zusammengefaßt in der sogenannten *Partitionsmatrix* $\mathbf{U}$. Das Element $u_{i,t}$ dieser Matrix kennzeichnet die Zuteilung des Datenpunktes P_i zum Cluster C_t. Die Partitionsmatrix hat genauso viele Zeilen wie Datenpunkte in der zu clusternden Stichprobe enthalten sind und genauso viele Spalten wie Cluster vorhanden sind. Für jeden Datenpunkt hat die über alle Cluster gebildete Summe der Zuteilungen den Wert 1. Ordnet man einem Datenpunkt P_i die Masse 1 zu, so läßt sich $u_{i,t}$ auch als der Anteil interpretieren, der dem Cluster C_t von der Masse des Datenpunktes P_i zugeteilt wird.

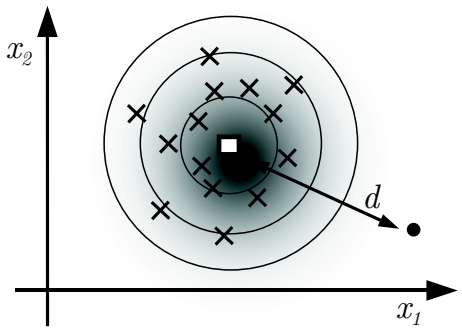


Abbildung 4.1: Zugeteilte Datenpunkte und daraus gebildetes Clusters beim FCM-Algorithmus. Skizziert ist das Clusterzentrum und die Form der aufgrund von Gl. (4.3) resultierenden Äquidistanzlinien im Eingangsraum.

Zu einer Stichprobe werden bei gegebener Partitionsmatrix die Clusterzentren und Ausgangsgrößenwerte jedes Clusters als eine Art Schwerpunkt der zugeteilten Datenpunktmassen gemäß

$$v_{t,j} = \frac{\sum_{i=1}^{N} u_{i,t}^{\alpha} x_{i,j}}{\sum_{i=1}^{N} u_{i,t}^{\alpha}} \tag{4.1}$$

$$c_t = \frac{\sum_{i=1}^{N} u_{i,t}^{\alpha} y_i}{\sum_{i=1}^{N} u_{i,t}^{\alpha}} \tag{4.2}$$

berechnet. Dabei bezeichnet j die jeweilige Eingangsgröße und t das jeweilige Cluster. N ist die Anzahl der Datenpunkte und α ist ein wählbarer *Fuzziness-Parameter*[1], für den eine übliche und auch verwendete Wahl $\alpha = 2$ ist.

Der Abstand eines Datenpunktes zu einem Cluster wird im Eingangsraum gemäß

$$d^2(\mathbf{x}, \mathbf{v}) = \sum_{j=1}^{m} (x_j - v_j)^2 \tag{4.3}$$

als Euklidischer Abstand zwischen dem Eingangsvektor $\mathbf{x}$ des Datenpunktes und dem Zentrum $\mathbf{v}$ des Clusters ermittelt. Zu einer Stichprobe werden bei gegebenen Clustern C_t die Elemente $u_{i,t}$ der Partitionsmatrix nach der Vorschrift

$$u_{i,t} = \frac{1}{\sum_{w=1}^{K} \left(\frac{d_{i,t}}{d_{i,w}} \right)^{\frac{2}{\alpha-1}}} \tag{4.4}$$

ermittelt. Dabei ist $d_{i,t}$ der nach Gl. (4.3) ermittelte Abstand des Datenpunktes $P_i = (\mathbf{x}_i, y_i)$ zum Clusterzentrum $\mathbf{v}_t$ und K ist die Anzahl der Cluster. Die Zuteilung eines Datenpunktes zu einem Cluster ist hiernach um so größer, je geringer der Abstand des Datenpunktes zum betrachteten Cluster – relativ zu den Abständen zu den anderen Clustern – ist.

4.2.2 Modifikationsmatrix

Die Partitionsmatrix wird unüberwacht, d.h. ohne Berücksichtigung der Ausgangsgrößenwerte der jeweiligen Datenpunkte und Cluster ermittelt. Anhand des Ausgangsgrößenwertes eines Clusters ist jedoch feststellbar, wie gut ein Datenpunkt hinsichtlich seines Ausgangsgrößenwertes zu einem Cluster *paßt*. Die Idee des SMBC-Algorithmus besteht darin, diese Information bereits während des Clustervorgangs auszunutzen. Hierzu wird die Zuteilung eines Datenpunktes mittels eines Modifikationsfaktors ω abgeschwächt, wenn sich die Ausgangsgrößenwerte des Datenpunktes und des Clusters stark voneinander unterscheiden. Zur Ermittlung der Modifikationsfaktoren werden im folgenden für den Fall der Regression bzw. der Klassifikation unterschiedliche Vorgehensweisen beschrieben. Analog zur Partitionsmatrix werden die Modifikationsfaktoren in der Matrix $\mathbf{\Omega}$ zusammengefaßt. Das Element $\omega_{i,t}$ gibt an, in welchem Maße die Zuteilung des Datenpunktes P_i zum Cluster C_t abzuschwächen ist.

Anstelle der ursprünglichen Partitionsmatrix wird dann in den Gln. (4.1) und (4.2) zur Bestimmung der Clusterparameter die modifizierte Partitionsmatrix $\mathbf{U}_{mod}$ verwendet, die sich ergibt, indem die ursprüngliche Partitionsmatrix $\mathbf{U}$ elementweise mit der Modifikationsmatrix $\mathbf{\Omega}$ multipliziert wird. Gegebenenfalls ist es sinnvoll, die modifizierte Partitionsmatrix so zu renormieren, daß die Summe der Elemente jeder Zeile wieder den Wert 1 ergibt.

Regression Bei hinreichend gleichmäßiger Verteilung der Ausgangsgrößenwerte der Datenpunkte wird die Ähnlichkeit zweier Ausgangsgrößenwerte anhand eines Gaußkernels nach

$$\omega_{i,t} = \exp - \left(\frac{c_t - y_i}{2\sigma_y \beta} \right)^2 \tag{4.5}$$

[1] In der Literatur wird dieser Parameter meist mit m bezeichnet, was hier aufgrund der Verwechselungsgefahr mit der Anzahl der Eingangsgrößen nicht möglich war.

bewertet. Dabei bezeichnet σ_y die auf der zu clusternden Stichprobe berechnete Standardabweichung der Ausgangsgröße und β ist ein einzustellender Parameter des Verfahrens. Er bestimmt die Breite des Ausgangsgrößenintervalls, das von einem Cluster abgedeckt wird.

Bei stark unterschiedlicher Verteilung der Ausgangsgrößenwerte der Datenpunkte kann es vorteilhaft sein, die Ausgangsgröße in einigen Wertebereichen genauer als in anderen aufzulösen. Dann empfiehlt es sich, den Parameter β – basierend auf der Verteilung der Ausgangsgrößenwerte – für den jeweiligen Ausgangsgrößenwert des Clusters gemäß

$$\omega_{i,t} = \begin{cases} \exp -\left(\dfrac{c_t - y_i}{\beta_{l\,t}}\right)^2 & y < c_t \\ \exp -\left(\dfrac{c_t - y_i}{\beta_{r\,t}}\right)^2 & y \geq c_t \end{cases} \tag{4.6}$$

individuell anzupassen. Die angepaßten Werte für $\beta_{l\,t}$ und $\beta_{r\,t}$ werden durch

$$\beta_{l\,t} = c_t - y_l \qquad \text{und} \qquad \beta_{r\,t} = y_r - c_t \tag{4.7}$$

festgelegt. Dabei werden y_l und y_r so bestimmt, daß innerhalb der Intervalle $[y_l, c_t]$ bzw. $[c_t, y_r]$ jeweils ungefähr N' Datenpunkte enthalten sind. Der Wert des Parameters N' ist über den einzustellenden Parameters γ an die Anzahl der zu clusternden Datenpunkte N und an die aktuelle Clusteranzahl K gemäß

$$N' = \frac{\gamma}{K} N \tag{4.8}$$

gekoppelt.

Klassifikation Im Falle der Klassifikation sind die vorkommenden Ausgangsgrößenwerte der Datenpunkte ganzzahlig und es existiert keinerlei Ordnung der Zahlenwerte wie es beispielsweise bei einer ordinalen Größe der Fall wäre. Daher werden die Modifikationsfaktoren durch

$$\omega_{i,t} = \begin{cases} 1 & y_i = c_t \\ 0 & \text{sonst} \end{cases} \tag{4.9}$$

festgelegt. Dies hat zur Folge, daß sich der Ausgangsgrößenwert eines Clusters nach dessen Initialisierung nicht mehr ändern kann.

4.2.3 Suche

Zu gegebenen Datenpunkten werden die Clusterparameter mit einem iterativen EM-Algorithmus, ähnlich zum prototypischen Ablauf partitionierender Clusterverfahren in Tabelle 2.1, ermittelt.

- *Initialisierung:* Zunächst werden K_{Start} Cluster initialisiert, indem zufällig K_{Start} Datenpunkte der Stichprobe ausgewählt und die Eingangsvektoren und Ausgangsgrößenwerte dieser Datenpunkte als Clusterzentrum und Clusterausgangsgrößenwert verwendet werden.
- *E-Schritt:* Anschließend wird zu den aktuellen Clusterparametern die modifizierte Partitionsmatrix anhand von Gl. (4.4) und einer der Gln. (4.5), (4.6) oder (4.9) berechnet.

- *M-Schritt:* Darauf folgend werden die Clusterparameter zu gegebener Partitionsmatrix anhand von Gl. (4.1) und (4.2) neu ermittelt.
- *Löschen, Hinzufügen und Vereinigen:* Um die Anzahl der Cluster automatisch den Eigenarten der jeweiligen Lernstichprobe anzupassen, werden im Anschluß an den M-Schritt die in den Abschnitten 4.2.6 und 4.2.7 beschriebenen Mechanismen zum Löschen, Hinzufügen und Vereinigen von Clustern eingesetzt.
- *Abbruchbedingung:* Damit sind alle Teilschritte einer Lernepoche beendet und es wird bis zum Erreichen eines geeignet definierten Abbruchkriteriums mit Wiederholung des E-Schritts fortgefahren.

4.2.4 Prognosemechanismus

Wie in Kapitel 2.9 beschrieben ist jedes Cluster C analog zu Gl. (2.40) direkt in eine Fuzzy-Regel

$$\text{Wenn } \mu(\mathbf{x}) \text{ Dann } y = c \tag{4.10}$$

überführbar. Der Aktiviertheitsgrad $\mu(\mathbf{x})$ der Prämisse, d.h. das Maß nach dem bestimmt wird, in welchem Grade ein Eingangsvektor $\mathbf{x}$ innerhalb eines Clusters C liegt, wird durch

$$\mu = \exp(-d^2) \tag{4.11}$$

definiert. Dabei wird d im Basisalgorithmus nach Gl. (4.3) als Abstand des Eingangsvektors zum Clusterzentrum berechnet.

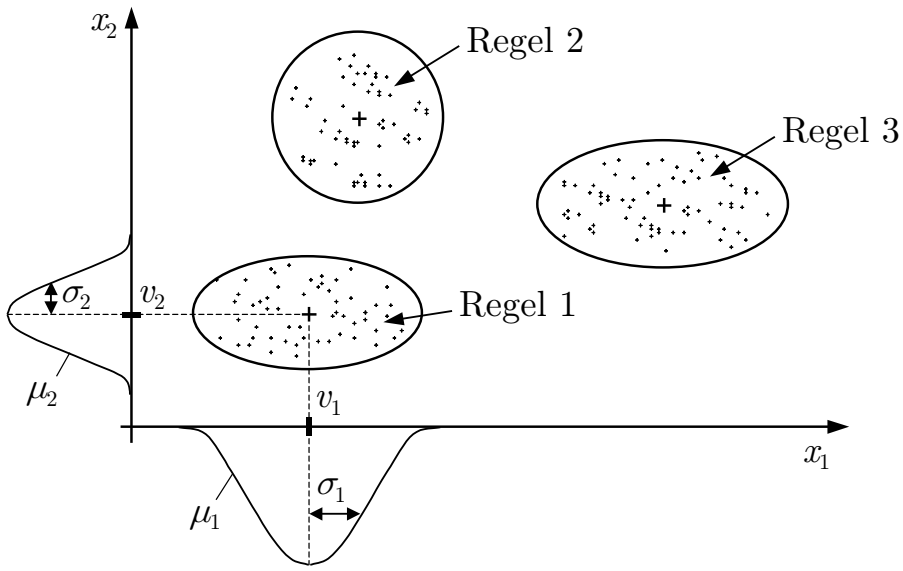


Abbildung 4.2: Projektion der Cluster auf die einzelnen Eingangsgrößenachsen für den Fall, daß, abweichend zum Text, die Cluster zusätzlich, wie in Kapitel 4.3.1 beschrieben, durch diagonalisierte Kovarianzmatrizen beschrieben werden.

Die Überführung von Clustern in Regeln ist in Abbildung 4.2 illustriert. Wenn dies gewünscht ist, kann die Regel durch Projektion der Cluster auf die einzelnen Eingangsgrößenachsen auch auf die folgende, üblichere Form

$$\text{Wenn } x_1 \textit{ ist } \mu_1 \text{ Und} \ldots \text{ Und } x_n \textit{ ist } \mu_n \text{ Dann } y = c \qquad \textit{mit} \qquad \mu_j = \exp(-d_j^2) \tag{4.12}$$

gebracht werden. Wenn zur Verknüpfung mehrerer linguistischer Terme in der Prämisse als Und-Operator die Multiplikation verwendet wird, tritt durch die Projektion für nach Gl. (4.3) berechnete Abstände kein Informationsverlust auf.

4.2.5 Relevanzbewertung

Im Rahmen des Fr-Algorithmus sind zur Bewertung der Relevanz einer Regel mehrere statistisch motivierte Kriterien entwickelt worden (Siehe Kapitel 2.2). Der Grundgedanke besteht darin, jede einzelne Regel danach zu bewerten, ob sie einen signifikanten Zusammenhang des Ein/Ausgangsverhaltens beschreibt. Die Übertragung des Relevanzkonzeptes auf eine Regel, die wie beschrieben einem Cluster zugeordnet wird, erfolgt derart, daß die Homogenität der Ausgangsgrößenwerte der Datenpunkte innerhalb eines Clusters betrachtet wird. Gemessen wird der Homogenitätsgrad als Trefferquote

$$\varrho_t = \frac{\sum_{i=1}^{N} \omega_{i,t}\mu_{i,t}}{\sum_{i=1}^{N} \mu_{i,t}} \,, \tag{4.13}$$

die sich als das Verhältnis der Summe der mit den Modifikationsfaktoren gewichteten Zugehörigkeiten der Datenpunkte zum Cluster C_t zur Summe der ungewichteten Zugehörigkeiten ergibt.

Man beachte, daß in Gl. (4.13) nicht die Zuteilung sondern die Zugehörigkeit der Datenpunkte zu den Clustern verwendet wird, damit jedes Cluster unabhängig von den anderen bewertet wird. Dies ist für den im Abschnitt 4.2.7 beschriebenen Vereinigungsmechanismus und für die Interpretierbarkeit der generierten Cluster von Bedeutung. Die bei der Relevanzbewertung der Cluster erhaltenen Werte ϱ_t werden als Glaubensgrad bei der Verarbeitung des resultierenden Regelsatzes verwendet.

4.2.6 Löschen und Hinzufügen von Clustern

Aufgrund ihrer Äquivalenz zu Regeln sollten Cluster gelöscht werden, wenn ihre Relevanzbewertung zu schlecht ist oder wenn die Summe der Zuteilungen zu gering ist. Für diese beiden Löschbedingungen sind passende Schwellwerte ϱ_{min} und u^{α}_{min} zu wählen[2].

Dort wo Datenpunkte liegen, die in – bezüglich der Ausgangsgrößenwerte – inhomogenen Clustern zusammengefaßt sind, werden, in einer ähnlichen Vorgehensweise wie beim *Growing-Neural-Gas-Algorithmus* [Fri97], gezielt neue Cluster eingefügt: Das Konzept der Modifikationsfaktoren induziert, daß dem einzelnen Datenpunkt sogenannte *Verlustmasse*[3] zugeordnet wird. Dies ist die Differenz zwischen der ursprünglichen Masse 1 eines Datenpunktes und der Summe der Teilmassen, die sich aufgrund der modifizierten Zuteilung ergeben. Am Ende eines Lernschritts wird ein Datenpunkt, für den sich eine von 0 verschiedene Verlustmasse ergeben

[2]Bei Berechnung der Clusterparameter werden in Gl. (4.1) und (4.2) die mit α potenzierten Zuteilungen verwendet. Dementsprechend bezieht sich der Schwellwert u^{α}_{min} auf die Summe $\sum_{i=1}^{N} u^{\alpha}_{i,t}$.

[3]Diese wird unabhängig von einer eventuellen Re-Normalisierung der Partitionsmatrix nachgehalten.

hat, zufällig ausgewählt. Die Auswahlwahrscheinlichkeit wird proportional zu den Verlustmassen der Datenpunkte angesetzt. Der ausgewählte Datenpunkt dient zur Initialisierung eines neuen Clusters in der aktuellen Clusterpopulation. Wie auch bei der Initialisierung wird die Eingangsraumposition des Datenpunktes als neues Clusterzentrum und die Ausgangsgröße des Datenpunktes als Clusterausgangswert verwendet.

4.2.7 Vereinigen von Clustern

Eine geringere Anzahl von Clustern bedeutet ein weniger komplexes Modell und wirkt sich günstig auf die Interpretierbarkeit und oft auch auf die Generalisierungsfähigkeit und damit auf die Prognosegüte aus.

Beim ISODATA-Algorithmus [BD67], einem dem k-Means-Algorithmus ähnlichem Clusterverfahren, werden zwei Cluster miteinander vereinigt, wenn der Abstand der Clusterzentren einen vorzugebenden Schwellwert unterschreitet. Hier interessiert jedoch, ob der von zwei Clustern beschrieben Zusammenhang des Ein/Ausgangsverhaltens auch durch ein einziges, generalisiertes Cluster korrekt beschrieben werden kann. Deshalb werden hier eng benachbarte Cluster, deren zugeteilte Datenpunkte ähnliche Ausgangsgrößenwerte aufweisen und denen damit ähnliche Clusterausgangswerte zugewiesen worden sind, als Kandidaten für eine Vereinigung angesehen. Sie werden miteinander vereinigt, wenn die Relevanzbewertung des vereinigten Clusters nicht schlechter ist, als die beste Relevanzbewertung der bisherigen einzelnen Cluster. Die Vereinigung zweier Cluster geschieht, indem die entsprechenden Spalten der modifizierten Partitionsmatrix addiert und anschließend die Clusterparameter und die Relevanzbewertung neu berechnet werden.

Die Cluster, für die eine Vereinigung versucht wird, werden wie folgt ausgewählt: Für jedes Cluster C_t wird aus der Menge der Cluster gleichen oder[4] ähnlichen Ausgangsgrößenwertes das am nächsten liegende sogenannte *Partnercluster* ermittelt. Die Cluster werden mit zunehmenden Abstand zu ihrem Partnercluster in einer Rangliste sortiert[5]. Nun wird die Rangliste schrittweise durchgegangen und versucht, das jeweilige Cluster mit seinem Partnercluster zu vereinigen, wenn keines der beiden betrachteten Cluster bisher vereinigt wurde und somit nicht mehr in der Clusterpopulation existiert.

4.2.8 Pseudo-Code

Im folgenden wird der Pseudo-Code einer auf dem FCM-Algorithmus basierenden SMBC Variante angegeben. Als Wert für den Fuzziness-Parameter wird dabei $\alpha = 2$ gewählt und die Modifikationsfaktoren werden für Regressionsaufgaben nach Gl. (4.5) und für Klassifikationsaufgaben nach Gl. (4.9) berechnet. Die Routinen zum Löschen, Hinzufügen und Vereinigen von Clustern sind nicht mit angegeben.

[4]Im Falle der Klassifikation werden nur die Cluster mit gleichem Ausgangsgrößenwert betrachtet. Im Falle der Regression werden zu C_t dagegen alle Cluster betrachtet, deren Ausgangsgrößenwert – hier y_i genannt – in Verbindung mit dem Ausgangsgrößenwert c_t des Clusters C_t nach Gl. (4.5) einen Modifikationsfaktor ergibt, der oberhalb eines festzulegenden Schwellwertes ω_{min} liegt.

[5]Sollte ein Cluster C_t kein Partnercluster besitzen wird dieses Cluster ignoriert.

Algorithm 2: SMBC($\mathcal{P}$)

comment: Learn clusters $\mathcal{C}$ from given data tuples $\mathcal{P}$. The global K_{Start} is the number of clusters to start with and global β is used to calculate modification weights for regression tasks. Global σ_y is the deviation of the output.

global $K_{Start}, \beta, \sigma_y$

procedure INITIALIZE($\mathcal{P}$)
comment: select K_{Start} random data tuples $P_i = (\mathbf{x}_i, y_i)$ to initialize clusters
for $t = 1$ **to** K_{Start}
do $\begin{cases} i = \textit{index of random data tuple drawn without repetition} \\ C_t.\mathbf{v} = \mathbf{x}_i \\ C_t.c = y_i \end{cases}$
return ($\mathcal{C}$)

procedure PREDICT($\mathbf{x}, \mathcal{C}$)
comment: predict output value $\widehat{y}$ given input vector $\mathbf{x}$ using clusters $\mathcal{C}$
each cluster is a fuzzy rule and gets activated with a degree of $e^{-d_t^2}$ *with* d_t *being the distance of cluster* t *to the input* $\mathbf{x}$
if *problem is regression task*
then $\widehat{y} = COG$ *evaluation of activated rules*
else $\widehat{y} = MOM$ *evaluation of activated rules*
return ($\widehat{y}$)

procedure CLUSTERS($\mathcal{P}, \mathbf{U_{mod}}$)
comment: calculate clusters given data tuples $\mathcal{P}$ and the modified partition matrix $\mathbf{U}_{mod}$
for each *cluster* t
do $\begin{cases} \textit{set cluster center } \mathbf{v}_t \textit{ and output } c_t \textit{ to be a kind of} \\ \textit{weighted average of input and output values of} \\ \textit{assigned data tuples} \end{cases}$
return ($\mathcal{C}$)

procedure PARTITIONMATRIX($\mathcal{P}, \mathcal{C}$)
comment: calculate partition matrix $\mathbf{U}$ given data tuples $\mathcal{P}$ and clusters $\mathcal{C}$
set element $u_{i,t} = {d_{i,t}}^{-1}$ *with* $d_{i,t}$ *being the distance of tuple* i *to cluster* t
normalize partition matrix to have each row with a sum of 1
return ($\mathbf{U}$)

procedure MODIFICATIONMATRIX($\mathcal{P}, \mathcal{C}$)
comment: calculate modification matrix $\mathbf{\Omega}$ given data tuples $\mathcal{P}$ and clusters $\mathcal{C}$
for each *data tuple* i *and cluster* t
do $\begin{cases} \textbf{if } \textit{problem is regression task} \\ \quad \textbf{then } \omega_{i,t} = \exp - \left(\frac{C_t.c - y_i}{2\sigma_y \beta} \right)^2 \\ \quad \textbf{else } \omega_{i,t} = \begin{cases} 1 & C_t.c = y_i \\ 0 & \text{else} \end{cases} \end{cases}$
return ($\mathbf{\Omega}$)

main
$\mathcal{C}$ = INITIALIZE($\mathcal{P}$)
repeat
$\begin{cases} \mathbf{U} = \text{PARTITIONMATRIX}(\mathcal{P}, \mathcal{C}) \\ \mathbf{\Omega} = \text{MODIFICATIONMATRIX}(\mathcal{P}, \mathcal{C}) \\ \mathbf{U}_{mod} = \textit{element wise multiplication of matrices } \mathbf{U} \textit{ and } \mathbf{\Omega} \\ \mathcal{C} = \text{CLUSTERS}(\mathcal{P}, \mathbf{U}_{mod}) \\ \textit{delete, add, and merge clusters} \end{cases}$
until *abortion criterion is met*

return ($\mathcal{C}$)

4.3 Details und Erweiterungen

4.3.1 Geometrieparameter

Beim FCM-Algorithmus werden die Cluster lediglich durch ein Clusterzentrum gekennzeichnet; die Verteilung der Eingangsvektoren der dem Cluster zugeteilten Datenpunkte im Eingangsraum wird nicht berücksichtigt. Damit impliziert das verwendete Clustermodell sphärische, kugelförmige Cluster gleicher Größe. Im Gegensatz dazu werden die Clusterprototypen beim GK-Algorithmus durch Bestimmung der Kovarianzmatrix anhand der zugeteilten Datenpunkte durch zusätzliche Geometrieparameter charakterisiert. Dabei werden die Elemente der Kovarianzmatrix[6] bei gegebener Stichprobe durch die Partitionsmatrix[7] $\mathbf{U}$ zu

$$\mathbf{Q}_t = \sum_{i=1}^{N} u_{i,t}^{\alpha} (\mathbf{x}_i - \mathbf{v}_t)^T (\mathbf{x}_i - \mathbf{v}_t) \tag{4.14}$$

bestimmt. Ferner wird für die Berechnung der Partitionsmatrix anstelle des euklidischen Abstandes zwischen Clusterzentrum und Datenpunkt für jedes Cluster analog zu Gl. (2.27) eine Art individuelle Mahalanobis-Distanz

$$d_{i,t}^2 = \sqrt[m]{\det(\mathbf{Q}_t)}\ (\mathbf{x}_i - \mathbf{v})^T\ \mathbf{Q}_t^{-1}\ (\mathbf{x}_i - \mathbf{v}_t) \tag{4.15}$$

verwendet. Der Term $\sqrt[m]{\det(\mathbf{Q}_t)}$ bewirkt eine Normalisierung der Clustergröße. Damit impliziert das beim GK-Algorithmus verwendete Clustermodell ellipsoide Cluster beliebiger Orientierung aber gleicher Größe. Für die Berechnung der Zugehörigkeit für die Relevanzbewertung der Cluster und zur Auswertung des äquivalenten Regelsatzes wird hier jedoch in den Gln. (4.11) und (4.13) auf diese Normierung verzichtet. Für die Startinitialisierung der Geometrieparameter wird die Einheitsmatrix verwendet. Beim Hinzufügen eines neuen Clusters werden die Geometrieparameter vom nächstliegenden Cluster übernommen.

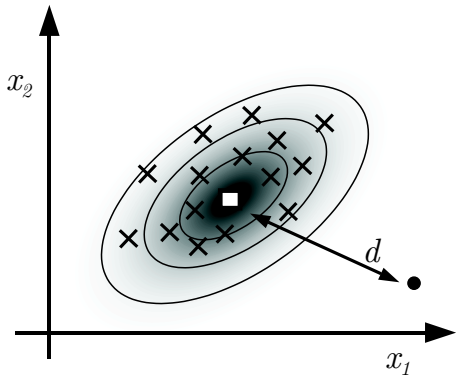


Abbildung 4.3: Zugeteilte Datenpunkte und daraus gebildetes Cluster beim GK-Algorithmus. Skizziert ist das Clusterzentrum und die Form der aufgrund von Gl. (4.15) resultierenden Äquidistanzlinien im Eingangsraum.

[6] Gustafson & Kessel verwendeten ursprünglich sogenannte Fuzzy-Kovarianzmatrizen, die sich nach Gl. (4.14) ergeben wenn zusätzlich sämtliche Elemente durch $\sum_{i=1}^{N} u_{i,t}^{\alpha}$ dividiert werden. Siehe hierzu [HKK97] Seite 43.

[7] Wir verwenden hier die modifizierte Partitionsmatrix $\mathbf{U}_{mod}$.

Die Anzahl der zu berechnenden Parameter der Kovarianzmatrix steigt quadratisch mit der Dimensionalität des Problems. Damit steigt gleichermaßen die Anzahl der Datenpunkte, die notwendig sind, um die Parameter mit einer bestimmten statistischen Robustheit zu bestimmen. Vergleiche hierzu Kapitel 2.5. Ebenso ist es bei Verwendung der vollen Kovarianzmatrix nicht mehr möglich, die den Clustern äquivalenten Fuzzy-Regeln auf die Form (4.12) zu bringen, da aufgrund der Drehung des Koordinatensystems keine komponentenweisen Abstände mehr ermittelt werden können. Oft werden daher nur die Diagonalelemente der Kovarianzmatrix ermittelt und sämtliche Elemente der Nebendiagonalen zu 0 gesetzt, womit die Clusterprototypen jedoch wieder nur achsparallele Strukturen implizieren. Die Diagonalelemente sind bis auf einen zu vernachlässigenden konstanten Faktor identisch mit den komponentenweise berechneten Varianzen der den Cluster zugeteilten Datenpunkte bezüglich des Clusterzentrums.

Ein allgemeinerer von Banfield & Raftery entwickelter Weg wird in [BR93, FR98] beschrieben. Durch Re-Parametrisierung der Kovarianzmatrix mittels Singulärwertzerlegung ist es möglich, sehr flexibel eine große Vielzahl von verschiedenen Clustermodellen mit der kennzeichnenden Matrix

$$\mathbf{Q}_t = \mathbf{D}_t \; \mathbf{\Lambda}_t \; \mathbf{D}_t^T \tag{4.16}$$

aufzustellen. Dabei ist $\mathbf{D}_t$ eine orthogonale Matrix mit den Eigenvektoren der Matrix $\mathbf{Q}_t$. Die Matrix $\mathbf{\Lambda}_t$ ist eine Diagonalmatrix mit den Eigenwerten der Matrix $\mathbf{Q}_t$ als Diagonalelementen; sie kann weiter in $\Lambda_t = \lambda_t \mathbf{A}_t$ zerlegt werden. Die Matrix $\mathbf{D}_t$ bestimmt damit die Orientierung, die Matrix $\mathbf{A}_t$ die Form und der Parameter λ_t die Größe des Clusterellipsoids. Jeder Term wird entweder individuell für jedes Cluster einzeln ermittelt, als Mittel über alle Cluster berechnet oder aber a priori festgelegt.

4.3.2 Skalierbarkeit

Von großer Bedeutung für die Abschätzung der praktischen Anwendbarkeit eines Lernverfahrens für eine gegebene Lernaufgabe ist die sogenannte *Ordnung*. Sie gibt an, wie sich die Laufzeit bzw. der Speicherplatzbedarf von einem Algorithmus mit zunehmender Größe des zu bearbeitenden Problems erhöht. Die Größe des Problems ist im Kontext dieser Arbeit gegeben durch die Anzahl der Lerndatenpunkte N_L und durch die Anzahl der Eingangsgrößen m. Wenn mit einer festen maximalen Anzahl an Lernepochen und einer Obergrenze für die Anzahl der Cluster gearbeitet wird, hat der SMBC-Algorithmus eine jeweils mit der Anzahl der Datenpunkte linear und mit der Anzahl der Eingangsgrößen kubisch zunehmende Ordnung $O(N_L\, m^3)$. Dies ergibt sich wie folgt: Für jede Lernepoche müssen einmalig die Partitions- und Modifikationsmatrix berechnet und daran anschließend die Clusterparameter neu ermittelt werden. Diese Operationen skalieren linear mit der Anzahl der Datenpunkte und der Cluster. Für die Berechnung der Abstände der Datenpunkte zu den Clustern ist die Inversion einer $m \times m$ Matrix erforderlich, die bei Verwendung eines üblichen Gauß'schen Eliminationsverfahrens eine Ordnung von $O(m^3)$ aufweist[8].

Allerdings ist zu beachten, daß die obigen Voraussetzungen mit steigender Stichprobengröße ggf. nicht mehr zu rechtfertigen sind und es sinnvoll sein kann, eine größere Anzahl an Lernschritten und Clustern vorzusehen.

[8] Je nach verwendeten Clustermodell ist die Inversion einer Matrix nicht erforderlich bzw. bei einer Diagonalmatrix mit einer Ordnung von $O(m)$ durchführbar, womit sich die Ordnung des Verfahrens auf $O(Nm)$ reduziert.

4.4 Einordnung

Vergleich mit RBF-Netzen

RBF-Netze [Orr96] modellieren den Zusammenhang zwischen der Ausgangsgröße und den Eingangsgrößen durch eine Linearkombination

$$y = f(\mathbf{x}) = \sum_{t=1}^{K} c_t^* g_t(\mathbf{x}). \tag{4.17}$$

von K Teilfunktionen g_t. Als Teilmodell werden häufig durch einen Mittelpunkt $\mathbf{v}$ und skalare oder komponentenweise Radien r bzw. $\mathbf{r}$ definierte Kernelfunktionen gewählt. Ein einfaches Beispiel für den eindimensionalen Fall ist

$$g(x) = \exp - \left(\frac{x - v}{r} \right)^2. \tag{4.18}$$

Die Erweiterung für den mehrdimensionalen Fall ist trivial. Neben Exponentialfunktionen werden in der Literatur verschiedene andere Funktionen untersucht. Beispielsweise schlägt Orr [Orr96] den allgemeinen Ansatz

$$g(\mathbf{x}) = F(d^2) \qquad \text{mit} \qquad d^2 = (\mathbf{x} - \mathbf{v})^T \mathbf{R}^{-1} (\mathbf{x} - \mathbf{v}) \tag{4.19}$$

für die Modelle $g(\mathbf{x})$ vor, wobei $\mathbf{R}$ eine geeignet definierte und invertierbare Matrix und F irgendeine eindimensionale Kernelfunktion ist, die vorzugsweise kontinuierlich, beschränkt und integrierbar zu 1 zu wählen ist. Das Lernen eines RBF-Netzes wird oft auf die Bestimmung günstiger Gewichtsfaktoren für einen festen Kanon von möglichen Kernelfunktionen beschränkt. Für eine bestimmte Auswahl aus der Menge der möglichen Kernelfunktionen ist dann die Bestimmung optimaler Gewichtsfaktoren durch Lösung eines linearen Gleichungssystems mittels der Pseudoinversen[9] durchführbar. Die Initialisierung, d.h. die Vorgabe möglicher Kernelfunktionen erfolgt u.a. durch Clusterverfahren. Alternativ werden alle Lerndatenpunkte als mögliche Kernelmittelpunkte verwendet. Die Ausdehnungsparameter ergeben sich dann – zumeist sehr einfach – aus statistischen Kennwerten der Lerndaten.

Das aus Abschnitt 4.2.4 im Falle der Regression resultierende Modell liefert zu einem gegebenen Eingangsvektor $\mathbf{x}$ den prognostizierten Ausgangsgrößenwert y in der Form

$$y(\mathbf{x}) = \frac{\sum_{t=1}^{K} c_t \mu_t(\mathbf{x})}{\sum_{t=1}^{K} \mu_t(\mathbf{x})}. \tag{4.20}$$

Darin sind c_t die den Clustern C_t zugeordneten Ausgangsgrößenwerte und $\mu_t(\mathbf{x})$ die Zugehörigkeitsgrade des Vektors $\mathbf{x}$ zu den einzelnen Clustern C_t. Kennzeichnend für unseren Ansatz ist, daß sich die Werte c_t aus dem Clusterprozeß ergeben. Dieser ist – wie beschrieben – im Interesse einer guten Interpretierbarkeit der resultierenden Regeln so angelegt, daß sich möglichst homogene Cluster ergeben. Dies sind Cluster, die möglichst nur Datenpunkte mit gleichen oder

[9] Die Pseudoinverse $\mathbf{A}^+ = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$ löst die Gl. $\mathbf{Ax} = \mathbf{b}$ im Sinne der Minimierung der quadratischen Fehler zu $\mathbf{x} = \mathbf{A}^+\mathbf{b}$.

ähnlichen Ausgangsgrößenwerten abdecken. Dementsprechend lassen sich die Funktionen $\mu_t(\mathbf{x})$ und die Werte c_t als lokale Charakterisierungen der Menge der Datenpunkte interpretieren. Mit

$$u_t(\mathbf{x}) = \frac{\mu_t(\mathbf{x})}{\sum_{h=1}^{K} \mu_h(\mathbf{x})} \tag{4.21}$$

läßt sich Gl. (4.20) auch in der Form

$$y(\mathbf{x}) = \sum_{t=1}^{K} c_t u_t(\mathbf{x}) \tag{4.22}$$

schreiben. Damit kann man die aus dem Clusterprozeß hervorgegangenen Funktionen $u_t(\mathbf{x})$ zur Initialisierung der Kernelfunktionen für ein RBF-Netz verwenden und anstelle der bisherigen Werte c_t mit Hilfe der Pseudoinversen neue Werte c_t^* bestimmen.

In ersten Versuchen konnte die Prognosegüte damit jedoch nur geringfügig gesteigert werden. Im Gegensatz dazu führte die Initialisierung der Menge möglicher Kernelfunktionen mit allen Lerndatenpunkten zu wesentlich besseren Ergebnissen; allerdings unter letztendlicher Verwendung einer viel größeren Anzahl an Kernelfunktionen.

Die Ursache hierfür ist darin zu sehen, daß die Auswahl der Kernelfunktionen beim üblichen RBF-Ansatz mit Blick auf das Gesamtverhalten erfolgt. Es wird dabei also das Zusammenspiel der Kernelfunktionen von Anfang an berücksichtigt. Dies führt zu einer hohen Prognosegüte – allerdings auf Kosten der Interpretierbarkeit: Häufig ergeben sich stark überlappende Kernelfunktionen, d.h. für eine Eingangsraumposition sind meistens die Funktionswerte mehrerer Kernelfunktionen signifikant von 0 verschieden. Zudem sind die sich ergebenden Gewichtsfaktoren – die positiv und negativ sein können – nicht mehr lokal interpretierbar.

Der hier aufgezeigte Zielkonflikt zwischen Modellierungsgüte und Interpretierbarkeit ist im übrigen generell bei der datenbasierten Fuzzy-Modellierung anzutreffen. Man kann – wie beim FR-Algorithmus – auf Regeln abzielen, die jede für sich einen lokal sinnvollen Aspekt der betrachteten Daten darstellen und somit interpretierbar sind. Alternativ kann man auch auf die Interpretierbarkeit verzichten und die Regeln nebst Zugehörigkeitsfunktionen allein danach auswählen bzw. bestimmen, daß das resultierende Modell eine möglichst hohe Prognosegüte liefert.

Zur Abrundung der Einordnung und zum Vergleich sei hier angemerkt, daß sich die für RBF-Netze kennzeichnende Vorschrift (4.17) auch wie folgt interpretieren läßt: Jedem Tupel $(c_t^*, g_t(\mathbf{x}))$ entspricht eine Regel

$$\text{WENN } \mathbf{x} \textit{ ist } g_t(\mathbf{x}) \text{ DANN } y = c_t^*, \tag{4.23}$$

wobei der Erfülltheitsgrad der Prämisse sich direkt zu

$$\mu_t(\mathbf{x}) = g_t(\mathbf{x}) \tag{4.24}$$

ergibt und die Defuzzifizierung durch die Drehmomentmethode (TOR) [Kie97] erfolgt. Diese Sicht macht deutlich, daß die datenbasierte Generierung von interpretierbaren Regeln für ein Fuzzy-Modell, in dem die Drehmomentmethode zur Defuzzifizierung verwendet wird, ein noch nicht befriedigend gelöstes Problem aufwirft: Die Relevanz einer Regel kann nicht durch eine isolierte Betrachtung dieser Regeln, sondern nur durch Berücksichtigung des Zusammenspiels aller Regeln ermittelt werden.

Kapitel 5

Experimente und Anwendungsbeispiele

In diesem Kapitel werden eine Einführung in mögliche Maße zur Bewertung der Prognosegüte eines Lernverfahrens gegeben und mit der Kreuz-Validierung und der mehrfachen Wiederholung Verfahren zur Ermittlung derselben vorgestellt. Ein Problem stellt die Einstellung der freien Parameter eines Lernverfahrens dar. Erfolgt diese nicht systematisch, kann dies die Validität der erhaltenen Ergebnisse beeinträchtigen. Daher wird basierend auf einer Arbeit von Salzberg die Vorgehensweise der *harten Validierung* definiert. Es folgt eine Beschreibung der durchgeführten Experimente und Untersuchungen sowie eine Darstellung und Diskussion der erhaltenen Ergebnisse.

5.1 Gütemaße

Prognosegüte

Die wichtigsten Gütemaße zur Bewertung eines Modells stellen diejenigen Maße dar, die die Prognosegüte bewerten. Einige weit verbreitete Maße für Klassifikations- und Regressionsaufgaben sind im folgenden beschrieben. Üblicherweise werden diese Maße bezüglich einer Teststichprobe ausgewertet. Dazu wird zu jedem Datenpunkt der Teststichprobe anhand des Eingangsvektors ein Prognosewert für die Ausgangsgröße ermittelt und die Abweichung vom tatsächlichen Ausgangsgrößenwert dann mit dem Gütemaß bewertet. Es ist prinzipiell auch möglich – wenn auch oft nicht besonders aussagekräftig – die Gütemaße bezüglich der Lernstichprobe auszuwerten. Um Mißverständnisse zu vermeiden, sei darauf hingewiesen, daß sich der Begriff *Prognosegüte*, so nicht explizit anders angegeben, in dieser Arbeit immer auf eine Ermittlung bezüglich einer Teststichprobe bezieht.

Klassifikation Der mittlere Klassifizierungsfehler (MCE) gibt an, für wieviel Prozent der Datenpunkte die Ausgangsklasse falsch prognostiziert wurde.

$$MCE = \frac{\sum_{i=1}^{N} e_i}{N} \quad \text{mit} \quad e_i = \begin{cases} 0 & \widehat{y}_i = y_i \\ 1 & \widehat{y}_i \neq y_i \end{cases} \tag{5.1}$$

Dabei bezeichnet $\widehat{y}_i$ jeweils die prognostizierte und y_i die tatsächliche Ausgangsklasse eines Datenpunktes und N die Stichprobengröße. Die erhaltenen Gütewerte liegen innerhalb des Intervals $[0, 1]$ und sind direkt interpretierbar.

Um tieferen Einblick in die genaue Art der Fehlklassifikationen zu erhalten, kann die sogenannte *Konfusionsmatrix* **O** aufgestellt werden. Diese erlaubt es festzustellen, wie oft für eine bestimmte Ausgangsklasse fälschlicherweise eine andere bestimmte Ausgangsklasse prognostiziert wurde. Die Konfusionsmatrix ist eine quadratische Matrix deren Größe der Anzahl S_y der möglichen Ausgangsklassen entspricht. Ihre Elemente werden gebildet zu

$$o_{w,z} = \sum_{i=1}^{N} e_i \quad \text{mit} \quad e_i = \begin{cases} 1 & \widehat{y}_i = w \ \wedge \ y_i = z \\ 0 & \text{sonst} \end{cases} \quad . \tag{5.2}$$

Teilweise ist die Fehlklassifikation einer bestimmten Ausgangsklasse wesentlich stärker zu bewerten als bei einer anderen Ausgangsklasse. So verursacht z.B. die fälschliche Einstufung eines *kreditunwürdigen* Kunden als *kreditwürdig* wesentlich höhere Kosten als im umgekehrten Falle die Einstufung eines *kreditwürdigen* Kunden als *kreditunwürdig*. Im ersten Falle muß der gesamte Kredit abgeschrieben werden, im zweiten Falle entsteht lediglich ein Umsatzverlust. Bei solchen Lernaufgaben empfiehlt es sich daher, die Kosten der jeweiligen Fehlklassifikation mit zu berücksichtigen und als Gütemaß die mittleren Fehlklassifizierungskosten (MMC) anzugeben. Im allgemeinen Fall sind die Fehlklassifizierungskosten in einer quadratischen Matrix **B**, deren Größe der Anzahl S_y der möglichen Ausgangsklassen entspricht, angegeben. Sämtliche Diagonalelemente dieser Matrix sind 0. Das Nebendiagonalelement $b_{w,z}$ enthält die Kosten, die entstehen, wenn für einen Datenpunkt der Ausgangsklasse w die Ausgangsklasse z prognostiziert wird. Die mittleren Fehlklassifizierungskosten ergeben sich dann durch das elementweise multiplizieren und anschließende Aufsummieren der Elemente der Matrizen **O** und **B** zu

$$MMC = \frac{\sum_{w=1}^{S_y} \sum_{z=1}^{S_y} o_{w,z} \ b_{w,z}}{N} \ . \tag{5.3}$$

Regression Der mittlere absolute bzw. der mittlere quadratische Fehler (MAE bzw. MSE) gibt an, welcher absolute bzw. quadratische Fehler im Mittel bei der Prognose über alle Datenpunkte der betrachteten Stichprobe gemacht wird.

$$MAE = \frac{\sum_{i=1}^{N} |\widehat{y}_i - y_i|}{N} \tag{5.4}$$

$$MSE = \frac{\sum_{i=1}^{N} (\widehat{y}_i - y_i)^2}{N} \tag{5.5}$$

Wiederum bezeichnet dabei $\widehat{y}_i$ jeweils die prognostizierten und y_i die tatsächlichen Ausgangsgrößenwerte eines Datenpunktes und N ist die Stichprobengröße. Die sich ergebenden Gütewerte sind meist nicht direkt interpretierbar. Um einen Gütewert auf den ersten Blick einschätzen zu können, kann eine Normierung auf denjenigen Fehler, der sich bei einer sogenannten *Baseline-Prognose* [RNH+96] ergeben hätte, vorgenommen werden. Bei einer solchen Prognose wird unabhängig vom jeweiligen Eingangsgrößenvektor für jeden Datenpunkt immer ein konstanter Wert y_{const} prognostiziert. Für das Gütemaß MAE bzw. MSE wird y_{const} als Medianwert bzw. arithmetischer Mittelwert der Ausgangsgrößenwerte der Lernstichprobe angesetzt[1]. Die

[1]Für das MSE-Kriterium ist diese Vorgehensweise identisch mit einer Normierung auf die Streuung der Ausgangsgrößenwerte

normierten MAE- und MSE-Gütewerte sollten wesentlich kleiner als eins sein, ansonsten sind Zweifel an der Prognoseeignung des gelernten Modells angebracht, da eine einfache *Baseline-Prognose* zu ähnlichen Prognosegüten führt.

Experiment-Design

Lernverfahren können experimentell anhand einer Lernaufgabe bzw. der dazugehörigen Stichprobe bezüglich der erzielten Prognosegüten miteinander verglichen werden. Im folgenden wird zunächst der Begriff der tatsächlichen Güte eines Modells Q_P^* eingeführt und darauf aufbauend die tatsächliche Güte eines Lernverfahrens Q^* definiert. Anschließend werden mit der Kreuz-Validierung und der mehrfachen Wiederholung zwei Schätzer zur Ermittlung derselben vorgestellt.

Unter tatsächlicher Güte Q_P^* eines Modells für eine gegebene Lernaufgabe wird diejenige Güte bezüglich eines Gütemaßes verstanden, die sich ergeben würde, wenn mittels der zugrundeliegende wahren Funktion $y = f_{tf}(\mathbf{x})$ unendlich viele Datenpunkte generiert und zum Testen verwendet würden. Die Eingangsvektoren dieser Testdatenpunkte würden dabei anhand der Verteilung $\mu_{tf}(\mathbf{x})$ produziert.

Sowohl $y = f_{tf}(\mathbf{x})$, als auch $\mu_{tf}(\mathbf{x})$ sind typischerweise nicht bekannt, weshalb die tatsächliche Güte auch meist nicht exakt ermittelt werden kann, und daher eher eine theoretische Größe darstellt. Bei Aufteilung einer gegebenen Stichprobe in Lern- und Testdaten steht nur eine endliche Anzahl an Testdatenpunkten zur Verfügung, und somit kann nur die Güte $\widehat{Q}_P$ bezüglich dieser Stichprobe ermittelt werden.

Unter tatsächlicher Güte Q^* eines Lernverfahrens an einer Lernaufgabe bei feststehender Lernstichprobengröße wird diejenige Güte bezüglich eines Gütemaßes verstanden, die sich im arithmetischen Mittel der tatsächlichen Güten Q_P^* der Modelle ergibt, wenn unendlich oft auf einer N_L-elementigen Lernstichprobe ein Modell gelernt wird. Auch Q_P^* ist i.allg. nicht exakt ermittelbar. Um den Wert zu schätzen kommen daher üblicherweise Methoden der Kreuz-Validierung oder mehrfachen Wiederholung zum Einsatz.

- **$\mathbf{N_R}$-fache Kreuz-Validierung:** Die Stichprobe wird zufällig in N_R etwa gleich große kleinere Stichproben aufgeteilt. Jede der kleineren Stichproben wird einmal als Teststichprobe zurückgehalten und unter Verwendung der Datenpunkte der restlichen $N_R - 1$ Stichproben wird ein Modell gelernt und dessen Prognosegüte $\widehat{Q}_P$ bezüglich der zurückgehaltenen Teststichprobe ermittelt. Kohavi berichtet in [Koh95], daß $N_R = 10$ eine gute Wahl ist, auch wenn die Rechenkapazitäten einen größeren Wert zulassen würden. Eine besondere Form der Kreuz-Validierung ist die sogenannte *Leave-One-Out Cross-Validation* (LOOCV). Bei dieser wird N_R zur Anzahl der Datenpunkte der Stichprobe gewählt.

- **$\mathbf{N_R}$-fache Wiederholung:** Die Stichprobe wird N_R mal zufällig in Lern- und Teststichproben aufgeteilt. Auf der Lernstichprobe wird ein Modell gelernt und die Prognosegüte $\widehat{Q}_P$ desselben bezüglich der Teststichprobe ermittelt.

Als Ergebnis $\widehat{Q}$ wird der arithmetische Mittelwert der N_R einzelnen Prognosegüten $\widehat{Q}_P$ angegeben. Es ist möglich, den sogenannten *Standard-Fehler* σ_Q^* der Schätzung zu

$$\sigma_Q^* = \frac{\sigma_Q}{\sqrt{N_R}} \tag{5.6}$$

anzugeben. Dabei bezeichnet σ_Q die Standardabweichung der Prognosegüten $\widehat{Q}_P$. Wenn davon ausgegangen wird, daß die Prognosegüte $\widehat{Q}_P$ normalverteilt ist, so ist der erhaltene Schätzwert mit einer Wahrscheinlichkeit von etwa 67% nicht weiter als σ_Q^* vom tatsächlichen Gütewert entfernt[2].

Weitere Gütemaße

Neben der Prognosegüte gibt es eine Reihe weiterer Aspekte, nach denen die gelernten Modelle, und allgemein auch die zum Lernen verwendeten Verfahren, miteinander verglichen werden können. Es sind dies:

- **Größe** des Modells, wobei darunter üblicherweise der benötigte Speicherplatzbedarf für das gelernte Modell verstanden wird. Bei einem regelbasierten Lernverfahren ist die Größe bestimmt durch die über alle Regeln aufsummierte Anzahl der Prämissen- und Konklusionsterme. Die Größe ist u.a. wichtig, wenn für das erhaltene Modell die Eigenschaft der Interpretierbarkeit bewertet werden soll.
- **Zeitbedarf**, d.h. Rechenaufwand der zum Lernen eines Modells bzw. zum Treffen einer Prognose benötigt wird.
- **Notwendiges Hintergrundwissen**, das ein Benutzer besitzen muß, um das Lernverfahren korrekt und mit akzeptablen Prognosegüten anwenden zu können. An dieser Stelle ist insbesondere die Problematik der Einstellung freier Parameter eines Lernverfahrens zu berücksichtigen.

5.2 Einstellung freier Parameter

Salzberg benennt in [Sal99] das Problem des *Repeated Tuning*:

> *Many researchers tune their algorithms repeatedly in order to make them perform optimally on at least some data sets. At the very least, when a system is being developed, the developer spends a great deal of time determining what parameters it should have and what the optimal values should be. [...] Fortunately one can perform virtually unlimited tuning without corrupting the validity of the results. The solution is to use cross-validation entirely within the training set itself. [...] When the parameters appear to be at their optimal settings, accuracy can finally be measured on the test data.*
> *Thus when doing comparative evaluations,* everything *that is done to modify or prepare the algorithms must be done in advance of seeing the test data. This point will seem obvious to many experimental researchers, but the fact is that papers are still appearing in which this methodology is not followed. In the survey by Flexer [Fle96], only 3 out of 43 experimental papers in leading neural network journals used a separate data set for parameter tuning; the remaining 40 papers either did not explain how they adjusted parameters or else did their adjustments after using the test set.*

Damit hat Salzberg sowohl ein häufig in der Literatur auftretendes Problem benannt, als auch einen Ausweg skizziert, wie dem Problem mittels des Einsatzes einer Kreuz-Validierung oder

[2] Es handelt sich hierbei um eine sehr optimistische Schätzung, da in Gl. (5.6) vorausgesetzt wird, daß es sich um N_R unabhängige Experimente handelt, was nicht gegeben ist, wenn Datenpunkte mehrfach in den Experimenten verwendet werden.

ähnlicher Ansätze zum automatischen Parametertuning begegnet werden kann. Diese Vorgehensweise wird im folgenden als *harte Validierung*, im Gegensatz zur üblicherweise verwendeten *normalen Validierung*, bezeichnet.

Jedoch, und dies wird von SALZBERG nur teilweise deutlich gemacht, gibt es Situationen, in denen die normale Validierung durchaus ihre Berechtigung hat. So ist die Durchführung einer harten Validierung mit einem wesentlich höheren Arbeitsaufwand seitens des Entwicklers verbunden und für die notwendigen Experimente wird deutlich mehr Rechenleistung benötigt. Die Definition geeigneter Standard-Parametersätze, aus denen während des Parametertunings der letztlich verwendete Parametersatz ausgewählt wird, als auch die Realisierung des Parametertunings selbst, sind nicht triviale Aufgaben. Bei der Entwicklung eines neuen Algorithmus bzw. einer neuen Idee für einen bestehenden Algorithmus stellt sich zunächst die Frage der prinzipiellen Funktionsfähigkeit. Designentscheidungen sind zu treffen und für einige wenige Strategieelemente sind Parameter vorzusehen, um die Arbeitsweise des Algorithmus flexibel an verschiedene Probleme anpassen zu können. In diesem Entwicklungsprozeß ist eine normale Validierung vollkommen ausreichend, eine harte Validierung aufgrund des viel höheren Arbeitsaufwandes hingegen eher ungeeignet.

Prinzipielle Vorgehensweise zur harten Validierung

Die im folgenden vorgestellte Vorgehensweise zur harten Validierung orientiert sich an den Ausführungen in [Sal97]. Es wird zunächst davon ausgegangen, daß für alle freien Parameter des zu testenden Algorithmus' sogenannte *Standard-Parametersätze* definiert sind.

Die gegebene Stichprobe wird mittels N_R-facher Wiederholung oder Kreuz-Validierung in Lern- und Teststichproben aufgeteilt. Für jede Lernstichprobe wird der vermutlich beste Parametersatz aus der Menge der Standard-Parametersätze durch ein sogenanntes *Parametertuning* ausgewählt. Mit dem ausgewählten Parametersatz wird dann auf der betrachteten Lernstichprobe ein Modell gelernt und dessen Prognosegüte auf der zugehörigen Teststichprobe ermittelt. Der arithmetische Mittelwert über die N_R einzelnen Prognosegüten wird berichtet.

Im folgenden wird die Vorgehensweise zum Parametertuning genauer beschrieben: Die jeweils betrachtete Lernstichprobe wird weiter in die zum Parametertuning verwendeten Tuning-Lern- und Tuning-Teststichproben aufgeteilt. Zur Aufteilung wird diesmal eine N_{Tune}-fache Wiederholung oder Kreuz-Validierung verwendet. Für jeden möglichen Parametersatz wird nun auf den Tuning-Lernstichproben ein Modell gelernt und auf den zugehörigen Tuning-Teststichproben dessen Prognosegüte ermittelt. Anhand dieser Prognosegüten und ggf. unter Berücksichtigung weiterer Kriterien, wie beispielsweise der Größe der gelernten Modelle, wird der vermutlich beste Parametersatz ausgewählt. Zwei einfache Auswahlmechanismen sind im Rahmen dieser Arbeit betrachtet worden:

- **mittlere *Lernstichproben-Prognosegüte*:** Die mittlere Lernstichproben-Prognosegüte ergibt sich als arithmetischer Mittelwert über die einzelnen Prognosegüten für die N_{Tune} Wiederholungen oder Kreuz-Validierungen.
- **mittlerer *Lernstichproben-Rang*:** Für jede der N_{Tune} Wiederholungen oder Kreuz-Validierungen werden die Parametersätze mit abnehmender Prognosegüte sortiert. Der Parametersatz mit dem besten Gütewert erhält einen Rang von 1, die anderen Parametersätze folgen entsprechend. Bei gleicher Prognosegüte mehrerer Parametersätze erhalten alle den gleichen Rang. Danach wird mit dem nächsthöheren Rang fortgefahren. Der mittlere Lernstichproben-Rang ergibt sich als arithmetischer Mittelwert über die einzelnen N_{Tune} Ränge.

Man beachte das grundlegende Problem, daß bei der automatischen Auswahl des vermutlich besten Parametersatzes *Overfitting* auftreten kann, denn der ausgewählte Parametersatz ist lediglich optimal bezüglich der aus der jeweiligen Lernstichprobe generierten Tuning-Lern- und Tuning-Teststichproben, nicht aber garantiertermaßen optimal bezüglich der Teststichprobe auf der letztendlich die Prognosegüte ermittelt wird. Dies kann sich negativ auf die bei einer harten Validierung erreichten Prognosegüten auswirken.

Parameterklassen bei harter Validierung

Bei der oben beschriebenen Vorgehensweise zur harten Validierung wird jeder freie Parameter des zu testenden Algorithmus' im Rahmen des Parametertunings betrachtet. Der Rechenaufwand steigt jedoch mit zunehmender Anzahl der betrachteten Parameter, so daß man bestrebt ist, möglichst viele freie Parameter vorab festzulegen, und dann nicht weiter betrachten zu müssen. Durch Überlegungen zur Wirkungsweise der einzelnen freien Parameter oder durch Voruntersuchungen an einigen ausgewählten Benchmarks können einige freie Parameter, wie beispielsweise die Art und Ausgestaltung der verwendeten Abstandsfunktion, festgelegt werden. Die derart festgelegten Parameter werden im folgenden als *Designparameter* bzw. *Designentscheidungen* bezeichnet. Die verbleibenden freien Parameter des Algorithmus werden fortan *Strategieparameter* genannt.

Der beim Parametertuning verwendete Auswahlmechanismus und die zur Ermittlung der Lernstichproben-Prognosegüte bzw. des Lernstichproben-Rangs durchgeführte Bestimmungsmethode stellen auch Parameter dar, und müssen daher ebenfalls in Voruntersuchungen festgelegt werden. Im folgenden werden diese Parameter als *Tuningparameter* bezeichnet.

Ferner sind die Parameter, die das Experiment-Design bestimmen, zu betrachten. Dies ist beispielsweise die Art, wie die gegebene Stichprobe in Lern- und Teststichproben aufgeteilt wird und die Anzahl der durchgeführten Wiederholungen oder Kreuz-Validierungen N_R. Diese sogenannten *Experimentparameter* sind unkritisch, d.h. sie müssen und können nicht in Voruntersuchungen festgelegt werden. In der Literatur finden sich zahlreiche Beispiele zu einer sinnvollen Wahl der Experimentparameter [WD95, Dom97, WM00a].

Tabelle 5.1: Parameterklassen bei harter Validierung.

Klasse	**Erläuterung**
Designparameter	Freie Parameter des Algorithmus, die durch Überlegungen zur Wirkungsweise festgelegt werden.
Designentscheidungen	Freie Parameter des Algorithmus, die durch Voruntersuchungen festgelegt werden.
Strategieparameter	Freie Parameter des Algorithmus, die im Rahmen des Parametertunings betrachtet werden.
Tuningparameter	Parameter, die die genaue Vorgehensweise beim Parametertuning bestimmen.
Experimentparameter	Parameter, die das Experiment-Design an sich festlegen.

Die Einteilung der Parameter in die verschiedenen Klassen ist in Tabelle 5.1 zusammenfassend aufgeführt. Die Zuordnung eines freien Parameter des zu testenden Algorithmus zu einer der drei dabei möglichen Klassen ist vom Experimentator vorzunehmen und damit in gewisser Weise subjektiv. Für die in den Voruntersuchungen verwendeten Benchmarks ist prinzipiell keine

harte Validierung mehr möglich. Die erzielten Ergebnisse dürfen entweder nicht im Rahmen einer harten Validierung berichtet werden, oder aber sie müssen entsprechend gekennzeichnet sein.

5.3 Ergebnisse PNC 2

Ein neuer Lernalgorithmus sollte mit den ihm am *ähnlichsten* in der Literatur bekannten Algorithmen verglichen werden [Sal99]. Dies sind der im Kapitel 3.4 beschriebene NGE und RISE, sowie der im Kapitel 2.3 beschriebe kNN-Algorithmus. Das vom PNC 2-Algorithmus generierte Modell kann in die Form eines Fuzzy-Regelsatzes transformiert werden. Daher wird auch der in Kapitel 2.2.1 beschriebene FR-Algorithmus betrachtet. Zum Vergleich werden umfangreiche Benchmarkstudien durchgeführt. Derartige Studien wurden auch bereits mit der Vorgängerversion des PNC 2-Algorithmus gemacht und in [Hae01b, Hae01a] veröffentlicht. Dabei wurden zwar einige sinnvoll erscheinende freie Parameter des Algorithmus als Strategieparameter eingestuft, die übrigen Parameter jedoch, ebenso wie die benötigten Standard-Parametersätze und die Tuningparameter, ad hoc aufgrund einfacher Vorüberlegungen eingestellt. Für den PNC 2-Algorithmus werden dagegen alle Parameter in die verschiedenen Klassen nach Tabelle 5.1 eingeteilt, systematisch in ihrer Wirkungsweise untersucht und dann festgelegt. Dazu werden Vorüberlegungen angestellt und in Voruntersuchungen Experimente mit einigen ausgewählten Benchmarks durchgeführt.

Tabelle 5.2: Überblick über die durchgeführten Voruntersuchungen zum PNC 2-Algorithmus.

Kürzel	**Fragestellung**	**Erläuterungen**
EXPA	Designentscheidungen	Wahl der Normierungs- und Diskretisierungsmethode, Entscheidung über die Verwendung von Glaubensgraden zur Regelbewertung etc.
EXPB	Strategieparameter	Parameterstudien zur Definition geeigneter Standard-Parametersätze
EXPC	$N_{G\,Max}$	Auswirkung des Designparameters $N_{G\,Max}$ auf Laufzeit, Prognosegüte und Modellgröße bei zunehmender Größe der Lerndatenstichprobe
EXPE	Tuningparameter	
	1: Validierungsmethode	Entscheidung zwischen 10 facher Kreuz-Validierung und 10 bzw. 50 facher Wiederholung
	2: Randbedingungen	Erprobung der Randbedingungen der maximalen Modellgröße und der minimalen Vereinigungsrate
	3: Gütekriterium	Entscheidung zwischen Lernstichproben-Prognosegüte und Lernstichproben-Rank
	4: CSFS	Auswirkung einer permanenten Aktivierung der kontext-sensitiven Eingangsgrößenselektion

Die Tabelle 5.2 gibt einen Überblick über die jeweilige Fragestellung der durchgeführten Voruntersuchungen. Bis auf bei den Experimenten EXPC werden dabei die Benchmarks AUSTRALIAN, HEPATITIS, MUSHROOMS, SEG und TEMP verwendet. Bei den Experimenten EXPC kommen davon abweichend die Benchmarks DNA, LETTER und WAVE zur Anwendung, da eine größere Anzahl an Lerndatenpunkten benötigt wurde. Die bei den Experimenten EXPC gewonnenen

Erkenntnisse können allerdings keinesfalls in den späteren Benchmarkstudien genutzt werden, da dies eine Verletzung der Bedingungen der harten Validierung bedeuten würde!

In den folgenden Kapiteln sind die angestellten Vorüberlegungen und durchgeführten Voruntersuchungen dokumentiert. Die Ergebnisse der Benchmark-Vergleichsstudie finden sich im Kapitel 5.3.6.

5.3.1 Vorüberlegungen

Die sechs in Tabelle 5.3 aufgeführten freien Parameter des PNC 2-Algorithmus sind als Designparameter klassifiziert worden und werden – so nicht anders angegeben – in sämtlichen Experimenten wie angegeben eingestellt. Im folgenden werden die einzelnen Designparameter kurz erläutert und die Überlegungen, die zur Wahl des jeweiligen Wertes geführt haben, dargelegt.

Tabelle 5.3: Designparameter.

ρ	υ	$N_{G\,Max}$	N_{Bins}	δ_{Symb}	$W_{Kernel\,Min}$
1	2	250	10	0.3^{-1}	0

- Der Parameter ρ wird in der Minkowski-Metrik nach Gl. (3.15) verwendet. Die Einstellung auf den Wert 1 erschien am besten zu der beim PNC 2-Algorithmus verwendeten Hyperquader-Repräsentation zu passen. Andere Werte wurden versuchsweise verwendet. Für die in den folgenden Voruntersuchungen verwendeten Benchmarks ergaben sich dabei teilweise leicht bessere Prognosegüten. Da jedoch kein systematischer Zusammenhang feststellbar war, hätte dieser Parameter nicht als Designentscheidung festgelegt werden können, sondern hätte als Strategieparameter im Rahmen des Parametertunings auf den jeweils betrachteten Benchmark angepaßt werden müssen. In Anbetracht der eher geringen Auswirkungen dieses Parameters wurde dies nicht für sinnvoll gehalten.

- Der Parameter υ bestimmt bei Regressionsaufgaben die Form der aus den Hyperquadern generierten eingangsseitigen ZGF. Der Wert 2 bedingt gauss-förmig abfallende Flanken der ZGF, was als die plausibelste Form angesehen wurde. Ferner ergaben sich in ersten Versuchen mit dem Benchmark TEMP für den Wert 1 deutlich schlechtere Prognosegüten.

- Der Parameter $N_{G\,Max}$ dient, wie im Kapitel 3.3.3 beschrieben, zur Reduktion der Laufzeit bei größeren Lernstichproben. Dieser Parameter wurde zunächst ad hoc auf einen Wert von 250 eingestellt. In den Experimenten der Reihe EXPC in Kapitel 5.3.4 wird diese Wahl nachträglich hinterfragt und die Wirkung dieses Parameters systematisch untersucht.

- Der Parameter N_{Bins} wird, wie im Kapitel 2.8.2 beschrieben, zur äquidistanten oder äquifrequenten Diskretisierung kontinuierlicher Eingangsgrößen im Rahmen der Berechnung der Transinformation verwendet. Die in der Literatur üblicherweise verwendeten Werte variieren von einigen wenigen bis zu 20 und mehr Intervallen. Hier wird wie in [WD95, DKS95] der Wert 10 verwendet.

- Der Parameter δ_{Symb} wird, wie im Kapitel 2.8.1 beschrieben, zusammen mit der Spannweiten- und 4σ-Normierung verwendet, um den Normierungsfaktor δ_j für symbolische Eingangsgrößen festzulegen. Die Einstellung des Parameters erfolgte aufgrund der Untersuchungen in [WM97a]. Dort wurde für mehrere, auf dem NN-Algorithmus basierende,

Lernverfahren experimentell ein Wert von 0.3^{-1} als bezüglich der Prognosegüte günstigste Einstellung ermittelt. Man beachte, daß dieser Parameter bei einer d_{avr}-Normierung irrelevant ist, denn dort werden auch für symbolische Eingangsgrößen die Normierungsfaktoren aus der Lerndatenstichprobe berechnet. Bemerkenswerterweise ergeben sich bei den Benchmarks AUSTRALIAN und HEPATITIS im Mittel der für symbolische Eingangsgrößen berechneten Normierungsfaktoren mit $\overline{\delta} = 2.8$ bzw. $\overline{\delta} = 2.5$ jedoch ähnliche Werte.

- Der Parameter $W_{Kernel\,Min}$ dient zusammen mit dem Parameter W_{Kernel} zur Steuerung des Nachbarschaftsbereichs bei dem im Kapitel 3.3.5 beschriebenen Prognosemechanismus für Klassifikationsaufgaben. In Anbetracht der Zielvorgabe, die Anzahl der Strategieparameter möglichst gering zu halten, erscheint es nicht sinnvoll, zur Steuerung eines Strategieelementes zwei verschiedene Parameter zu verwenden. Daher wurde der Wert dieses Parameters zu 0 gewählt.

5.3.2 Experimente ExpA: Designentscheidungen

Versuchsbeschreibung In Tabelle 5.4 sind die freien Parameter des PNC 2-Algorithmus, die im Rahmen einer Designentscheidung festgelegt werden sollen, aufgeführt. Zur Berechnung der konfidenten Trefferquote wird dabei eine Irrtumswahrscheinlichkeit von $\alpha = 0.05$ verwendet.

Tabelle 5.4: Überblick über die zu treffenden Designentscheidungen.

Designentscheidung	**Alternativen**	**Kapitel**
Neuberechnung Clusterausgangsgrößenwerte	Aus, An*	3.3.5
Vereinigungstest	einfach, doppelt*	3.2.1
Normalisierungsmethode	Spannweite*, 4σ, d_{avr}	2.8.1
Glaubensgrade zur Regelbewertung	Aus MFC*, Aus MTB, TQ, konfidente TQ, laplace korrigierte TQ	2.2
Diskretisierungsmethode	äquidistant, äquifrequent*	2.8.2
Eingangsgrößengewichte	An*, Aus	2.8.2
CSFS	An*, Aus	3.3.4

Zeichenerklärung: * = Default-Alternative, MTB = Mass Tie Breaking, MFC = Most Frequent Class, TQ = Trefferquote

Theoretisch sollten alle Kombinationen der verschiedenen Alternativen betrachtet und die daraus resultierenden PNC 2-Varianten miteinander verglichen werden. Jedoch würde dies zu einem unvertretbar hohem Arbeitsaufwand führen. Daher wird für jede Designentscheidung eine Default-Alternative festgelegt und die Prognosegüte der daraus resultierenden PNC 2-Default-Variante mit den Prognosegüten derjenigen PNC 2-Varianten verglichen, die sich ergeben, wenn für jeweils genau eine Designentscheidung nicht die Default-Alternative gewählt wird.

Zur Festlegung der Default-Alternative wurden drei Auswahlkriterien berücksichtigt. Zum einen wurde versucht, aufgrund von Plausibilitätsüberlegungen eine vermutlich günstige Alternative zu identifizieren. Zum anderen wurde der üblichsten bzw. einfachsten Alternative oder der Alternative, die zu den kleineren generierten Modellen führen dürfte, der Vorzug gegeben.

Alle verbleibenden Parameter des PNC 2-Algorithmus sind eigentlich Strategie-Parameter, werden hier jedoch, wie in Tabelle 5.5 angegeben, fest eingestellt. Die Werte wurden nach ersten Versuchen mit den Entwicklungs-Benchmarks festgelegt. Diese feste Einstellung der Strategieparameter ohne vorherige systematische Untersuchungen ist problematisch. Besser wäre

Tabelle 5.5: Wahl der Strategieparameter bei den Experimenten ExpA.

Benchmark	N_{Int}	η	w_{COD}	p_{min}	W_{Kernel}	σ_{ZGF}
Temp	15	0.5	0	2	–	0.001
alle anderen	–	0.5	3	2	2	–

es, bereits hier Standard-Parametersätze festzusetzen und jeweils ein Parametertuning durchzuführen. Dies ist aus Gründen des damit verbundenen Arbeitsaufwandes unterlassen worden.

Die Prognosegüten werden mittels der Methode der N_R-fachen Wiederholung – bei Aufteilung in etwa gleich große Stichproben zum Lernen und Testen – ermittelt. Für Klassifikationsaufgaben wird der mittlere Klassifizierungsfehler MCE und für Regressionsaufgaben der mittlere absolute Fehler MAE betrachtet. Mittels des, im Anhang C beschriebenen, zweiseitigem t-Tests für gepaarte Ereignisse wird die Irrtumswahrscheinlichkeit, d.h. die Wahrscheinlichkeit der Null-Hypothese, daß der beobachtete Unterschied der mit der Default-Variante verglichenen Prognosegüten zufällig ist, berechnet. Tabelle 5.6 gibt einen Überblick über die Stichprobengröße N, die Anzahl und Art der Eingangsgrößen und die Anzahl der durchgeführten Wiederholungen N_R. Man beachte, daß beim Benchmark Mushrooms nur 1000 der insgesamt 8124 Datenpunkte verwendet werden.

Tabelle 5.6: Benchmarküberblick ExpA.

Benchmark	N	m	nom	cont	N_R
Australian	690	14	8	6	100
Hepatitis	155	19	13	6	100
Mushrooms	1000	22	22	0	100
Seg	2130	19	0	19	25
Temp	1126	3	0	3	25

Anmerkung: *nom* und *cont* bezeichnen die Anzahl nominaler bzw. kontinuierlicher Eingangsgrößen

Auswertung Die vollständigen Ergebnisse der Experimente sind in Anhang B aufgeführt. Unterschiede der mit der Default-Variante verglichenen Prognosegüte $\triangle Q_T[\%]$ werden ab einer Irrtumswahrscheinlichkeit P_{H0} von weniger als 10% als signifikant angesehen. Irrtumswahrscheinlichkeiten von weniger als 0.1% werden als 0% angegeben. Tabelle 5.7 gibt einen Überblick über die letztlich getroffenen Entscheidungen.

Tabelle 5.7: Getroffene Designentscheidungen.

Designentscheidung	**Wahl**
Neuberechnung Clusterausgangsgrößenwerte	An
Vereinigungstest	Doppelt
Normalisierungsmethode	Spannweite
Glaubensgrade zur Regelbewertung	Aus MFC
Diskretisierungsmethode	Äquifrequent
Eingangsgrößengewichte	An
Kontext-sensitive Eingangsgrößenselektion	*Strategieparameter*

Im folgenden werden die Ergebnisse für jede Designentscheidung einzeln diskutiert. Zur Vereinfachung werden im folgenden die Begriffe *besser* und *schlechter* in dem Sinne benutzt, daß die jeweilige alternative Designentscheidung beim PNC 2-Algorithmus bezüglich der betrachteten Benchmarks zu besseren bzw. schlechteren Prognosegüten der generierten Modelle führt.

- *Keine Neuberechnung der Clusterausgangsgrößenwerte*

Benchmark	$\triangle Q_T[\%]$	P_{H0}
TEMP	-51.7 %	0.0 %

  Diese Designentscheidung betrifft ausschließlich Regressionsaufgaben und kann daher nur anhand der Ergebnisse beim Benchmark TEMP getroffen werden. Die Prognosegüte ist mit Neuberechnung um etwa 50% besser. Die Neuberechnung wird daher ausgewählt.

- *Normierung*

  - *4σ-Normierung*

Benchmark	$\triangle Q_T[\%]$	P_{H0}
HEPATITIS	-3.1 %	4.5 %
SEG	-4.2 %	4.1 %
TEMP	1.0 %	0.2 %

  - *d_{avr}-Normierung*

Benchmark	$\triangle Q_T[\%]$	P_{H0}
AUSTRALIAN	-0.2 %	8.5 %
HEPATITIS	-4.1 %	2.2 %
MUSHROOMS	-9.6 %	0.8 %
SEG	-5.0 %	5.0 %
TEMP	3.5 %	0.2 %

  Die 4σ- und d_{avr}-Normierung sind meist signifikant schlechter als die Spannweiten-Normierung. Überraschend deutlich fällt das Ergebnis bei der d_{avr}-Normierung aus. In [HC99] wird dagegen über teilweise bessere Prognosegüten im Vergleich zur Spannweiten- und 4σ-Normierung berichtet; allerdings müßte, um die hier erzielten Ergebnisse direkt vergleichen zu können, der Wert des Normierungsfaktors δ_{Symb} zu 1 gewählt werden. Die Experimente werden daher versuchsweise mit dieser Einstellung bei den Benchmarks AUSTRALIAN und HEPATITIS wiederholt. Die sich dann beim Benchmark HEPATITIS für die Spannweiten-Normierung ergebende Prognosegüte ist um 5.1% ($P_{H0} = 0.6\%$) signifikant schlechter als vorher.

  Die Spannweiten-Normierung wird ausgewählt, denn sie ist besser als die beiden anderen betrachteten Normierungsmethoden, wenn der Wert des Normierungsfaktors δ_{Symb} zu 0.3^{-1} gewählt wird.

- *Einfacher Vereinigungstest*

Benchmark	$\triangle Q_T[\%]$	P_{H0}
AUSTRALIAN	0.5 %	0.0 %
HEPATITIS	-10.5 %	0.0 %
MUSHROOMS	-470.7 %	0.0 %
SEG	-13.0 %	0.0 %
TEMP	-35.3 %	0.2 %

Erwartungsgemäß führt der einfache Vereinigungstest zu einer stärkeren Vereinigungsrate und damit zu kleineren Modellen bezüglich der Clusteranzahl. Die durchschnittliche Trefferquote der Cluster ist dabei geringer, der Abdeckungsgrad, d.h. der Anteil der Testdatenpunkte, deren Eingangsraumposition innerhalb eines oder mehrerer Hyperquader der generierten Cluster liegt, dagegen größer. Der doppelte Vereinigungstest wird ausgewählt.

- *Glaubensgrade zur Regelbewertung*
  - *Aus Mass Tie Breaking* (MTB)

Benchmark	$\triangle Q_T[\%]$	P_{H0}
MUSHROOMS	-39.1 %	0.0 %

Beim MTB wird bei mehreren im gleichen Maße empfohlenen Ausgangsgrößenwerte derjenige gewählt, dessen empfehlende Regel die höchste Masse aufweist. Dies führt beim Benchmark MUSHROOMS zu einer um etwa 40% schlechteren Prognosegüte. Bei diesem Benchmark gibt es zwei Ausgangsklassen mit den unbedingten Auftrittswahrscheinlichkeiten von etwa 20% und 80%. Demnach scheint die bei der Default-Alternative *Aus MFC* verfolgte Strategie, im Zweifelsfall die am häufigsten auftretende Ausgangsklasse zu prognostizieren, die günstigere zu sein. Zudem müßte für jedes Cluster die Masse gespeichert werden, was zu einer Vergrößerung der Modelle und damit zu einem Verlust an Interpretierbarkeit führen würde.

  - *Laplace Korrigierte Trefferquote*

Benchmark	$\triangle Q_T[\%]$	P_{H0}
HEPATITIS	0.5 %	3.5 %
MUSHROOMS	-38.5 %	0.0 %
SEG	-1.7 %	9.1 %
TEMP	-2.0 %	0.0 %

  - *Konfidente Trefferquote*

Benchmark	$\triangle Q_T[\%]$	P_{H0}
HEPATITIS	1.6 %	1.9 %
MUSHROOMS	-44.8 %	0.0 %
TEMP	-2.2 %	0.0 %

– *Einfache Trefferquote*

Benchmark	$\triangle Q_T[\%]$	P_{H0}
SEG	1.6 %	1.6 %

Beim PNC 2-Algorithmus werden aufgrund dieser Ergebnisse keine Glaubensgrade zur Regelbewertung verwendet, da dies meist keinen oder aber einen negativen Einfluß auf die Prognosegüte hat. Allenfalls wäre die Verwendung der normalen Trefferquote in Betracht gekommen, da dies bei einem Benchmark zu Verbesserungen führt. Jedoch ist zu bedenken, daß die Glaubensgrade als zusätzliche Größe der Modelle gespeichert werden müßten.

- *Äquidistante Diskretisierung*

Benchmark	$\triangle Q_T[\%]$	P_{H0}
SEG	-8.1 %	1.0 %

Erwartungsgemäß hat die verwendete Diskretisierungsmethode bei vielen Benchmarks einen eher geringen Einfluß auf die Prognosegüten des PNC 2-Algorithmus, da sich diese Entscheidung nur indirekt über leicht andere Eingangsgrößengewichte auswirkt. Die äquifrequente Diskretisierung ist beim Benchmark SEG signifikant besser und zudem robuster bei eventuell in den Daten enthaltenen Ausreißern. Daher wird sie ausgewählt.

- *Keine Eingangsgrößengewichte*

Benchmark	$\triangle Q_T[\%]$	P_{H0}
AUSTRALIAN	-2.8 %	0.3 %
HEPATITIS	-2.8 %	9.1 %
MUSHROOMS	-11.9 %	0.8 %

In [WD95] wurden Gewichtsfaktoren in gleicher Weise wie hier zusammen mit NGE- und kNN-Algorithmen eingesetzt und dabei das gleiche Ergebnis erhalten: Die Verwendung von Gewichtsfaktoren führt zu denselben oder zu teils deutlich besseren, nie jedoch zu schlechteren Prognosegüten. Daher werden Gewichtsfaktoren beim PNC 2-Algorithmus prinzipiell verwendet.

- *Keine kontext-sensitive Eingangsgrößenselektion* (CSFS)

Benchmark	$\triangle Q_T[\%]$	P_{H0}
AUSTRALIAN	0.5 %	0.1 %
HEPATITIS	2.8 %	4.6 %

Bei allen Benchmarks ist die Prognosegüte ohne CSFS leicht besser, jedoch ist dieser Unterschied nur bei den Benchmarks AUSTRALIAN und HEPATITIS signifikant. Auf der anderen Seite ist bei Anwendung der CSFS die Größe der generierten Modelle etwa um den Faktor 3.6 geringer. Daher wird an dieser Stelle nicht endgültig über die Verwendung der CSFS entschieden, sondern diese Entscheidung im folgenden als Strategieparameter betrachtet und im Rahmen des Parametertunings festgelegt.

5.3.3 Experimente ExpB: Strategieparameter

Versuchsbeschreibung Um geeignete Standard-Parametersätze für den PNC 2-Algorithmus zu definieren wird für jeden Strategieparameter ein als sinnvoll erachteter Wertebereich festgelegt und innerhalb dieses Wertebereichs werden einige etwa gleichmäßig verteilte Werte ausgewählt. Der Wertebereich des Strategieparameters ω_{COD} wird an die Anzahl der Eingangsgrößen m gekoppelt zu

$$[0, \omega_{max}] \quad \text{mit} \quad \omega_{max} \approx \frac{m}{3} \; . \tag{5.7}$$

Innerhalb dieses Intervalls werden drei bis fünf etwa gleichmäßig verteilte Werte ausgewählt, wobei bei Benchmarks mit vielen Eingangsgrößen Werte im unteren Wertebereich vermieden werden, damit im mittleren bis oberen Wertebereich entsprechend feiner aufgelöst werden kann. Bei Benchmarks mit einem hohen Anteil an nominalen Eingangsgrößen kann ferner die obere Wertebereichsgrenze reduziert werden. Die verwendeten Standard-Parametersätze sind in den Tabellen 5.8 und 5.9 aufgeführt.

Tabelle 5.8: Standard-Parametersätze.

N_{Int}	η	ω_{COD}	p_{min}	W_{Kernel}	CSFS	σ_{ZGF}
{10, 15, 20}	{0, 0.5, 1}	*siehe Text*	{7, 4, 2}	{1, 2, 3}	{0, 1}	{0.01, 0.001, 0.0001}

Für alle möglichen Kombinationen dieser Werte wird mittels N_R-facher Wiederholung – bei Aufteilung in etwa gleich große Stichproben zum Lernen und zum Testen – die Prognosegüte ermittelt. Für Klassifikationsaufgaben wird der mittlere Klassifizierungsfehler MCE und für Regressionsaufgaben der mittlere absolute Fehler MAE betrachtet. Die Parametersätze werden dann nach der resultierenden Prognosegüte sortiert. Der Parametersatz der die beste Prognosegüte erzielt wird als *bester* Parametersatz bezeichnet. Tabelle 5.9 gibt einen Überblick über die Stichprobengröße N, die Anzahl und Art der Eingangsgrößen, die Anzahl der durchgeführten Wiederholungen N_R und die für den Strategieparameter w_{COD} verwendeten Werte. Man beachte, daß beim Benchmark MUSHROOMS nur 1000 der insgesamt 8124 Datenpunkte verwendet wurden.

Tabelle 5.9: Benchmarküberblick ExpB.

Benchmark	N	m	nom	cont	N_R	ω_{COD}
AUSTRALIAN	690	14	8	6	400	{0, 1.5, 3, 4.5}
HEPATITIS	155	19	13	6	100	{0, 2, 4, 6}
MUSHROOMS	1000	22	22	0	1500	{0, 1.5, 3, 4.5}
SEG	2130	19	0	19	150	{1, 2.5, 4, 5.5}
TEMP	1126	3	0	3	100	{0, 0.5, 1}

Anmerkung: *nom* und *cont* bezeichnen die Anzahl nominaler bzw. kontinuierlicher Eingangsgrößen

Der Wert des Experimentparameters N_R wird für jeden Benchmark so eingestellt, daß sich auch bei Wahl von $\frac{N_R}{2}$ dieselben besten Parametersätze ergeben. Damit wird dem im folgenden beschriebenen Problem des Overfittings entgegengewirkt: Die ermittelten Prognosegüten $\widehat{Q}$ sind lediglich Schätzwerte der tatsächlichen Prognosegüten Q^* und werden mit einer bestimmten, aber unbekannten Verteilung um den tatsächlichen Wert streuen. Durch die Sortierung nach

der Prognosegüte wird ein *Bias* erzeugt, der bedingt, daß als bester Parametersatz bevorzugt ein Parametersatz ausgewählt wird, für den ein zu optimistischer Gütewert ermittelt wurde.

Auswertung In Tabelle 5.10 sind für jeden Benchmark für beide möglichen Werte des Parameters *CSFS* die jeweils besten, sowie die jeweils schlechtesten Parametersätze aufgeführt. Für die dazugehörigen Modelle ist die erzielte Prognosegüte $\widehat{Q}_T$, die Anzahl der generierten Cluster K bzw. K_{Red} und die über alle Cluster ermittelte durschnittliche Anzahl betrachteter Eingangsgrößen $\varnothing\ m$ angegeben. Die reduzierte Clusteranzahl K_{Red} ergibt sich, indem alle Cluster eliminiert werden, deren Masse den Schwellwert p_{min} nicht überschreitet.

Tabelle 5.10: Ergebnisse der Experimente ExpB.

Benchmark	N_{Int}	η	w_{COD}	p_{min}	W_{Kernel}	σ_{ZGF}	CSFS	$\widehat{Q}_T$	K	K_{Red}	$\varnothing\ m$
Australian	–	1	4.5	7	1	–	1	14.53 %	168.3	3.8	5.2
	–	1	4.5	7	2	–	0	14.59 %	168.3	3.8	14.0
	–	0	4.5	7	1	–	1	33.40 %	264.0	1.9	5.8
Hepatitis	–	0	4	2	3	–	1	17.88 %	19.8	9.3	4.9
	–	0.5	0	7	3	–	0	19.62 %	5.6	2.1	19.0
	–	0	6	7	1	–	1	29.58 %	32.7	1.4	5.2
Mushrooms	–	0	1.5	2	1	–	0	0.36 %	5.7	5.3	22.0
	–	0	3	2	1	–	1	0.47 %	8.1	6.9	4.6
	–	1	0	7	3	–	1	5.91 %	2.9	2.7	3.9
Seg	–	0	1	2	3	–	0	4.19 %	32.3	26.3	19.0
	–	0	1	2	3	–	1	4.33 %	32.3	26.3	2.5
	–	0	5.5	7	1	–	0	11.73 %	147.6	30.7	19.0
Temp	10	0	0.5	2	–	0.0001	1	1.159	44.2	29.5	1.4
	10	1	0.5	2	–	0.001	0	1.269	28.1	19.4	3.0
	20	0	0.5	7	–	0.001	1	2.903	72.4	9.1	1.5

Über die Auswirkungen der verschiedenen Strategieparameter lassen sich die folgenden Aussagen ableiten:

- η: Eine Wahl von $\eta \to 1$ führt zu einer stärkeren Vereinigungsrate und damit zu einer geringeren Clusteranzahl K. Die Masse der Cluster wird größer sein, womit der Reduktionseffekt durch den Parameter p_{min} geringer ausfallen wird.

- w_{COD}: Die Wahl von $w_{COD} \to \omega_{max}$ führt zu einer geringeren Vereinigungsrate, die bei Wahl eines zu großen Wertes dazu führt, daß kaum noch Datenpunkte miteinander vereinigt werden und die Clusteranzahl K fast der ursprünglichen Größe der Lernstichprobe entspricht. Beim Benchmark Australian tritt dadurch Instabilität des Parametersatzes in dem Sinne auf, daß kleine Änderungen der Parameter signifikant schlechtere Prognosegüten zur Folge haben. Der beste und der schlechteste Parametersatz unterscheiden sich dort lediglich im Wert des Parameters η. Beim Parametertuning sollte daher eine minimale Vereinigungsrate als Randbedingung beachtet werden. Dies dürfte auch zu besseren Laufzeiten des Pnc 2-Algorithmus führen, da bei einer geringen Vereinigungsrate mehr Vereinigungstests durchgeführt werden müssen. Insgesamt scheint der für den Parameter vorgesehene Wertebereich ausreichend zu sein, da die in den jeweiligen besten Parametersätzen aufgetretenen Werte meist innerhalb des Wertebereichs liegen.

- p_{min}: Eine Wahl von $p_{min} \to 7$ bewirkt eine stärkere Elimination von Clustern und damit kleinere Modelle. Diese Wirkung ist insbesondere in Verbindung mit größeren Werten

für w_{COD} zu beobachten, weil dann aufgrund der geringeren Vereinigungsrate sehr viele Cluster eine kleine Masse aufweisen und eliminiert werden.

- W_{Kernel}: Erwartungsgemäß führt bei einigen Benchmarks die Wahl von $W_{Kernel} > 1$, bei anderen die Wahl von $W_{Kernel} = 1$, zu besseren Prognosegüten. Damit verhält sich dieser Parameter ähnlich wie der Parameter k des kNN-Algorithmus, bei dem für einige Benchmarks ebenfalls eine Wahl von $k = 1$ am günstigsten ist.

- *CSFS*: Teilweise führt die Anwendung der CSFS zu besseren Prognosegüten. Dies zeigt, daß die in Rahmen der Experimente ExpA getroffene Entscheidung, die Anwendung der CSFS als Strategieparameter zu betrachten, sinnvoll ist.

5.3.4 Experimente ExpC: Laufzeit und der Parameter $N_{G\,Max}$

Versuchsbeschreibung Mittels der Experimente ExpC1 wird die Wirkung des Designparameters $N_{G\,Max}$ näher untersucht und die zunächst ad hoc getroffene Einstellung auf den Wert 250 überprüft. Der Parameter $N_{G\,Max}$ ist eine Schwellwert für die Anzahl der zu clusternden Datenpunkte einer Ausgangsklasse und bedingt bei Überschreitung die Aufteilung der Datenpunkte in mehrere Sub-Gruppen, die dann nacheinander bearbeitet und abschließend wieder zusammengeführt werden. Die genaue Wirkungsweise ist im Kapitel 3.3.3 beschrieben. Mittels der Experimente ExpC2 wird die Erhöhung der Laufzeit und die Entwicklung der Prognosegüte und Modellgröße mit zunehmender Anzahl an Lerndatenpunkten dokumentiert. Anders als in den anderen Voruntersuchungen werden hier die Benchmarks Dna, Letter und Wave verwendet, da Benchmarks mit einer größeren Anzahl an Datenpunkten benötigt werden. Die gewonnenen Erkenntnisse können allerdings nicht in den späteren Benchmark-Vergleichsstudien genutzt werden, da dies eine Verletzung der Bedingungen der harten Validierung bedeuten würde! Für die Strategieparameter werden die in Tabelle 5.11 angegebenen Werte fest eingestellt. Die Experimente wurden auf einem Pentium III 700 MHz (Letter und Wave) bzw. auf einem Amd Duron 1 GHz (Dna) durchgeführt.

Tabelle 5.11: Wahl der Strategieparameter bei den Experimenten ExpC.

η	ω_{COD}	p_{min}	W_{Kernel}	CSFS
0.5	3	1	2	0 *oder** 1

* Gewählt wird für den jeweiligen Benchmark die Einstellung, die zu den besseren Prognosegüten führt.

Bei den Experimenten ExpC1 wird der Parameter $N_{G\,Max}$ auf die Werte 50, 75, 100, 125, 250, 500 und 1000 eingestellt und jeweils mittels N_R-facher Wiederholung die Prognosegüte ermittelt. Die Aufteilung in Lern- und Teststichproben und die Anzahl der durchgeführten Wiederholungen ist Tabelle 5.12 zu entnehmen. Ferner ist die anhand der Lernstichprobengröße zu erwartende maximale Anzahl $\widetilde{N}_{G\,Max}$ an Datenpunkten mit derselben Ausgangsklasse angegeben. Eine Erhöhung von $N_{G\,Max}$ über den Wert von $\widetilde{N}_{G\,Max}$ hinaus sollte keine Wirkung mehr haben.

Bei den Experimenten ExpC2 wird für die feste Einstellung des Parameters $N_{G\,Max}$ zu 125 und zu ∞ jeweils mittels N_R-facher Wiederholung die Prognosegüte ermittelt. Die Anzahl der Datenpunkte in der Lernstichprobe N_L wird dabei schrittweise erhöht. Die übrigen Datenpunkte werden jeweils zum Testen verwendet. Man beachte, daß mit zunehmender Größe der Lernstichprobe gleichzeitig weniger Datenpunkte zum Ermitteln der Prognosegüte verwendet

Tabelle 5.12: Benchmarküberblick ExpC1.

Benchmark	N	m	nom	cont	N_L	N_T	N_R	$\tilde{N}_{G\,Max}$
Dna	3175	60	60	0	1700	1475	200	850
Letter	20000	16	0	16	16000	4000	75	650
Wave	5000	21	0	21	2550	2450	50	850

Anmerkung: *nom* und *cont* bezeichnen die Anzahl nominaler bzw. kontinuierlicher Eingangsgrößen

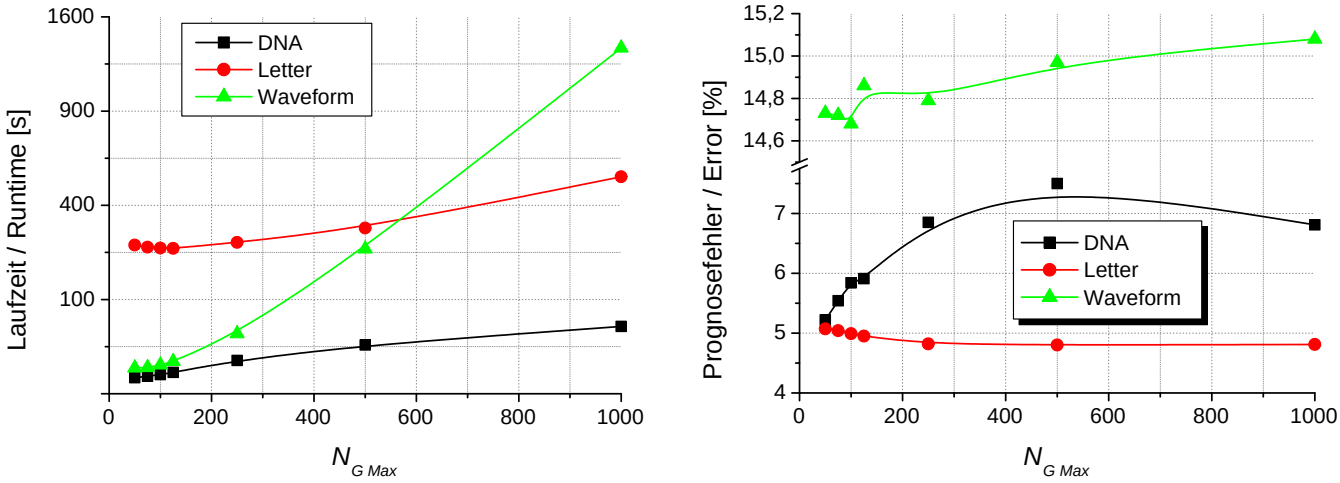
werden, wodurch die Genauigkeit der ermittelten Schätzwerte abnimmt. Für $N_{G\,Max} = 125$ wird der in Tabelle 5.13 angegebene Wert für die Anzahl der Wiederholungen N_R verwendet. Für $N_{G\,Max} = \infty$ wurden aufgrund der stark ansteigenden Rechenzeiten nur jeweils 2 Wiederholungen durchgeführt.

Tabelle 5.13: Benchmarküberblick ExpC2.

Benchmark	N	m	nom	cont	N_R	$N_{L\ 100\%}$
Dna	3175	60	60	0	40	3060
Letter	20000	16	0	16	10	18000
Wave	5000	21	0	21	10	4590

Anmerkung: *nom* und *cont* bezeichnen die Anzahl nominaler bzw. kontinuierlicher Eingangsgrößen

Auswertung In den Abbildungen 5.1 und 5.2 ist für alle drei Benchmarks jeweils die Laufzeit, die resultierende Modellgröße und die erzielte Prognosegüte über den Wert des Parameters $N_{G\,Max}$ aufgetragen.



(a) **(b)**

Abbildung 5.1: Zeitbedarf (quadratisch) (a) und Prognosefehler $\widehat{Q}_T$ (b) beim Lernen eines Modells mit dem Pnc 2-Algorithmus in Abhängigkeit des Parameters $N_{G\,Max}$.

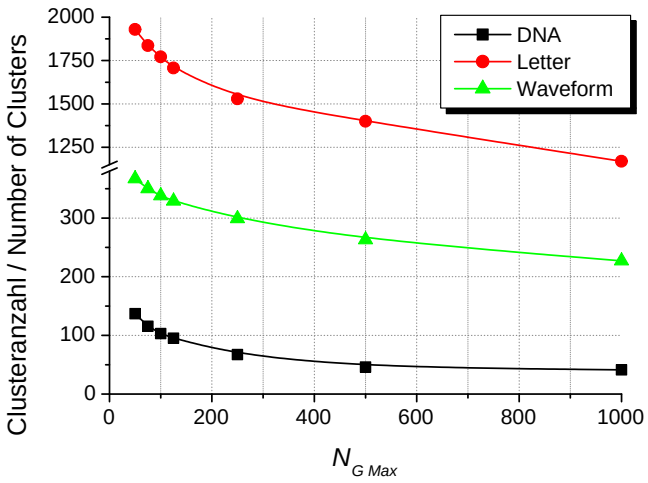


Abbildung 5.2: Clusteranzahl eines mit dem PNC 2-Algorithmus gelernten Modells in Abhängigkeit des Parameters $N_{G\,Max}$.

Erwartungsgemäß vergrößert sich die Laufzeit und verringert sich die Modellgröße mit zunehmenden Werten des Parameters $N_{G\,Max}$. Überraschenderweise wird jedoch nur beim Benchmark LETTER die erzielte Prognosegüte dabei ebenfalls besser, wohingegen sie sich bei den Benchmarks DNA und WAVE verschlechtert. Dies weist darauf hin, daß es sich bei der Bildung der Sub-Gruppen de facto um eine Kombination mehrerer – auf Teilmengen der Lerndaten gelernter – Modelle handelt, was, wie im folgenden Abschnitt diskutiert werden wird, zu einer Erhöhung der Prognosegüte führen kann. Die Wahl eines Wertes von 125 scheint ein guter Kompromiß zwischen erzielbarer Prognosegüte auf der einen und benötigter Laufzeit und resultierender Modellgröße auf der anderen Seite zu sein. Um jedoch nicht die Bedingungen der harten Validierung zu verletzen, wird weiterhin der bisherige Wert von 250 verwendet.

In den Abbildungen 5.3 und 5.4 sind, für den kleineren oder beide Werte des Parameters $N_{G\,Max}$ und für alle drei Benchmarks, jeweils die Laufzeit, die erzielte Prognosegüte und die resultierende Modellgröße über die Anzahl der Lerndatenpunkte dargestellt. Dabei entsprechen 100% der Lerndatenpunkte den für den jeweiligen Benchmark in Tabelle 5.13 in der Spalte $N_{L\ 100\%}$ angegeben Wert. Wie zu sehen ist, steigt die Laufzeit für die Wahl von $N_{G\,Max} = 125$ in etwa quadratisch mit der Anzahl der Lerndatenpunkte an.

Diskussion Ein in letzter Zeit mit steigendem Interesse verfolgter Forschungszweig untersucht die Fragestellung, ob durch die Kombination mehrerer Modelle eine Verbesserung der Prognosegüte erzielt werden kann [KHDM98, Die98, BK99]. Dafür werden für eine gegebene Lernstichprobe mehrere Modelle gelernt und die Prognosewerte der einzelnen Modelle werden typischerweise mit einem gewichteten oder ungewichteten Mehrheitsentscheid ausgewertet. Dies führt dann zu einer Erhöhung der Prognosegüte, wenn für die einzelnen Modelle die Fehler untereinander unkorreliert sind und die Fehlerraten unter 50% liegen [Die98].

Um Modelle, die zu unkorrelierten Prognosefehlern führen, zu generieren, ist es zunächst notwendig, daß die einzelnen Modelle unterschiedlich sind, d.h. zumindest in einigen Teilen des Eingangsraumes unterschiedliche Ausgangsgrößenwerte prognostizieren. Daher werden die einzelnen Modelle entweder mit verschiedenen Lernverfahren gelernt, oder aber es wird zwar dasselbe Lernverfahren verwendet, dieses jedoch unterschiedlich parametriert oder auf mehreren,

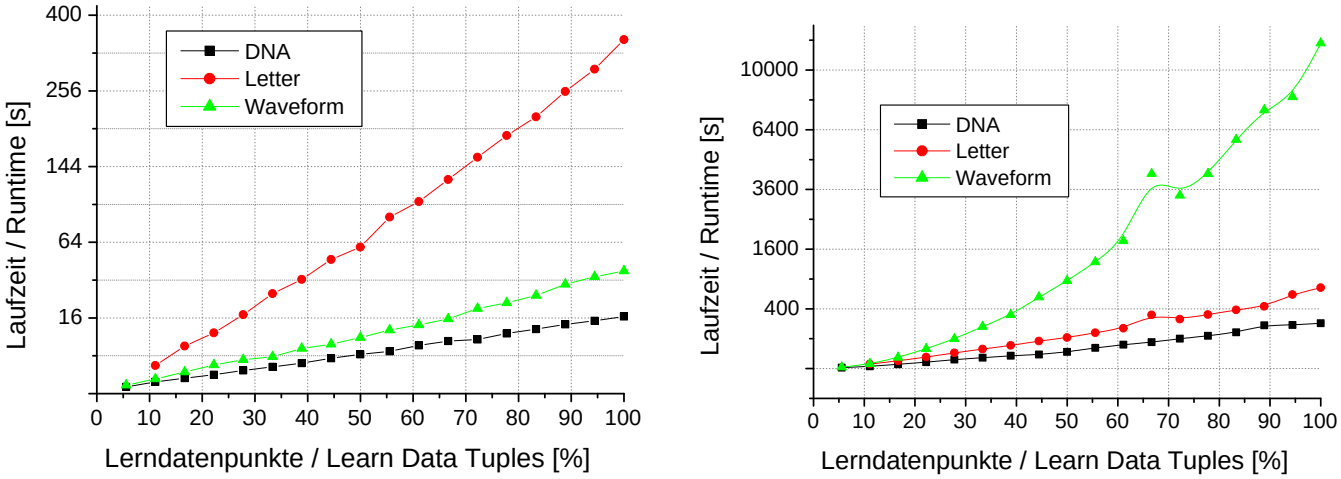


Abbildung 5.3: Zeitbedarf (quadratisch) zum Lernen eines Modells mit dem PNC 2-Algorithmus abhängig von der Lernstichprobengröße für $N_{G\,Max} = 125$ (a) und $N_{G\,Max} = \infty$ (b).

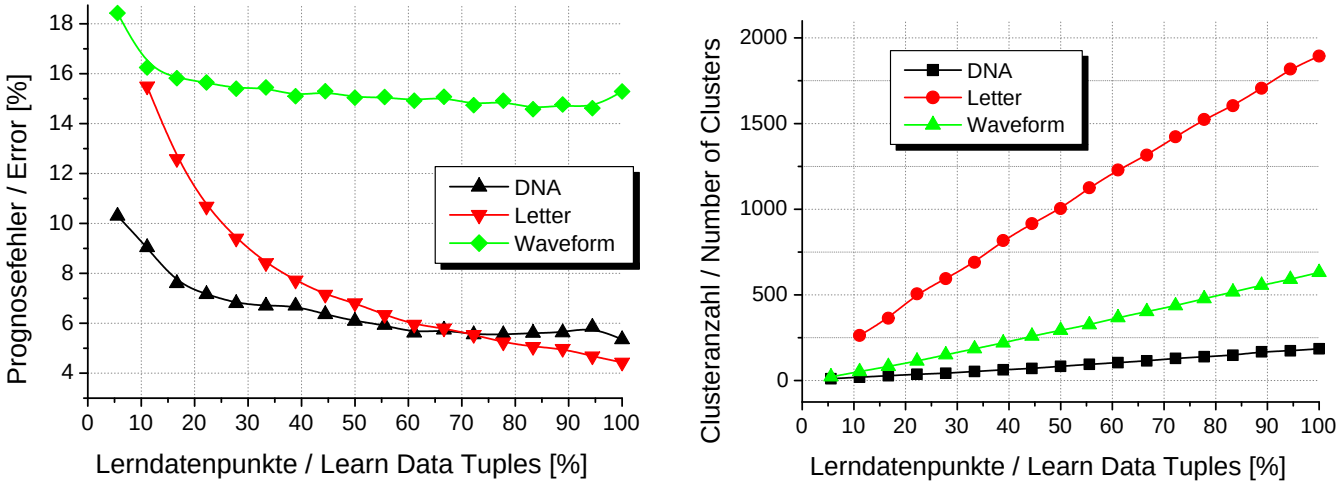


Abbildung 5.4: Prognosefehler $\widehat{Q}_T$ und Clusteranzahl der mit dem PNC 2-Algorithmus gelernten Modelle abhängig von der Lernstichprobengröße für $N_{G\,Max} = 125$.

aus der Lernstichprobe gebildeten, kleineren Stichproben eingesetzt[3]. Im zweiten Falle ist eine weitere Bedingung für den erfolgreichen Einsatz, daß das verwendete Lernverfahren instabil ist, d.h. kleine Änderungen der Lernstichprobe müssen ein stark unterschiedliches Prognosemodell zur Folge haben. Regelinduktionsverfahren, Entscheidungsbäume und neuronale Netze sind in diesem Sinne instabil, während lineare Regressionsmodelle oder Nearest-Neighbor-Verfahren generell stabil sind [Die98].

[3]In [BK99] werden verschiedene sogenannte *Bagging*- und *Boosting*-Verfahren zur Aufteilung der Lernstichprobe in mehrere kleinere Stichproben beschrieben. In einer umfangreichen Vergleichsstudie werden die Vor- und Nachteile der beschriebenen Aufteilungsvarianten in Kombination mit mehreren Lernverfahren untersucht.

Die zur Laufzeitreduktion eingesetzte Aufteilung der Lernstichprobe stellt offensichtlich eine Kombination mehrerer Modelle dar, die sich bei zwei der hier betrachteten Benchmarks positiv auf die Prognosegüte auswirkt.

5.3.5 Experimente ExpE: Tuningparameter

Mittels der Experimente ExpE1 bis ExpE4 wird die genaue Vorgehensweise des Parametertunings festgelegt. Zur Ermittlung der Prognosegüte wird bei allen Experimenten eine N_R-fache Wiederholung – bei Aufteilung in etwa gleich große Stichproben zum Lernen und zum Testen – verwendet.

Tabelle 5.14: Wahl von N_R für die Experimente ExpE.

Australian	Hepatitis	Mushrooms	Seg	Temp
100	200	50	100	100

Tabelle 5.14 gibt Auskunft über die Anzahl der für den jeweiligen Benchmark durchgeführten Wiederholungen N_R. Für Klassifikationsaufgaben wird der mittlere Klassifizierungsfehler MCE und für Regressionsaufgaben der mittlere absolute Fehler MAE betrachtet. Es werden die im Rahmen der Experimente ExpB definierten und erprobten Standard-Parametersätze, wie in den Tabellen 5.8 und 5.9 angegeben, verwendet. Die einzelnen Experimente werden im folgenden beschrieben und ausgewertet. Die daraufhin gewählten Tuningparameter sind in Tabelle 5.15 dargestellt. Aus Platzgründen sind die Ergebnisse der einzelnen Experimente nur zusammenfassend dokumentiert.

Tabelle 5.15: Gewählte Tuningparameter.

Auswahlkriterium für den *besten* Parametersatz	Lernstichproben-Prognosegüte MCE/MAE
Ermittlung Lernstichproben-Prognosegüte	50-fache Wiederholung*
$N_{L\,Tune}$ bzw. $N_{T\,Tune}$	0.8 N_L bzw. 0.2 N_L *
min. Vereinigungsrate	50 %
max. Modellgröße	10 %

* Bei größeren Stichproben wird zur Reduktion des Rechenaufwands eine geringere Wiederholungsanzahl und Tuning-Lern- und Tuning-Teststichprobengröße verwendet.

Experimente ExpE1

Zunächst wird beim Parametertuning derjenige Parametersatz ausgewählt, der auf der jeweiligen Lernstichprobe die beste Lernstichproben-Prognosegüte erzielt. Im Rahmen der Experimente ExpE1 werden drei verschiedene Varianten zur Ermittlung dieser Prognosegüte betrachtet: Zum einen die 10-fache Kreuz-Validierung und zum anderen eine 10- bzw. 50-fache Wiederholung bei Aufteilung der jeweiligen Lernstichprobe in einem Verhältnis von 80 zu 20.

Bezüglich der damit erzielten Prognosegüten sind nur beim Benchmark Seg signifikante Unterschiede vorhanden: Dort führt die Verwendung der 50-fachen Wiederholung zu einer Prognosegüte von 4.3%, wohingegen mittels einer 10-fachen Kreuz-Validierung nur eine Prognosegüte von 4.64% erzielt werden kann. Dieser Unterschied ist – bei Bewertung mit dem in Anhang C beschriebenen t-Test für gepaarte Ereignisse bei zweiseitiger Fragestellung – signifikant mit

einer Irrtumswahrscheinlichkeit von 0.1%. Zudem sind die generierten Modelle mit einer ursprünglichen Clusteranzahl K von 55 zu 36 und einer reduzierten Clusteranzahl K_{Red} von 34 zu 27 etwas kleiner.

Es ist zu vermuten, daß bei den kleineren Benchmarks, die nur einige Hundert Datenpunkte umfassen, eine 10-fache Kreuz-Validierung bzw. eine 10-fache Wiederholung vollkommen ausreichend ist und deshalb mit einer 50-fachen Wiederholung auch keine Verbesserung erzielt werden kann. Hingegen ist es bei dem mit 2130 Datenpunkten größeren Benchmark SEG vermutlich möglich, mittels einer 50-fachen Wiederholung die Lernstichproben-Prognosegüte der einzelnen Parametersätze genauer zu ermitteln und daher teilweise einen günstigeren Parametersatz auszuwählen, was sich positiv auf die Prognosegüte auswirkt. In allen folgenden Experimenten wird daher – so nicht anders angegeben – eine 50-fache Wiederholung zur Ermittlung der Lernstichproben-Prognosegüte verwendet.

Experimente ExpE2

Motiviert durch die in den Experimenten der Reihe ExpB beim Benchmark AUSTRALIAN gemachte Beobachtung, daß zu hohe Werte des Parameters ω_{COD} zu instabilen Parametersätzen führen können, werden die minimale Vereinigungsrate und die maximale Modellgröße als zusätzliche Randbedingungen eingeführt. Die Vereinigungsrate berechnet sich als Quotient aus der Anzahl der Cluster K und der Anzahl der Lerndatenpunkte N. Die Modellgröße ergibt sich als Produkt der reduzierten Anzahl der Cluster K_{Red} und der – über alle verbliebenen Cluster ermittelten – mittleren Anzahl betrachteter Eingangsgrößen $\varnothing m$. Wenn bei einer der durchgeführten Wiederholungen zur Ermittlung der Lernstichproben-Prognosegüte das generierte Modell eine oder beide Randbedingungen nicht erfüllt, wird der betreffende Parametersatz bei der Auswahl ignoriert. Die Wirkungsweise der Randbedingungen wird am Benchmark AUSTRALIAN untersucht. Die dabei verwendeten Werte und die damit erzielten Ergebnisse sind in Tabelle 5.16 dargestellt. T_{Tune} bezeichnet dabei die zur Durchführung des Parametertunings benötigte Rechenzeit[4].

Tabelle 5.16: Ergebnisse der Experimente ExpE2.

max. Modellgröße	min. Vereinigungsrate	$\widehat{Q}_T$	K	K_{Red}	$\varnothing\ m$	T_{Tune}
100 %	100 %	15.01 %	95.2	9.2	10.7	14m35s
10 %	100 %	14.90 %	88.2	5.8	8.5	13m33s
10 %	50 %	14.94 %	53.0	5.7	9.1	3m21s

Um Rechenzeit bei Durchführung des Parametertunings zu sparen, sollten Simulationen dann nicht mehr durchgeführt werden, wenn davon auszugehen ist, daß der betreffende Parametersatz ignoriert werden wird. Wenn für einen Parametersatz a bereits feststeht, daß er die Randbedingungen verletzt, so wird hier davon ausgegangen, daß jeder Parametersatz b für den gilt

$$(\eta_a \geq \eta_b) \ \wedge \ (\omega_{COD\,a} \leq \omega_{COD\,b}) \ \wedge \ (p_{min\,a} \geq p_{min\,b}) \ \wedge \ (CSFS_a \geq CSFS_b), \tag{5.8}$$

ebenfalls ignoriert werden wird[5].

[4]Simuliert auf einem System mit einem PENTIUM III 700 MHz.

[5]Dies muß im allgemeinen nicht immer der Fall sein. Man bedenke, daß bei einem höheren Wert des Parameters ω_{COD} durch die dann geringere Vereinigungsrate viel mehr Cluster eine zu kleine Masse aufweisen können und reduziert würden. Dieser Fall sollte jedoch ohnehin vermieden werden.

Die Simulationen für die einzelnen Parametersätze sollten so durchgeführt werden, daß zunächst diejenigen Parametersätze betrachtet werden, die zu den kleineren Modellen führen werden.

Die Wahl der minimalen Vereinigungsrate zu 50% und der maximalen Modellgröße zu 10% führt zu sowohl deutlich kleineren Modellen, als auch zu einer Reduktion der für das Parametertuning benötigten Rechenzeit um etwa 80%. Damit scheint es sich bei diesen beiden Werten um eine günstige Wahl zu handeln, die in allen folgenden Experimenten verwendet wird.

Experimente ExpE3 Bisher wurde die Lernstichproben-Prognosegüte als Auswahlkriterium für den besten Parametersatz verwendet. Versuchsweise wird stattdessen der mittlere Lernstichproben-Rank ausprobiert. Zu dessen Ermittlung wird jeweils eine 10-fache Wiederholung durchgeführt. Da sich hierbei bei keinem der Benchmarks signifikante Unterschiede feststellen lassen, wird an der ursprünglichen Vorgehensweise festgehalten.

Experimente ExpE4 Die Entscheidung über die Anwendung der CSFS wird als Strategieparameter im Rahmen des Parametertunings eingestellt. Beim Benchmark TEMP wird in über 95% aller Fälle die CSFS aktiviert, bei anderen Benchmarks teilweise deutlich seltener. Die CSFS wird versuchsweise permanent aktiviert. Dies führt einerseits zu einer etwa 5% schlechteren Prognosegüte, andererseits wird jedoch die Reduktion der Modellgröße vom Faktor 1.8 auf den Faktor 4.6 gesteigert. In Anbetracht der geringeren Prognosegüte wird an der ursprünglichen Vorgehensweise festgehalten.

5.3.6 Benchmark-Vergleichsstudien

Der PNC 2-Algorithmus wird in diesem Kapitel experimentell mit dem NGE-, dem RISE-, dem kNN- und dem FR-Algorithmus verglichen. Der NGE- und der RISE-Algorithmus sind im Kapitel 3.4 und der kNN- und der FR-Algorithmus im Kapitel 2.3 bzw. 2.2.1 näher beschrieben. Zur Ermittlung der Ergebnisse des PNC 2-Algorithmus wird die, im Kapitel 5.1 beschriebene, harte Validierung durchgeführt. Für die anderen Algorithmen werden in der Literatur angegebene Ergebnisse herangezogen und die dort jeweils verwendete Aufteilung in Lern- und Teststichproben wird für die mit dem PNC 2-Algorithmus durchgeführten Untersuchungen übernommen. Es sei angemerkt, daß dies nur eine sub-optimale Vorgehensweise darstellt. Optimaler wäre es, für alle Algorithmen die Ergebnisse mit eigenen Simulationen zu ermitteln. Damit wäre gewährleistet, daß genau die gleichen Lern- und Teststichproben verwendet würden, und es wäre möglich, die Unterschiede in den erzielten Prognosegüten mittels eines t-Tests für gepaarte Ereignisse auf ihre Signifikanz hin zu bewerten. Die Anwendung des einfachen t-Tests mit den gemittelten Prognosegüten ermöglicht hingegen – aufgrund der vom Betrag her geringen Unterschiede und der dazu relativ großen Standardabweichung – kaum noch eine Aussage. Allerdings erfordert die Durchführung der optimaleren Vorgehensweise einen nicht unerheblich höheren Arbeitsaufwand, denn alle Algorithmen müßten implementiert oder anderweitig beschafft werden. Daher wurde darauf verzichtet.

Zur Ermittlung der Prognosegüte wird eine N_R-fache Wiederholung verwendet. Die dabei für den Vergleich mit den verschiedenen Algorithmen verwendete Aufteilung in Lern- und Teststichproben und die für den Strategieparameter ω_{COD} verwendeten Standard-Werte sind Tabelle 5.17 zu entnehmen. Abweichend von der Vorgehensweise in [WD95] werden, um die Genauigkeit der Ergebnisse zu erhöhen, bei den Benchmarks WAVE und WAVE+NOISE statt der Wahl von $N_T = 100$ alle restlichen verfügbaren Datenpunkte zum Testen verwendet. Des weiteren werden die in den Voruntersuchungen ermittelten Einstellungen für die freien Parameter

des PNC 2-Algorithmus verwendet. Dies sind die Designparameter nach Tabelle 5.3, die Designentscheidungen nach Tabelle 5.7, die Standard-Parametersätze nach Tabelle 5.8 und die Tuningparameter nach Tabelle 5.15. Um den Rechenaufwand zu begrenzen wird – abweichend von Tabelle 5.15 – bei den Benchmarks DNA, LETTER, MUSHROOMS, SAT und KIN32FM die Lernstichprobe in zwei etwa gleich große Tuning-Lern- und Teststichproben geteilt[6] und die Anzahl der durchgeführten Wiederholungen auf 5 für den Benchmark KIN32FM und auf 10 für die restlichen Benchmarks reduziert.

Eventuell in einer der Stichproben vorhandene fehlende Eingangsgrößenwerte werden zur Zeit im PNC 2-Algorithmus nicht gesondert berücksichtigt. Stattdessen werden fehlende Eingangsgrößenwerte in einem manuellen Vorverarbeitungsschritt bei nominalen Eingangsgrößen durch ein zusätzliches Symbol und bei kontinuierlichen Eingangsgrößen durch einen außerhalb des Wertebereichs der betreffenden Größe liegenden Wert ersetzt. Dies ist eine einfache, aber bei kontinuierlichen Eingangsgrößen ungünstige Lösung, da damit der komponentenweise Abstand eines fehlenden Wertes zu verschiedenen möglichen bekannten Werten unterschiedlich sein wird.

Bei den in den Tabellen 5.19 bis 5.21 angegebenen Ergebnissen bezeichnet K die Anzahl der Cluster vor, und K_{Red} jene nach der Reduktion aller Cluster mit einer Masse von nicht mehr als p_{min}; $\varnothing\ m$ ist die durchschnittliche Anzahl betrachteter Eingangsgrößen je Cluster. Als Prognosegüte wird bei Klassifikationsaufgaben der mittlere Klassifizierungsfehler MCE und bei Regressionsaufgaben der mittlere absolute Fehler MAE bezüglich der Teststichproben angegeben. Wenn verfügbar wird zusätzlich noch die Standardabweichung der Prognosegüte – mit dem Zeichen $\pm$ gekennzeichnet – mit angegeben.

Tabelle 5.17: Verwendete Standard-Werte des Strategieparameters ω_{COD} und Aufteilung in Lern- und Teststichproben für den Vergleich mit dem jeweiligen Algorithmus.

		[WD95]		RISE		FR	
Benchmark	ω_{COD}	N_L	N_T	N_L	N_T	N_L	N_T
CHESS	{0, 2, 4, 6}	–	–	2109	1087	–	–
DNA	{2, 4, 6, 8}	–	–	2117	1058	1588	1587
CLEVELAND	{1, 2, 3, 4, 5}	212	91	202	101	–	–
HEPATITIS	{0, 2, 4, 6}	–	–	103	52	–	–
IRIS	{0, 0.5, 1, 1.5}	105	45	100	50	75	75
LETTER	{1, 2, 3, 4, 5}	16000	4000	–	–	–	–
MACKEY	{0, 0.5, 1, 1.5}	–	–	–	–	500	500
MUSHROOMS	{0, 1.5, 3, 4.5}	–	–	5333	2167	–	–
SAT	{2, 4, 6, 8, 10}	–	–	–	–	3218	3217
VOTES	{1, 2, 3, 4}	305	130	290	145	–	–
WAVE	{1, 2.5, 4, 5.5, 7}	300	4700	–	–	–	–
WAVE+NOISE	{4, 6, 8, 10, 12}	300	4700	–	–	–	–
WINE	{1, 2, 3, 4, 5}	–	–	118	60	89	89

Vergleich mit dem NGE- und dem kNN-Algorithmus

Versuchsbeschreibung In [WD95] werden zahlreiche Varianten des NGE-Algorithmus in einer umfangreichen Studie experimentell mit verschiedenen kNN-Algorithmen verglichen. Der Vorgänger des PNC 2-Algorithmus wurde bereits in [Hae01b] und [Hae01a] anhand dieser Studie

[6] Beim Benchmark LETTER werden dabei zudem nur 8000 der 16000 Datenpunkte der Lernstichprobe benutzt.

mit den NGE- und kNN-Algorithmen verglichen. Dabei wurde immer diejenige NGE- bzw. kNN-Variante zum Vergleich herangezogen, die auf dem jeweiligen Benchmark die beste Prognosegüte erzielt. Dies ist problematisch, denn es kann nicht garantiert werden, daß mittels eines Parametertunings auf den jeweiligen Lernstichproben eben jene jeweils beste Variante ausgewählt werden würde. Für den Vergleich mit dem kNN-Algorithmus kann dieses Problem jedoch umgangen werden, denn bei genauerer Betrachtung ist es möglich, aus den vier kNN-Varianten eine für den Vergleich auszuwählen. Die verschiedenen kNN-Varianten sind in Tabelle 5.18 aufgelistet. Zwei Varianten verwenden – wie der PNC 2-Algorithmus – auf Basis der Transinformation berechnete Eingangsgrößengewichte. Von diesen beiden wird bei der Variante $k\text{NN}_{CV\,MI}$ eben jene Vorgehensweise verfolgt, den Parameter k mittels Parametertunings auf den jeweiligen Lerndaten einzustellen. Die Variante NN_{MI} ist insofern mit der Wahl von $k = 1$ in der Variante $k\text{NN}_{CV\,MI}$ enthalten. Somit braucht für einen Vergleich nur die Variante $k\text{NN}_{CV\,MI}$ betrachtet zu werden. Für den Vergleich mit dem NGE-Algorithmus ist es nicht möglich, eine bestimmte Variante fest auszuwählen. Daher wird, trotz der oben geschilderten Problematik, weiterhin das Ergebnis der jeweils besten Variante herangezogen. Des weiteren wird die mit der Variante OBNGE erzielte Prognosegüte angegeben, da diese Variante im Ansatz dem PNC 2-Algorithmus ähnlich ist.

Tabelle 5.18: In [WD95] verwendete kNN-Varianten.

$k\text{NN}_{CV}$	kNN-Algorithmus bei dem der Parameter k mittels 10-facher Kreuz-Validierung auf der jeweiligen Lernstichprobe eingestellt wird
NN	kNN-Algorithmus mit fester Wahl von $k = 1$
$k\text{NN}_{CV\,MI}$	wie $k\text{NN}_{CV}$, jedoch werden zusätzlich aufgrund der Transinformation berechnete Eingangsgrößengewichte verwendet
NN_{MI}	wie NN, ebenfalls mit Eingangsgrößengewichten wie $k\text{NN}_{CV\,MI}$

Die Ergebnisse in [WD95] wurden mit 25-facher Wiederholung ermittelt. Für die dabei verwendete Aufteilung siehe Tabelle 5.17. Der in [WD95] angegebene Standard-Fehler der Prognosegüte wird mit dem Faktor $\sqrt{N_R} = 5$ auf die Standardabweichung der Prognosegüte umgerechnet. Da die Werte allerdings nur mit einer Genauigkeit von einer Nachkommastelle angegeben sind, kann es bei der Umrechnung zu größeren Ungenauigkeiten kommen. Für den Benchmark LETTER ist in [WD95] ein Standard-Fehler von 0.0% angegeben. Es ist unwahrscheinlich, daß der Wert exakt 0 gewesen ist. Daher wird hier für den Benchmark LETTER keine Angabe zur Standardabweichung der Prognosegüte des $k\text{NN}_{CV\,MI}$-Algorithmus gemacht.

Auswertung Der PNC 2-Algorithmus erzielt auf allen betrachteten Benchmarkbeispielen deutlich bessere Prognosegüten als die jeweils beste Variante des NGE-Algorithmus. Mit der Variante OBNGE lassen sich nur ausgesprochen schlechte Prognosegüten erzielen. Dies ist besonders interessant, da der OBNGE dem PNC 2-Algorithmus im Ansatz ähnlich ist. Jedoch wird beim PNC 2-Algorithmus mit dem Strategieparameter ω_{COD} ein Mechanismus zur Begegnung der COD-Problematik eingesetzt. Wenn dieser durch die Wahl von $\omega_{COD} = 0$ deaktiviert wird, führt dies – ebenso wie beim OBNGE – an zahlreichen höherdimensionalen Benchmarks zu einer sehr starken Vereinigungsrate, damit zu einer sehr kleinen Clusteranzahl und letztendlich im Extremfall zu gleichermaßen schlechten Prognosegüten. In [WD95] wurde aus dem schlechten Abschneiden des OBNGE gefolgert, daß der beim OBNGE verfolgte Ansatz ungeeignet sei, und daß ein Problem der NGE-Algorithmen in dem Entstehen von überlappenden Hyperquadern zu sehen sei. Die hier erzielten Ergebnisse stellen diese Folgerungen in Frage.

Die Prognosegüte des $k\text{NN}_{CV\,MI}$-Algorithmus wird bei einigen Benchmarks leicht übertroffen,

Tabelle 5.19: Vergleich des PNC 2- mit dem NGE- und kNN-Algorithmus.

		[WD95]				
	$k\text{NN}_{CV\,MI}$	NGE_{Best}		OBNGE		
Benchmark	$\widehat{Q}_T$	$\widehat{Q}_T$	K	$\widehat{Q}_T$	K
CLEVELAND	18.3 % ± 3.0 %	21.5 %	57	28.7 %	63
IRIS	4.9 % ± 2.5 %	5.3 %	6	6.8 %	4
LETTER	3.4 % ± 0.– %	8.7 %	2560	–*	–*
VOTES	4.6 % ± 2.0 %	5.3 %	46	11.2 %	38
WAVE	17.4 % ± 4.5 %	25.9 %	116	29.7 %	9
WAVE+NOISE	17.6 % ± 3.0 %	29.9 %	20	35.5 %	5

* keine Angabe, da laut [WD95] eine Simulation aufgrund des zu hohen Rechenaufwands nicht möglich war

	PNC 2				
Benchmark	$\widehat{Q}_T$	K	K_{Red}	ø m	N_R
CLEVELAND	18.12 % ± 3.6 %	74.9	6.3	7.0	250
IRIS	4.54 % ± 2.6 %	9.3	3.9	1.5	250
LETTER	5.71 % ± 0.4 %	1566	1109	7.6	25
VOTES	4.93 % ± 1.5 %	34.4	4.5	4.4	150
WAVE	18.10 % ± 1.1 %	41.7	22.0	10.1	150
WAVE+NOISE	17.21 % ± 0.9 %	37.1	27.6	16.1	150

bei anderen Benchmarks jedoch nicht erreicht. Die Größe der generierten Prognosemodelle ist bei allen Benchmarks wesentlich kleiner. Zum Vergleich bedenke man, daß beim $k\text{NN}_{CV\,MI}$ alle Lerndatenpunkte zum Treffen einer Prognose verwendet werden.

Die Prognosegüte des PNC 2-Algorithmus ist beim Benchmark WAVE+NOISE um etwa 5 % besser als beim Benchmark WAVE. Dieser Unterschied ist – bei Auswertung der Prognosegüten für die einzelnen Wiederholungen mittels des, in Anhang C beschriebenen, t-Tests für gepaarte Ereignisse – signifikant mit einer Irrtumswahrscheinlichkeit von 0.0%. Die beiden Benchmarks sind gleich bis auf den Unterschied, daß beim Benchmark WAVE+NOISE 19 irrelevante Eingangsgrößen, die ausschließlich mittelwertfreies Rauschen enthalten, hinzugefügt wurden. Daher wäre eigentlich ein gegenteiliges Ergebnis zu erwarten gewesen. Durch weitere Untersuchungen wurde herausgefunden, daß das Hinzufügen der irrelevanten Eingangsgrößen eine, individuell mit der Masse jedes Clusters ansteigende, dynamische Erhöhung des Strategieparameters ω_{COD} bedingt: Cluster, in denen bereits viele Datenpunkte vereinigt wurden, decken bezüglich der irrelevanten Eingangsgrößen einen großen Teil der möglichen Spannweite dieser Größen ab, und daher werden kaum noch Datenpunkte bezüglich dieser Größen außerhalb liegen. Dieser Effekt sollte zukünftig näher untersucht werden, um dabei Hinweise auf weitere Verbesserungen des PNC 2-Algorithmus zu erhalten.

Vergleich mit dem RISE-Algorithmus

Versuchsbeschreibung Der RISE-Algorithmus wird in [Dom96] experimentell an zahlreichen Benchmarks mit verschiedenen anderen Lernverfahren verglichen. Dabei werden die Prognosegüten mit einer 50-fachen Wiederholung ermittelt. Für die dabei verwendete Aufteilung siehe Tabelle 5.17. Von den dort verwendeten Benchmarks werden diejenigen, die auch bisher in dieser Arbeit Verwendung fanden und drei weitere, zufällig ausgewählte, zum Vergleich herangezogen.

N_{Mem} bezeichnet den Speicherplatzbedarf der gelernten Modelle und ergibt sich für den PNC 2-Algorithmus aus der Anzahl der Cluster K_{Red} und der durchschnittlichen Anzahl pro Cluster betrachteter Eingangsgrößen $\varnothing\ m$ zu

$$N_{Mem} = K_{Red}(\varnothing m + 1). \tag{5.9}$$

Tabelle 5.20: Vergleich des PNC 2- mit dem RISE-Algorithmus.

	RISE		PNC 2					
Benchmark	$\widehat{Q}_T$	N_{Mem}	$\widehat{Q}_T$	N_{Mem}	K	K_{Red}	$\varnothing\ m$	N_R
CHESS	1.8 % ± 0.2 %	3064	2.50 % ± 0.8 %	721	69.8	34.0	21.2	25
CLEVELAND	20.3 % ± 3.8 %	1461	17.97 % ± 3.5 %	46	71.5	5.8	6.9	250
DNA	6.9 % ± 1.6 %	8351	4.39 % ± 0.5 %	5117	285.4	208.0	23.6	25
HEPATITIS*+	21.7 % ± 5.7 %	1232	18.54 % ± 5.0 %	61	18.4	6.5	8.3	250
IRIS+	6.0 % ± 2.8 %	383	4.66 % ± 2.5 %	10	9.0	4.0	1.5	250
MUSHROOMS*	0.0 % ± 0.2 %	398	0.00 % ± 0.0 %	19	13.4	8.8	1.1	25
VOTES+	4.8 % ± 1.5 %	541	4.85 % ± 1.5 %	23	29.3	4.5	4.2	150
WINE+	3.1 % ± 2.0 %	896	3.36 % ± 2.0 %	52	8.8	6.4	7.1	250

* bzw. + bedeutet: Benchmark wurde zur Entwicklung beim PNC 2- bzw. beim RISE-Algorithmus benutzt

Anmerkung: Es ist unklar, wie sich beim RISE-Algorithmus beim Benchmark MUSHROOMS die angegebene Standardabweichung ergeben konnte.

Auswertung Es zeigt sich, daß sich mit dem PNC 2 meist bessere Prognosegüten erzielen lassen als mit dem RISE-Algorithmus. Zudem sind die generierten Modelle teilweise wesentlich kleiner. Jedoch ist anzumerken, daß für den RISE-Algorithmus ein Reduktionsmechanismus angegeben ist, dessen Anwendung – bei leichten Einbußen der Prognosegüte – durchschnittlich zu einer Verringerung der Modellgrößen um etwa 90 % führt. Siehe hierzu Kapitel 3.4. DOMINGOS argumentiert in [Dom96], die Prognosegüte des RISE könne nicht schlechter sein, als die des kNN-Algorithmus, da beim RISE-Algorithmus zunächst alle Lerndatenpunkte als triviale Regeln angesehen werden, womit in diesem Zustand der Algorithmus identisch zum kNN ist, und nur dann Regeln generalisiert werden, wenn dies keinen negativen Einfluß auf die bei Kreuz-Validierung bezüglich der Lerndaten erzielte Prognosegüte hat. Die Ergebnisse in den Tabellen 5.19 und 5.20 wurden mit in etwa gleicher Aufteilung der Datenpunkte in Lern- und Testdaten ermittelt. Beim Vergleich fällt auf, daß der RISE-Algorithmus oft schlechtere Prognosegüten als die von [WD95] verwendete kNN Variante erzielt. Die Aussage von DOMINGOS ist insofern zu konkretisieren, als daß sie sich nur auf die innerhalb des RISE-Algorithmus verwendete kNN-Variante bezieht.

Vergleich mit dem FR-Algorithmus

Versuchsbeschreibung In [Sla01] wurde für den FR-Algorithmus eine systematische Vorgehensweise definiert und an zahlreichen Lernaufgaben getestet. Dabei werden die benötigten eingangsseitigen ZGF anhand der Spannweite der jeweiligen Eingangsgröße als äquidistant verteilte trapez- oder dreieckförmige ZGF erzeugt. Zur Defuzzifizierung wird bei Klassifikationsaufgaben die Maximum- und für Regressionsaufgaben die COG-Defuzzifizierung verwendet. Die weiteren freien Parameter des FR-Algorithmus – wie beispielsweise der verwendete Regeltest – werden, ähnlich der beschriebenen Vorgehensweise bei harter Validierung, wie folgt ermittelt: Für eine gegebene Lernstichprobe wird für alle möglichen Einstellungen der freien Parameter ein Modell

gelernt und die Prognosegüte dieses Modells bezüglich der verwendeten Lernstichprobe ermittelt. Der Parametersatz, der dabei zur besten Prognosegüte führt, wird ausgewählt. Dabei wird die Randbedingung, daß die Anzahl der generierten Regeln des Modells nicht größer sein darf, als die Hälfte der Anzahl der Lerndatenpunkte, berücksichtigt. Man beachte, daß hier nicht wie beim PNC 2-Algorithmus eine Kreuz-Validierung o.ä. zur Ermittlung der Lernstichproben-Prognosegüte durchgeführt wird. Damit ist die Vorgehensweise beim FR-Algorithmus einfacher und dürfte trotzdem ausreichend sein, da *eine hohe Korrelation zwischen dem Modellierungsfehler auf Lern- und Testdaten vorliegt* [Sla01](S. 125). Beim PNC 2-Algorithmus wäre eine derartige Vorgehensweise nicht möglich, da eine fast optimale Prognosegüte auf Lerndaten sich allein dadurch erzielen läßt, daß z.B. durch die Wahl von $\omega_{COD} = m$ keine Vereinigungen mehr zugelassen werden und das Verfahren zum kNN entartet.

Zur anschließenden Reduktion der Regelanzahl kommt die optimierende Konfliktreduktion (OCR) [Kro99] zum Einsatz. Dabei wird der Regelsatz durch Ein- und Ausschalten einzelner Regeln mittels eines Evolutionären-Algorithmus bezüglich der auf den Lerndaten erzielten Prognosegüte optimiert. Die systematische Vorgehensweise wird analog zu [Sla01] im folgenden als FR_{Sys} bezeichnet. Wird anschließend noch eine optimierende Konfliktreduktion durchgeführt, wird die Bezeichnung $\text{FR}_{Sys+OCR}$ verwendet.

In [SNK01] wird ein Verfahren zur gleichzeitigen Eingangsgrößenselektion und Optimierung dazugehöriger ZGF für den FR-Algorithmus vorgestellt. Dabei werden iterativ mittels einer Vorwärts-Selektion sowohl Eingangsgrößen ausgewählt, als auch gleichzeitig für diese ausgewählten Eingangsgrößen ZGF mittels einer evolutionären Suche optimiert. Im folgenden wird der Einsatz dieses Verfahrens zusammen mit dem FR-Algorithmus als FR_{EFS} bezeichnet.

Die angegebenen Ergebnisse des FR-Algorithmus sind mittels einer 2-fachen Kreuz-Validierung erzielt worden. Damit läßt sich, vor allem bei den vom Umfang her kleineren Benchmarkbeispielen, nur ein grober Schätzwert für die Prognosegüten ermitteln. Um robustere Schätzwerte zu erhalten, wird die Prognosegüte für den PNC 2 – bei gleicher Aufteilung wie bei einer 2-fachen Kreuz-Validierung – mittels einer N_R-fachen Wiederholung ermittelt.

Auswertung Es zeigt sich, daß sich mit dem PNC 2 oft bessere Prognosegüten erzielen lassen als mit dem Standardansatz des FR-Algorithmus. Zur Vergleichbarkeit von Regelanzahl und Clusteranzahl sei darauf hingewiesen, daß beim FR-Algorithmus nur maximal 2 Terme in den Regelprämissen verknüpft werden. Beim PNC 2-Algorithmus ergibt sich die Anzahl der Prämissenterme als durchschnittliche Anzahl betrachteter Eingangsgrößen $\varnothing\ m$. Somit sind die mit dem PNC 2-Algorithmus generierten Modelle teilweise signifikant größer. Verglichen mit der Variante FR_{EFS} erzielt der PNC 2-Algorithmus teilweise bessere und teilweise schlechtere Prognosegüten.

Die anschließende Verwendung der optimierenden Konfliktreduktion führt beim FR-Algorithmus bei allen Benchmarks zu kleineren Regelsätzen bei gleichzeitiger Verbesserung der Prognosegüte[7]. Bei der optimierenden Konfliktreduktion wird eine globale Optimierung des Regelsatzes bezüglich der Prognosegüte durchgeführt. Beim PNC 2-Algorithmus ist derzeit kein derartiges nachgeschaltes Verfahren realisiert. Es wäre zu untersuchen, ob bzw. inwieweit sich mit einem solchen oder ähnlichem Verfahren beim PNC 2-Algorithmus Verbesserungen bezüglich der Prognosegüte oder der Modellgröße erzielen lassen.

In [Kra01] wird gezeigt, daß sich bei Regressionsaufgaben die Prognosegüte der mit dem FR-Algorithmus generierten Regelsätze durch die Verwendung von TSK-Modellen für die Regelkon-

[7] Allerdings ist dafür bei den vom Stichprobenumfang her größeren Benchmarks ein nennenswerter Rechenaufwand verbunden. Beim Benchmark DNA bzw. SAT dauerte die Optimierung auf einem seinerzeit üblichen Büro-Computer etwa eine Stunde bzw. einen Tag.

Tabelle 5.21: Vergleich des PNC 2- mit dem FR-Algorithmus.

Benchmark	FR_{Sys} $\widehat{Q}_T$	K	$\text{FR}_{Sys+OCR}$ $\widehat{Q}_T$	K	FR_{EFS} $\widehat{Q}_T$	K
DNA	5.5 %	1567	5.4 %	500	5.8 %	141
IRIS	6.0 %	7	6.0 %	7	0.0 %	9
SAT	18.8 %	2683	18.2 %	1044	$^{+}$15.9 %	161
WINE	11.2 %	105	10.7 %	15	2.6 %	34
KIN32FM	0.22	1530	0.12	457	*–	*–
Mackey	0.07	59	0.05	20	$^{+}$0.04	47

* Es liegen keine Ergebnisse vor.
$^{+}$ Bisher nicht veröffentlichte Ergebnisse mit OCR laut persönlicher Kommunikation

Benchmark	PNC 2 $\widehat{Q}_T$	K	K_{Red}	ø m	N_R
DNA	4.78 % ± 0.4 %	217.7	162.9	23.7	25
IRIS	4.82 % ± 2.0 %	6.9	3.5	1.4	250
SAT	9.63 % ± 0.4 %	374.9	332.8	14.3	25
WINE	3.53 % ± 2.0 %	7.6	5.1	7.0	250
KIN32FM	0.20 ± 0.004	846	505.4	11.2	10
Mackey	0.03 ± 0.002	51.8	37.0	2.1	150

klusionen – um den Preis einer größeren Komplexität des Modells – weiter erhöhen läßt. Beim Benchmark MACKEY wird damit die erzielte Prognosegüte auf 0.0094 bei 24 Regeln erhöht. Wenn zugleich die verwendeten ZGF mit dem oben genannten Verfahren [SNK01] optimiert werden, ergibt sich eine weitere Verbesserung der Prognosegüte auf 0.0054 bei etwa 120 Regeln. Es wäre zu untersuchen, wie gut sich dieser Ansatz auf den PNC 2-Algorithmus übertragen läßt.

5.4 Ergebnisse SMBC

In diesem Kapitel wird zunächst die Arbeitsweise des SMBC-Algorithmus an zwei einfachen Illustrationsbeispielen dokumentiert. Anschließend wird der SMBC in einer normalen Validierung mit dem FR-Algorithmus verglichen. Die freien Parameter des Algorithmus werden, wie in Tabelle 5.22 aufgeführt, ad hoc eingestellt. Wie in Kapitel 2.2 beschrieben wird als UND-Operator die Multiplikation, als ODER-Operator die gewöhnliche Summe und zur Defuzzifizierung bei Regressionsaufgaben die COG- und bei Klassifikationsaufgaben die MOM-Methode verwendet.

5.4.1 Illustrationsbeispiel *quadratische Form*

Als zu modellierendes System wird zunächst die quadratische Form

$$y = \mathbf{x}^T \begin{bmatrix} 2 & 0 \\ 1 & 1 \end{bmatrix} \mathbf{x} \tag{5.10}$$

Tabelle 5.22: Gewählte Einstellungen für die freien Parameter des SMBC-Algorithmus.

Parameter		**erläuternder Kontext**
K_{Start}	1	Anzahl Startcluster
α	2	Fuzziness-Parameter
β	0.2	Modifikationsfaktoren nach Gl. (4.5)
γ	6	Modifikationsfaktoren nach Gl. (4.6)
u^{α}_{min}	5	minimale Zuteilung; Löschen von Clustern
ϱ_{min}	0	minimale Relevanz; Löschen von Clustern
ω_{min}	0.9	Ähnlichkeitsschwelle für Vereinigung

betrachtet, für die 1000 Datenpunkte durch zufällig gewählte Eingangsvektoren erzeugt wurden. Für die Berechnung der Modifikationsfaktoren wird die histogramm-basierte Variante nach Gl. (4.6) gewählt und abweichend von Tabelle 5.22 der Parameter u^{α}_{min} auf den Wert 2 eingestellt. Abbildung 5.5 zeigt die Verteilung der Datenpunkte und die sich nach 40 Lernschritten ergebende Clusterkonfiguration im Eingangsraum. Für jedes Cluster sind dabei das Zentrum und die Ellipse, die alle Punkte mit gleicher Zugehörigkeit – und zwar $\mu = \frac{1}{e}$ – zum Cluster enthält, eingezeichnet. Offensichtlich sind die Cluster wunschgemäß entlang der Höhenlinien der quadratischen Form ausgerichtet – und dies, obwohl im Eingangsraum keinerlei Clusterstruktur vorhanden ist.

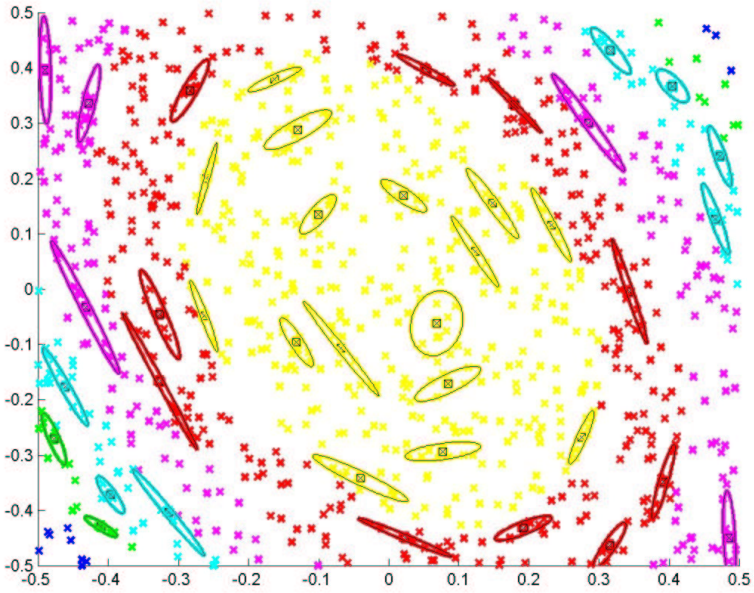


Abbildung 5.5: Datenpunkte und mit dem SMBC-Algorithmus generierte Cluster im Eingangsraum.

5.4.2 Illustrationsbeispiel *Klassifikation*

Im folgenden Beispiel soll die Wirkungsweise des Vereinigungsmechanismus' illustriert werden. Gegeben sei das in Abbildung 5.6 bzw. 5.7 im zweidimensionalen Eingangsraum dargestellte Klassifikationsproblem. Die Ausgangsgröße kann eines von vier verschiedenen Symbolen annehmen. Für die dargestellten Datenpunkte ist die zugehörige Ausgangsklasse farblich markiert.

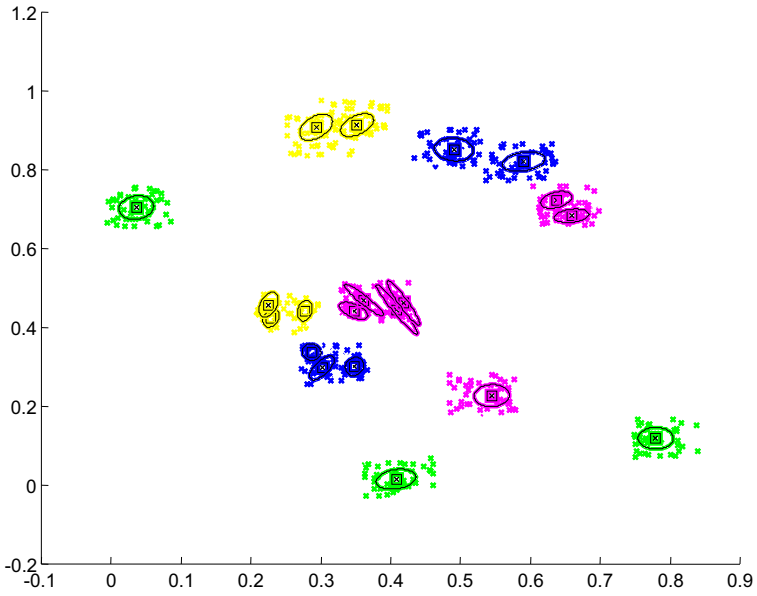


Abbildung 5.6: Datenpunkte und mit dem SMBC-Algorithmus generierte Cluster im Eingangsraum ohne Vereinigungsmechanismus.

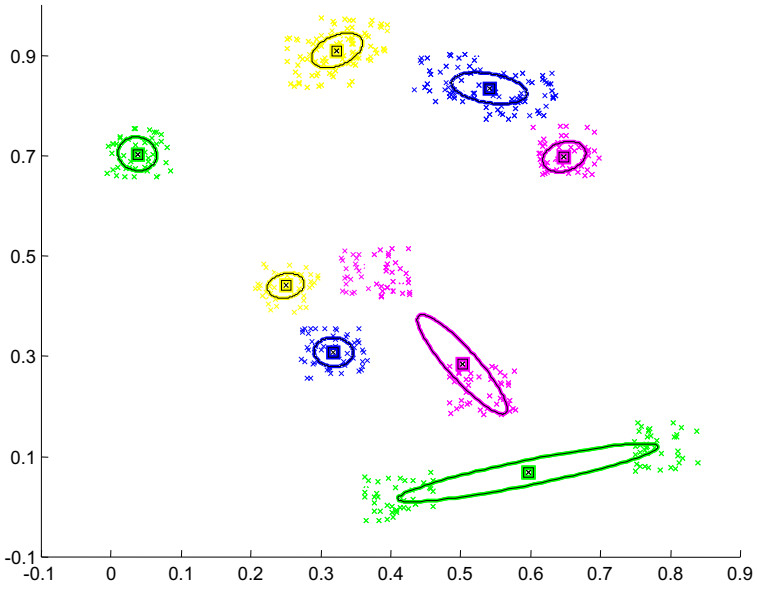


Abbildung 5.7: Datenpunkte und mit dem SMBC-Algorithmus generierte Cluster im Eingangsraum mit Vereinigungsmechanismus.

Abbildung 5.6 bzw. 5.7 zeigt die sich ergebende Clusterkonfiguration nach 20 Epochen, ohne bzw. mit aktiviertem Vereinigungsmechanismus. Für jedes Cluster sind dabei wiederum das Zentrum und die Ellipse, die alle Punkte mit gleicher Zugehörigkeit – und zwar $\mu = \frac{1}{e}$ – zum Cluster enthält, eingezeichnet. Wie zu sehen ist, ist das Clusterergebnis ohne Vereinigungsmechanismus zwar durchaus korrekt im Sinne einer Regelgenerierung, jedoch werden Zusammenhänge teilweise von mehr Clustern repräsentiert, als dies notwendig wäre. Dem wird

durch den Vereinigungsmechanismus Abhilfe geleistet. Es sei angemerkt, daß die Vereinigung der Cluster rechts unten in Abbildung 5.7 aus Clustergesichtspunkten sicherlich nicht sinnvoll erscheinen würde. Jedoch wird das Verfahren nicht zum Clustern an sich, sondern zur Regelgenerierung eingesetzt und unter diesem Gesichtspunkt sind diese Vereinigungen durchaus akzeptabel bzw. sogar erwünscht.

5.4.3 Benchmarkbeispiele

Versuchsbeschreibung Der SMBC-Algorithmus wird anhand eines Regressions- und dreier Klassifikationsaufgaben getestet und die Ergebnisse werden mit denen verglichen, die sich beim FR-Algorithmus bei Anwendung der in [Sla01] definierten Standardvorgehensweise bei 2-facher Kreuz-Validierung ergeben. Ebenso werden die in [SNK01] erzielten Ergebnisse bei zusätzlicher Anwendung einer vorgeschalteten Eingangsgrößenselektion und evolutionären Optimierung der zugehörigen ZGF betrachtet. Da der SMBC-Algorithmus nicht deterministisch arbeitet, werden für jede Kreuz-Validierung jeweils 10 Testläufe durchgeführt und die resultierenden Mittelwerte angegeben. Als Prognosegüte ist – je nach Aufgabentyp – jeweils der mittlere absolute Fehler MAE bzw. der mittlere Klassifizierungsfehler MCE bezüglich der Teststichprobe als Ergebnis angegeben. FR_{Sys} bezeichnet die Standardvorgehensweise, $\text{FR}_{Sys+OCR}$ bedeutet, daß der mit der Standardvorgehensweise generierte Regelsatz anschließend bezüglich der auf den Lerndaten erzielten Güte optimiert wurde und FR_{EFS} zeigt an, daß die verwendeten ZGF vorher mit dem evolutionären Verfahren zur Eingangsgrößenselektion optimiert wurden. Für nähere Erläuterungen dieser verschiedenen Varianten siehe die Ausführungen im Rahmen des Vergleichs des PNC 2- mit dem FR-Algorithmus in Kapitel 5.3.6. SMBC_{Cov} und SMBC_{Diag} bezeichnen das jeweils für den SMBC-Algorithmus verwendete Clustermodell. Bei ersterem wurde die volle, bei letzterem nur die diagonalisierte Kovarianzmatrix verwendet. Die Regelanzahl N_R bzw. die Clusteranzahl K sind angegeben.

Die Modifikationsfaktoren werden für Regressionsaufgaben nach Gl. (4.5) und für Klassifikationsaufgaben nach Gl. (4.9) ermittelt und die modifizierte Partitionsmatrix wird re-normalisiert. Die Eingangsgrößen werden mittelwert-bereinigt und auf die jeweilige Standardabweichung normiert[8]. Insgesamt werden jeweils 40 Lernschritte durchgeführt und als endgültige Clusterkonfiguration diejenige mit der besten Prognosegüte auf den Lerndaten ausgewählt.

Tabelle 5.23: Vergleich des SMBC- mit dem FR-Algorithmus.

Benchmark	FR_{Sys}		$\text{FR}_{Sys+OCR}$		FR_{EFS}		SMBC_{Cov}		SMBC_{Diag}	
	$\widehat{Q}_T$	N_R	$\widehat{Q}_T$	N_R	$\widehat{Q}_T$	N_R	$\widehat{Q}_T$	K	$\widehat{Q}_T$	K
IRIS*	6.0%	7	6.0%	7	0.0 %	9	4.8%	4.3	4.7%	4.3
SAT	18.8 %	2683	18.2 %	1044	+15.9 %	161	22.3 %	20.2	17.5 %	37.7
WINE	11.2%	105	10.7%	15	2.6 %	34	1.8%	3.8	2.4%	3.6
MACKEY	0.07	59	0.05	20	0.04	47	0.026	21.3	0.032	30.4

* Für den SMBC-Algorithmus sind nur die Eingangsgrößen 3 und 4 verwendet worden.
\+ Bisher nicht veröffentlichte Ergebnisse mit OCR laut persönlicher Kommunikation

Auswertung Es zeigt sich, daß mit dem SMBC- bessere Ergebnisse erzielbar sind als mit der Standardvorgehensweise des FR-Algorithmus. Verglichen mit der Variante FR_{EFS} erzielt der SMBC-Algorithmus in zwei Fällen eine bessere und in zwei Fällen eine schlechtere Prognosegüte.

[8] Hierbei werden auch auf Testdaten die auf den Lerndaten berechneten Werte verwendet.

Je nach Lernaufgabe ist ggf. die Variante SMBC$_{Diag}$ der Variante mit der vollen Kovarianzmatrix SMBC$_{Cov}$ vorzuziehen.

Die erzielten Ergebnisse des SMBC lassen sich auch mit denjenigen des PNC 2-Algorithmus in Tabelle 5.21 vergleichen. Dabei zeigt sich, daß der SMBC beim Benchmark WINE eine bessere, beim Benchmark SAT jedoch eine wesentlich schlechtere Prognosegüte erzielt. Bei den Benchmarks IRIS und MACKEY erzielen beide Verfahren im wesentlichen die gleiche Prognosegüte.

Die anschließende Verwendung der optimierenden Konfliktreduktion führt beim FR-Algorithmus bei allen Benchmarks zu kleineren Regelsätzen bei gleichzeitiger Verbesserung der Prognosegüte. Bei der optimierenden Konfliktreduktion wird eine globale Optimierung des Regelsatzes bezüglich der Prognosegüte durchgeführt. Beim SMBC-Algorithmus ist derzeit kein derartiges nachgeschaltes Verfahren realisiert. Es wäre zu untersuchen, ob bzw. inwieweit sich mit einem solchen oder ähnlichem Verfahren beim SMBC-Algorithmus Verbesserungen bezüglich der Prognosegüte oder der Modellgröße erzielen lassen.

In [Kra01] wird gezeigt, daß sich bei Regressionsaufgaben die Prognosegüte der mit dem FR-Algorithmus generierten Regelsätze durch die Verwendung von TSK-Modellen für die Regelkonklusionen – um den Preis einer größeren Komplexität des Modells – weiter erhöhen läßt. Beim Benchmark MACKEY wird damit die erzielte Prognosegüte auf 0.0094 bei 24 Regeln erhöht. Wenn zugleich die verwendeten ZGF mit dem oben genannten Verfahren [SNK01] optimiert werden, ergibt sich eine weitere Verbesserung der Prognosegüte auf 0.0054 bei etwa 120 Regeln. Es wäre zu untersuchen, wie gut sich dieser Ansatz auf den SMBC-Algorithmus übertragen läßt.

Kapitel 6

Zusammenfassung und Ausblick

Modelle für das Ein/Ausgangsverhalten eines Systems können in vielfältiger Weise zur Simulation, Kennfeldnachbildung, Objekterkennung und -klassifikation, zur Prognose oder auch zum Wissenserwerb über ein System eingesetzt werden. Mit zunehmender Rechnerkapazität kommen zur Modellerstellung vermehrt datenbasierte Vorgehensweisen zum Einsatz, bei denen am realen System Datenpunkte erhobenen werden, anhand derer dann automatisch mittels eines Lernverfahrens ein Modell generiert wird. Der Vorteil dieser Vorgehensweise ist, daß wenig oder kein Hintergrundwissen über das zu modellierende System benötigt wird und so mit relativ geringem Arbeitsaufwand ein Modell erstellt werden kann. Wenn neben einer hohen Prognosegüte auch die Interpretierbarkeit der generierten Modelle gefordert ist, empfiehlt sich der Einsatz eines regelbasierten Lernverfahrens, das die gelernten Zusammenhänge in Form von Wenn-Dann-Regeln repräsentiert. Jede Regel beschreibt einen signifikanten Teil des Ein/Ausgangsverhaltens, und ist somit sowohl einzeln für sich, als auch in ihrem Zusammenspiel mit anderen Regeln für einen Menschen intuitiv verständlich.

Clusterverfahren partitionieren eine gegebene Menge an Datenpunkten anhand eines Ähnlichkeitskriteriums in verschiedene Gruppen, die als Cluster bezeichnet werden. Im Kontext der datenbasierten Modellierung können Clusterverfahren zur direkten Regelgenerierung eingesetzt werden. Jedes Cluster entspricht dabei direkt einer zugehörigen Regel bzw. kann, beispielsweise durch Projektion, in eine solche überführt werden.

Im Gegensatz zur Clusteranalyse, bei der meist in irgendeiner Weise zusammengeballte Punktewolken im (Eingangs-)Raum identifiziert werden sollen, ist das Ziel der Regelgenerierung zusammenhängende Gebiete des Eingangsraums, in denen die Ausgangsgrößenwerte möglichst homogen sind, zu identifizieren. In dieser Arbeit werden mit dem Pnc 2- und dem Smbc-Algorithmus zwei Clusterverfahren entwickelt, indem bestehende unüberwachte Verfahren oder Strategien – in Hinblick auf die andere Zielvorgabe – geeignet erweitert oder angepaßt werden. Neben einer hohen Prognosegüte wird dabei insbesondere auf den Aspekt der Interpretierbarkeit des generierten Regelsatzes geachtet. Daher werden während des Clustervorgangs nur lokale Entscheidungskriterien eingesetzt, so daß jedes Cluster eine einzeln getestete und lokal sinnvolle Regel darstellt.

Der Pnc 2-Algorithmus basiert auf dem Konzept der hierarchischen agglomerativen Clusterverfahren, bei denen – ausgehend von einer Partitionierung, in der jeder Datenpunkt ein eigenes Cluster darstellt – schrittweise, bis zum Erreichen eines Abbruchkriteriums, immer zwei einander anhand eines Ähnlichkeitskriteriums nahe liegende Cluster miteinander vereinigt werden. Jedes Cluster wird durch einen Ausgangsgrößenwert und einen, im Eingangsraum definierten Hyperquader, der die Eingangsraumpositionen aller zugeordneten Datenpunkte einschließt, repräsentiert. Grundidee des Verfahrens ist es, eine Vereinigung nur dann zuzulassen, wenn das

dann entstehende generalisierte Cluster einen Regeltest besteht. Zur Bewältigung der COD-Problematik wird dieser Regeltest in hochdimensionalen Räumen geeignet modifiziert. Beim Einsatz des gelernten Regelsatzes wird, wenn keine einzige Regel aktiviert ist, der Prognosewert – ähnlich wie beim kNN-Algorithmus – aus den nächstliegenden Regeln abgeleitet. Damit kann der PNC 2-Algorithmus auch als IBL-Lernverfahren betrachtet werden, mit dem Unterschied, daß die Lerndatenpunkte zu Prototypen generalisiert werden. Der PNC 2-Algorithmus arbeitet deterministisch, ist in der Lage, kontinuierliche und nominale Eingangsgrößen gleichzeitig zu verarbeiten und benötigt zum Lernen bei den betrachteten Anwendungsbeispielen eine etwa quadratisch mit der Lernstichprobengröße zunehmende Laufzeit. Zur Vereinfachung der generierten Regeln wird eine nachgeschaltete kontext-sensitive Eingangsgrößenselektion realisiert, deren Einsatz die Größe der Prognosemodelle, bei im wesentlichen gleichbleibender Prognosegüte, stark reduziert.

Zur Ausgestaltung des Verfahrens und zur Wahl der enthaltenen freien Parameter wurden umfangreiche Untersuchungen durchgeführt. Einige wenige Parameter wurden als Strategieparameter ausgewählt, mit denen die Arbeitsweise des Verfahrens an das zu bearbeitende Problem angepaßt werden kann. Für diese Strategieparameter wurde eine systematische Vorgehensweise, mittels derer eine vollautomatische Einstellung im Rahmen einer Kreuz-Validierung bezüglich der jeweiligen Lerndaten erfolgt, entwickelt und getestet. Der PNC 2 wurde mit den ähnlichsten in der Literatur bekannten Algorithmen, namentlich dem NGE-, dem RISE-, dem kNN- und dem FR-Algorithmus, theoretisch und experimentell in einer Benchmarkstudie verglichen. Es zeigt sich, daß sich mit dem PNC 2 in vielen Fällen bessere Prognosegüten bei teilweise wesentlich kleineren Prognosemodellen erzielen lassen. Insbesondere im Vergleich zum kNN stellt der PNC 2 eine interessante Alternative dar, da die kleineren Prognosemodelle wesentlich besser zu interpretieren sind, geringeren Speicherplatz benötigen und das Treffen einer Prognose weniger Rechenaufwand erfordert.

Der SMBC-Algorithmus basiert auf dem Konzept der partitionierenden Clusterverfahren und erweitert unüberwacht arbeitende modell-basierte Clusterverfahren derart, daß die Information über die Ausgangsgrößenwerte der Datenpunkte während des Clustervorgangs gezielt ausgenutzt wird, um Cluster zu erhalten, deren zugeordnete Datenpunkte möglichst homogene Ausgangsgrößenwerte aufweisen. Die Cluster werden repräsentiert durch ein im Eingangsraum definiertes Zentrum und Geometrieparameter, sowie durch einen zugehörigen Ausgangsgrößenwert. Grundidee des Verfahrens ist es, anhand des Clusterausgangsgrößenwerts die Zuteilung von bezüglich der Ausgangsgröße *unpassenden* Datenpunkten zu modifizieren. Um eine automatische Anpassung der Clusteranzahl an die jeweils zu bearbeitende Problemstellung zu erreichen, finden Mechanismen zum Löschen, Vereinigen und Hinzufügen von Clustern Anwendung. Dazu wurde das Relevanzkonzept des FR-Algorithmus zur Bewertung von Regeln übertragen. Cluster mit besonders schlechter Bewertung werden gelöscht und redundante Cluster dann vereinigt, wenn dies anhand der resultierenden Bewertung des generalisierten Clusters sinnvoll erscheint. In Eingangsraumgebieten, in denen inhomogene Cluster, d.h. Cluster, deren zugeteilte Datenpunkte stark unterschiedliche Ausgangsgrößenwerte aufweisen, liegen, werden gezielt neue Cluster eingefügt.

Für den SMBC wurde die prinzipielle Wirkungsweise durch mehrere Illustrationsbeispiele verdeutlicht und die Anwendbarkeit anhand mehrerer Benchmarkbeispiele mit dem FR und dem PNC 2-Algorithmus verglichen. Dabei zeigt sich, daß sich bessere Prognosegüten als mit der Standardvorgehensweise des FR-Algorithmus erzielen lassen.

Die Prognosegüte des FR-Algorithmus kann durch eine vor- bzw. nachgeschaltete optimierende Konfliktreduktion bzw. evolutionäre Eingangsgrößenselektion teilweise deutlich erhöht werden. Dabei kommen globale Gütekriterien in Kombination mit evolutionären Suchverfahren zum Einsatz. Bei den beiden hier entwickelten Clusterverfahren sind zur Zeit keine derartigen

Mechanismen realisiert. Es wäre zu untersuchen, ob und inwieweit sich dadurch weitere Verbesserungen der Prognosegüte erreichen lassen. Bezüglich einer nachgeschalteten Optimierung des erhaltenen Regelsatzes ist beim SMBC zu berücksichtigen, daß das Verfahren tendenziell eher kleine Regelsätze generiert und es daher sinnvoll und möglich erscheint, Position und Ausdehnung der einzelnen Regeln zu optimieren. Beim PNC 2-Algorithmus könnte alternativ – aufgrund der Verwandschaft zu den IBL-Algorithmen – versucht werden, verschiedene bekannte Ansätze [WM00b] zur Reduktion der, vom IBL-Algorithmus zur Prognose verwendeten, Lerndatenpunkte für den Einsatz zur Regelreduktion zu übertragen.

Im Rahmen dieser Arbeit wurden ausschließlich Regeln der Form (2.40) betrachtet, die in ihrer Konklusion einen einzigen, konstanten Ausgangsgrößenwert empfehlen. Bei Regressionsaufgaben ist oftmals die Verwendung von TSK-Regeln der Form (2.10) günstiger. Analog zu [Kra01] könnte versucht werden, die beiden entwickelten Clusterverfahren für die Generierung von TSK-Modellen zu erweitern.

Ein weiterer Aspekt betrifft die Ausdehnung der Anwendungsgrenzen der beiden entwickelten Clusterverfahren in Bezug auf die Bewältigung sehr umfangreicher Lernstichproben mit mehreren hunderttausend Datenpunkten. Eine mögliche Abhilfe in solchen Fällen ist der vorgeschaltete Einsatz sogenannter *Sampling-Techniken* zur Reduktion der zu bearbeitenden Stichprobengröße. Im einfachsten Falle des sogenannten *Uniform Random Sampling* wird zufällig eine bestimmte Anzahl an Datenpunkten aus der ursprünglichen Lernstichprobe ausgewählt. Dabei kann jedoch das Problem auftreten, daß Zusammenhänge, die nur durch sehr wenige Datenpunkte repräsentiert werden, vollständig verlorengehen. Abhilfe schafft hier das in [PF00] vorgestellte *Density Biased Sampling*, bei dem Datenpunkte in Eingangsraumgebieten geringerer Wahrscheinlichkeitsdichte bevorzugt ausgewählt werden. Eine andere in [Dom97], in Kombination mit dem RISE-Algorithmus, angewandte Methode ist das sogenannte *Windowing*: Eine Anzahl Datenpunkte von beispielsweise $2\sqrt{N}$ wird zufällig so aus der ursprünglichen Lernstichprobe ausgewählt, daß alle vorkommenden Ausgangsklassen etwa gleich oft enthalten sind. Diese ausgewählten Datenpunkte bilden die Stichprobe $\mathcal{P}_L$ und die übrigen Datenpunkte die Stichprobe $\mathcal{P}_{Rest}$. Mit der Stichprobe $\mathcal{P}_L$ wird ein Modell gelernt und diejenigen Datenpunkte aus $\mathcal{P}_{Rest}$, die von diesem Modell falsch klassifiziert werden, werden zur Stichprobe $\mathcal{P}_L$ hinzugefügt. Dies wird bis zum Erreichen eines Abbruchkriteriums wiederholt.

Alternativ kommt auch die Überführung der beiden entwickelten Clusterverfahren in inkrementelle Varianten in Betracht. Erste Ansätze für den PNC 2 und den SMBC-Algorithmus sollen im folgenden kurz skizziert werden. Dabei kann die zu bearbeitende Lernstichprobe zusätzlich vorher mittels der oben angesprochenen Sampling-Techniken reduziert werden. Die inkrementellen Varianten ermöglichen zusätzlich eine gewisse Adaptivität der generierten Clusterpopulation in dem Sinne, daß in der weit zurückliegenden Vergangenheit bearbeitete Datenpunkte wenig oder gar keinen Einfluß mehr auf die aktuelle Clusterpopulation haben.

- PNC 2: Für eine kleine Grundlernstichprobe wird mit dem ursprünglichen PNC 2-Algorithmus eine Clusterpopulation gelernt. Anschließend wird inkrementell immer ein weiterer Datenpunkt bearbeitet, in ein elementares Cluster überführt, in die Population eingefügt und es wird versucht, dieses neue Cluster mit einem der bisherigen Cluster zu vereinigen. Für die Clusteranzahl ist eine Obergrenze, bei deren Überschreitung jeweils das Cluster mit der kleinsten Masse gelöscht wird, vorzusehen.

  Die im Rahmen des Vereinigungstests verwendeten Lerndatenpunkte stellen die *Menge potentieller negativer Beispiele* dar. Diese Menge wird nach Abschluß der Grundlernphase mit der Grundlernstichprobe initialisiert. Während des inkrementellen Betriebes wird jeweils der aktuell bearbeitete Datenpunkt in diese Menge eingefügt und dafür der *älteste* Datenpunkt gelöscht. Dadurch wird die Menge negativer Beispiele permanent aktualisiert

und somit Adaptivität bezüglich des Vereinigungstests erreicht. Um auch für die Cluster selber Adaptivität zu erzielen, wäre es möglich, die Hyperquader der Cluster in jedem Schritt um einen kleinen Prozentsatz schrumpfen zu lassen.

- SMBC: Eine inkrementelle Variante des SMBC-Algorithmus kann analog zu der in [Kie98] beschriebenen Vorgehensweise realisiert werden. Eine Grundlernphase wie beim PNC2-Algorithmus ist nicht notwendig. Für die zur Repräsentation der Cluster verwendeten Kenngrößen wie Schwerpunkt, Standardabweichung und Kovarianzmatrix existieren inkrementelle Schätzer. Der Ablauf einer inkrementellen Variante gestaltet sich daher wie folgt: Nach der Initialisierung wird inkrementell immer ein weiterer Datenpunkt bearbeitet und den verschiedenen Clustern zugeteilt. Anhand der inkrementellen Formeln erfolgt eine sofortige Aktualisierung der Clusterkenngrößen. Um Adaptivität der Cluster selber zu realisieren, können die inkrementellen Formeln derart angepaßt werden, daß in der Vergangenheit bearbeitete Datenpunkte zunehmend geringeren Einfluß haben.

Die COD-Problematik äußert sich beim PNC 2-Algorithmus dadurch, daß die bei Vereinigung von einer bestimmten Anzahl an Datenpunkten entstehenden Eingangsraumgebiete mit zunehmender Dimensionalität kleiner werden. Der verwendete Mechanismus, dieser Problematik mit Hilfe von Gl. (3.17) zu begegnen, stellt einen ersten heuristischen Lösungsversuch dar. Er funktioniert, indem das Volumen des von der zu testenden Regel abgedeckten Eingangsraumbereichs vergrößert wird. Weitergehende Ansätze könnten versuchen, die exakte Größe der abgedeckten Eingangsraumgebiete zu schätzen oder zu berechnen und das Ergebnis zu nutzen, um den in Gl. (3.17) verwendeten Parameter w_{COD} gezielt an die jeweils betrachtete Vereinigung anzupassen.

Ein weiterer Aspekt beim PNC 2-Algorithmus betrifft die Reihenfolge, in der versucht wird, die Cluster gleicher Ausgangsklasse miteinander zu vereinigen. In zahlreichen unüberwacht arbeitenden agglomerativen Clusterverfahren kommen Kriterien zum Einsatz, die in irgendeiner Weise die Dichte oder die Nähe der Datenpunkte zweier Cluster zueinander berücksichtigen. Damit wird versucht, die den Datenpunkten im Eingangsraum inhärente Struktur zu erfassen. Die beim PNC 2 verwendete Strategie, immer dasjenige Clusterpaar auszuwählen, bei dem die Außenkanten der Hyperquader den geringsten Abstand zueinander aufweisen, entspricht am ehesten der Strategie des Single-Linkage-Algorithmus, der dafür bekannt ist, daß er tendenziell eher zum Auffinden von langgezogenen, kettenartigen Clustern geeignet ist. Der Complete-Linkage-Algorithmus hingegen produziert eher kompakte Cluster, weshalb die in ihm verwendete Strategie auf den PNC 2 übertragen werden könnte, um damit die Vereinigungsreihenfolge geschickter festzulegen. Auch wäre es sinnvoll, beim PNC 2 die allgemeine Anwendbarkeit der Kriterien weiterer agglomerativer Clusterverfahren systematisch zu untersuchen.

Beim SMBC-Algorithmus findet zur Zeit – abgesehen vom Mechanismus zum Löschen von Clustern – keine Rückkopplung der Relevanz und der Größe der Cluster in den Clusterprozeß statt. Der Gath-Geva-Algorithmus (GG) [GG89] verwendet im Gegensatz zum GK-Algorithmus Prototypen, die ellipsoide Cluster unterschiedlicher Größe implizieren. Die hier entwickelten Strategieelemente könnten, statt mit dem FCM- oder GK-Algorithmus, auch problemlos mit dem GG-Algorithmus zusammen verwendet werden. Ob und inwieweit dies zu einer Verbesserung der erzielbaren Prognosegüten führen würde ist zu untersuchen.

Eine weitere, hier nicht näher untersuchte, Einsatzmöglichkeit für den SMBC-Algorithmus ergibt sich aus der Nutzung der in [Kie98] beschriebenen Querverbindung zwischen Clusterverfahren und Kohonenkarten. Die vorgestellten Methoden zur Überwachung des Clusterprozesses durch Rückkopplung der Ausgangsgrößenwerte und zum Hinzufügen, Löschen und Vereinigen von Clustern könnten dazu führen, daß sich die Neuronen bei der – dann in der Größe dynamischen – Kohonenkarte bezüglich der Homogenität der Ausgangsgrößenwerte besser positionieren.

Anhang A

Benchmarkübersicht

Die folgenden Aufzählungen beschreiben knapp die im Rahmen dieser Arbeit verwendeten Benchmarkbeispiele. In Tabelle A.1 findet sich ein Überblick mit Angaben wie beispielsweise der Anzahl der Ausgangsklassen S_y oder der Anzahl der Datenpunkte N und der Eingangsgrößen m. Die für die Quellen verwendeten Kürzel sind in Tabelle A.2 aufgeschlüsselt.

Klassifikationsaufgaben

- **Australian** Überprüfung von Kreditkartentransaktionen aufgrund von anonymisierten Charakteristika der Transaktion. Bei den ursprünglichen Daten enthielten etwa 5% der Datenpunkte ein oder mehrere fehlende Eingangsgrößenwerte. Diese wurden durch den Mittelwert bzw. das auf häufigsten auftretende Symbol ersetzt.
- **Chess-End-Game** Anhand einer gegebenen Situation der Figuren auf einem Schachbrett soll entschieden werden, ob der Spieler *Weiß* die Partie gewinnen kann.
- **Cleveland** Anhand von Patientendaten soll klassifiziert werden, ob ein Patient an einer Herzkrankheit leidet. Die ursprüngliche Ausgangsgröße nimmt ganzzahlige Werte von 0 bis 4 an. 0 bedeutet, der Patient ist gesund, 1 bis 4 bedeutet, der Patient leidet an einer Herzkrankheit, wobei diese umso stärker ausgeprägt ist, je höher der Ausgangsgrößenwert ist.
- **DNA** Klassifikation von Übergängen in Nukleotidsequenzen. Anhand eines Fensters von 60 Nukleotidsequenzen soll entschieden werden, ob sich in der Mitte des Fensters ein Intron-Exon-Übergang, ein Exon-Intron-Übergang oder aber kein Übergang befindet.[1]
- **Hepatitis** Anhand von Patientendaten soll entschieden werden, ob der Patient an Hepatitis leidet.
- **House-Votes-84** Klassifikation der Parteizugehörigkeit von Kongreßabgeordneten anhand ihres Abstimmverhaltens.
- **Iris** Klassifikation der Art von Schwertlilien anhand der Länge und Breite der Kelch- und Blumenblätter.
- **Landsat Satellite** Klassifikation der Oberflächenbeschaffenheit anhand von aus Satellitenbildern berechneten Merkmalen.
- **Letter-Recognition** Klassifikation von in 20 verschiedenen Fonts angezeigten Buchstaben anhand von berechneten Merkmalen.
- **Mushrooms** Klassifikation ob ein Pilz definitiv eßbar, definitiv giftig oder von unbekannter Herkunft ist. Der letztere Fall wird ebenfalls den giftigen Pilzen zugerechnet.
- **Segmentation** Klassifikation der Oberflächenbeschaffenheit anhand von aus Satellitenbildern berechneten Merkmalen.
- **Waveform** Klassifikation, aus welchen zwei Basiswellen eine Welle zusammengesetzt ist.
- **Waveform+Noise** Wie *Waveform*, jedoch mit 19 zusätzlichen Eingangsgrößen, die mittelwertfreies Rauschen mit einer Varianz von 1 enthalten.
- **Wine** Klassifikation von Weinsorten anhand der Ergebnisse von chemischen Analysen.

[1] Der hier verwendete Datensatz von Delve entstand aus dem vom UCI, indem 15 Datenpunkte entfernt wurden.

Regressionsaufgaben

- **Kin32fm** Prognose des Abstandes des Endeffektors einen 8 achsigen Roboters vom Zielpunkt.
- **Mackey-Glass** Prognose einer chaotischen Zeitreihe [MG77]. Die Zeitreihe wird beschrieben durch $x(t+1) = 0.9x(t) + \frac{0.2x(t-17)}{1+x(t-17)^{10}}$. Für $t < 0$ ist $x(t) = 0$, und für $t = 0$ ist $x(t) = 1.2$. Die Eingangsgrößen sind $x(t-18)$, $x(t-12)$, $x(t-6)$ und $x(t)$. Die Ausgangsgröße ist $x(t+6)$.
- **Wärmetauscher** Prognose der Temperatur eines Wärmetauschers für einen Batch-Reaktor.

Tabelle A.1: Benchmark Übersicht.

Benchmark	Kürzel/Bezeichnung	S_y	N	m	nom	cont	fehlende Werte	Quelle
Australian	Australian	2	690	14	8	6	Nein	UCI
Cleveland	Cleveland	2	303	13	9	4	Ja	UCI
Chess-End-Game	Chess	2	3196	36	36	0	Nein	UCI
DNA	Dna	3	3175	60	0	60	Nein	Delve
Hepatitis	Hepatitis	2	155	19	13	6	Ja	UCI
House-Votes-84	Votes	2	435	16	16	0	Ja	UCI
Iris	Iris	3	150	4	0	4	Nein	UCI
Landsat Satellite	Sat	6	6435	36	0	36	Nein	UCI
Letter-Recognition	Letter	26	20000	16	0	16	Nein	UCI
Mushrooms	Mushrooms	2	8124	22	22	0	Ja	UCI
Segmentation	Seg	7	2130	19	0	19	Nein	UCI
Waveform	Wave	3	5000	21	0	21	Nein	UCI
Waveform+Noise	Wave+Noise	3	5000	40	0	40	Nein	UCI
Wine	Wine	3	178	13	0	13	Nein	UCI
Kin32fm	Kin32fm	–	8192	32	32	0	Nein	Delve
Mackey-Glass	Mackey	–	1000	4	0	4	Nein	–
Wärmetauscher	Temp	–	1126	3	0	3	Nein	ESR

Anmerkung: *nom* und *cont* bezeichnen die Anzahl nominaler bzw. kontinuierlicher Eingangsgrößen

Tabelle A.2: Quellen.

Abkürzung	Beschreibung
Delve	Benchmarksammlung und definierte Vorgehensweisen zur Durchführung von Experimenten von der Universität Toronto [RNH+96]. Delve steht für *Data for Evaluation of Learning in Valid Experiments*. (`http://www.cs.utoronto.ca/~delve/`)
ESR	Innerhalb des Lehrstuhls verwendeter Benchmark.
UCI	*UCI Machine Learning Repository*: Benchmarksammlung der Universität Kalifornien, Irvine [BM98].

Anhang B

Ergebnisse der Experimente ExpA

Die Tabellen B.2 bis B.6 enthalten die Ergebnisse der in Kapitel 5.3.2 beschriebenen Experimente ExpA. Tabelle B.1 erläutert die angegeben Werte.

Tabelle B.1: Erläuterungen zu den Tabellen B.2 bis B.6.

Kürzel	**Erläuterung**
$\widehat{Q}_T$	Prognosegüte auf Testdaten; MCE für Klassifikations- und MAE für Regressionsaufgaben
$\widehat{Q}_L$	Prognosegüte auf Lerndaten; s.o.
K	Anzahl Cluster
K_{Red}	Anzahl Cluster nach Reduktion aller Cluster, deren Masse kleiner oder gleich dem Schwellwert p_{min} ist
ø m	über alle nicht reduzierten Cluster ermittelte durschnittliche Anzahl betrachteter Eingangsgrößen
Covered	Anteil an Testdatenpunkten, deren Eingangsraumposition innerhalb eines oder mehrerer Hyperquader der generierten Cluster liegt
ø TQ	über alle Cluster ermittelte durschnittliche Trefferquote
P_{H0}	Irrtumswahrscheinlichkeit bei zweiseitigem t-Test für gepaarte Ereignisse, d.h. Wahrscheinlichkeit der Null-Hypothese, daß der beobachtete Unterschied der mit der Default-Variante verglichenen Prognosegüte zufällig ist. Siehe dazu auch die Beschreibung des t-Tests im Anhang C.
Vergl.	relative Veränderung der Prognosegüte im Vergleich zur Default-Variante

Tabelle B.2: Ergebnisse ExpA für Benchmark Australian.

Alternative	$\widehat{Q}_T$	$\widehat{Q}_L$	K	K_{Red}	ø m	Covered	ø TQ	P_{H0}	Vergl.
Default	**14.61 %**	14.21 %	88.2	7.1	4.2	72.9 %	–	–	–
	± 1.41 %	± 1.40 %	± 20.4	± 5.7	± 1.2	± 12.8 %	–		
$\delta_{symb} = 1$	14.75 %	14.29 %	88	6.4	4.0	74.2 %	–	0.136	- 1.0 %
	± 1.56 %	± 1.77 %	± 22.5	± 5.7	± 1.2	± 13.0 %	–		
4σ-Normierung	14.62 %	14.18 %	88.3	7.3	4.3	73.4 %	–	0.324	- 0.1 %
	± 1.39 %	± 1.40 %	± 21.6	± 5.7	± 1.1	± 12.7 %	–		
d_{avr}-Normierung	14.64 %	14.12 %	93.0	8.8	4.5	69.2 %	–	0.085	- 0.2 %
	± 1.38 %	± 1.36 %	± 24.8	± 7.7	± 1.3	± 17.3 %	–		
Einfacher	**14.53 %**	14.44 %	44.0	2.7	2.1	100.0 %	83.9 %	0.029	0.5 %
Vereinigungstest	± 1.44 %	± 1.43 %	± 4.3	± 0.7	± 1.1	± 0.3 %	± 3.2 %		
MTB	**14.61 %**	14.21 %	88.2	7.1	4.2	72.9 %	–	1.000	0.0 %
	± 1.41 %	± 1.40 %	± 20.4	± 5.7	± 1.2	± 12.8 %	–		
Laplace korrigierte	**14.60 %**	14.20 %	88.2	7.1	4.2	72.9 %	85.4 %	0.208	0.0 %
TQ	± 1.41 %	± 1.40 %	± 20.4	± 5.7	± 1.2	± 12.8 %	± 3.0 %		
Konfidente TQ	**14.59 %**	14.20 %	88.2	7.1	4.2	72.9 %	55.4 %	0.200	0.1 %
	± 1.40 %	± 1.37 %	± 20.4	± 5.7	± 1.2	± 12.8 %	± 13.6 %		
TQ	**14.61 %**	14.16 %	88.2	7.1	4.2	72.9 %	94.6 %	0.283	0.0 %
	± 1.43 %	± 1.39 %	± 20.4	± 5.7	± 1.2	± 12.8 %	± 2.3 %		

Äquidistante Diskretisierung	**14.59 %** ± 1.43 %	14.28 % ± 1.49 %	115.0 ± 40.0	7.7 ± 6.4	4.2 ± 1.8	64.2 % ± 17.6 %	– –	0.341	0.1 %
Keine Gewichtsfaktoren	15.02 % ± 1.84 %	13.94 % ± 1.58 %	94.1 ± 25.7	9.6 ± 7.6	4.7 ± 1.1	68.6 % ± 17.6 %	– –	0.003	- 2.8 %
Keine CSFS	**14.53 %** ± 1.40 %	14.22 % ± 1.40 %	88.2 ± 20.4	7.1 ± 5.7	14.0 ± 0.0	67.0 % ± 12.6 %	– –	0.001	0.5 %

Tabelle B.3: Ergebnisse ExpA für Benchmark Hepatitis.

Alternative	$\widehat{Q}_T$	$\widehat{Q}_L$	K	K_{Red}	ø m	Covered	ø TQ	P_{H0}	Vergl.
Default	**18.40 %** ± 4.08 %	9.41 % ± 3.29 %	12.1 ± 2.4	5.7 ± 1.7	4.3 ± 0.8	67.9 % ± 11.9 %	– –		
$\delta_{symb} = 1$	19.35 % ± 4.04 %	9.93 % ± 3.00 %	12.5 ± 2.9	5.8 ± 1.8	4.4 ± 0.8	65.8 % ± 12.3 %	– –	0.006	- 5.1 %
4σ-Normierung	18.97 % ± 4.07 %	9.06 % ± 3.25 %	12.5 ± 2.6	6.1 ± 1.7	4.2 ± 0.7	65.6 % ± 12.0 %	– –	0.045	- 3.1 %
d_{avr}-Normierung	19.16 % ± 4.03 %	9.15 % ± 2.96 %	12.5 ± 2.8	6.1 ± 1.7	4.5 ± 0.8	66.0 % ± 12.3 %	– –	0.022	- 4.1 %
Einfacher Vereinigungstest	20.33 % ± 3.64 %	20.23 % ± 3.58 %	5.7 ± 1.1	3.1 ± 0.7	3.5 ± 0.9	99.4 % ± 2.3 %	76.5 % ± 5.0 %	0.000	- 10.5 %
MTB	18.44 % ± 4.13 %	9.51 % ± 3.29 %	12.1 ± 2.4	5.7 ± 1.7	4.3 ± 0.8	67.9 % ± 11.9 %	– –	0.160	- 0.2 %
Laplace korrigierte TQ	**18.31 %** ± 4.13 %	8.89 % ± 2.97 %	12.1 ± 2.4	5.7 ± 1.7	4.3 ± 0.8	67.9 % ± 11.9 %	83.8 % ± 1.6 %	0.035	0.5 %
Konfidente TQ	**18.11 %** ± 4.21 %	9.26 % ± 3.36 %	12.1 ± 2.4	5.7 ± 1.7	4.3 ± 0.8	67.9 % ± 11.9 %	44.4 % ± 5.7 %	0.019	1.6 %
TQ	18.45 % ± 4.22 %	7.52 % ± 3.05 %	12.1 ± 2.4	5.7 ± 1.7	4.3 ± 0.8	67.9 % ± 11.9 %	96.3 % ± 2.5 %	0.227	- 0.3 %
Äquidistante Diskretisierung	**18.23 %** ± 4.59 %	8.99 % ± 3.27 %	12.7 ± 2.6	5.9 ± 1.7	4.2 ± 0.8	66.5 % ± 12.1 %	– –	0.322	0.9 %
Keine Gewichtsfaktoren	18.92 % ± 4.26 %	8.50 % ± 3.18 %	13.2 ± 2.7	6.7 ± 2.0	4.5 ± 0.8	61.1 % ± 11.9 %	– –	0.091	- 2.8 %
Keine CSFS	**17.89 %** ± 4.12 %	9.39 % ± 3.01 %	12.1 ± 2.4	5.7 ± 1.7	19.0 ± 0.0	56.3 % ± 12.7 %	– –	0.046	2.8 %

Tabelle B.4: Ergebnisse ExpA für Benchmark Mushrooms.

Alternative	$\widehat{Q}_T$	$\widehat{Q}_L$	K	K_{Red}	ø m	Covered	ø TQ	P_{H0}	Vergl.
Default	**0.67 %** ± 0.50 %	0.21 % ± 0.20 %	4.3 ± 0.6	3.4 ± 0.6	3.3 ± 1.1	99.6 % ± 0.5 %	– –		
4σ-Normierung	**0.67 %** ± 0.50 %	0.21 % ± 0.20 %	4.3 ± 0.6	3.4 ± 0.6	3.3 ± 1.1	99.6 % ± 0.5 %	– –	1.000	0.0 %
d_{avr}-Normierung	0.73 % ± 0.53 %	0.25 % ± 0.22 %	4.3 ± 0.7	3.3 ± 0.6	3.2 ± 1.1	99.7 % ± 0.5 %	– –	0.008	- 9.6 %
einfacher Vereinigungstest	3.82 % ± 2.10 %	3.27 % ± 2.00 %	3.2 ± 0.4	3.0 ± 0.3	3.7 ± 0.8	99.8 % ± 0.4 %	93.4 % ± 2.8 %	0.000	- 470.7 %
MTB	0.93 % ± 0.54 %	0.62 % ± 0.44 %	4.3 ± 0.6	3.4 ± 0.6	3.3 ± 1.1	99.6 % ± 0.5 %	– –	0.000	- 39.1 %
Laplace korrigierte TQ	0.93 % ± 0.56 %	0.59 % ± 0.41 %	4.3 ± 0.6	3.4 ± 0.6	3.3 ± 1.1	99.6 % ± 0.5 %	95.5 % ± 2.3 %	0.000	- 38.5 %
Konfidente TQ	0.97 % ± 0.54 %	0.61 % ± 0.45 %	4.3 ± 0.6	3.4 ± 0.6	3.3 ± 1.1	99.6 % ± 0.5 %	82.9 % ± 8.7 %	0.000	- 44.8 %
TQ	0.68 % ± 0.51 %	0.22 % ± 0.21 %	4.3 ± 0.6	3.4 ± 0.6	3.3 ± 1.1	99.6 % ± 0.5 %	99.3 % ± 0.6 %	0.160	- 1.2 %
Äquidistante Diskretisierung	**0.67 %** ± 0.50 %	0.21 % ± 0.20 %	4.3 ± 0.6	3.4 ± 0.6	3.3 ± 1.1	99.6 % ± 0.5 %	– –	1.000	0.0 %
Keine Gewichtsfaktoren	0.75 % ± 0.53 %	0.23 % ± 0.19 %	4.3 ± 0.7	3.3 ± 0.7	3.3 ± 1.2	99.6 % ± 0.5 %	– –	0.008	- 11.9 %
Keine CSFS	**0.65 %** ± 0.53 %	0.21 % ± 0.21 %	4.3 ± 0.6	3.4 ± 0.6	22.0 ± 0.0	98.3 % ± 0.8 %	– –	0.239	3.3 %

Tabelle B.5: Ergebnisse ExpA für Benchmark Seg.

Alternative	$\widehat{Q}_T$	$\widehat{Q}_L$	K	K_{Red}	ø m	Covered	ø TQ	P_{H0}	Vergl.
Default	**4.82 %**	1.80 %	84.8	40.2	4.8	76.8 %	–		
	± 0.77 %	± 0.53 %	± 7.1	± 3.4	± 0.2	± 1.5 %	–		
4σ-Normierung	5.02 %	1.90 %	84.8	40.1	4.8	77.0 %	–	0.041	- 4.2 %
	± 0.78 %	± 0.46 %	± 6.9	± 3.9	± 0.2	± 1.6 %	–		
d_{avr}-Normierung	5.06 %	1.84 %	85.3	41.5	4.8	77.1 %	–	0.050	- 5.0 %
	± 0.72 %	± 0.60 %	± 6.2	± 3.2	± 0.2	± 1.2 %	–		
Einfacher	5.44 %	2.47 %	52.2	24.6	5.3	86.0 %	78.4 %	0.000	- 13.0 %
Vereinigungstest	± 0.78 %	± 0.47 %	± 10.3	± 2.1	± 0.2	± 1.7 %	± 1.7 %		
MTB	4.83 %	1.92 %	84.8	40.2	4.8	76.8 %	–	0.280	- 0.2 %
	± 0.77 %	± 0.53 %	± 7.1	± 3.4	± 0.2	± 1.5 %	–		
Laplace korrigierte	4.90 %	1.88 %	84.8	40.2	4.8	76.8 %	63.8 %	0.091	- 1.7 %
TQ	± 0.82 %	± 0.51 %	± 7.1	± 3.4	± 0.2	± 1.5 %	± 1.8 %		
Konfidente TQ	4.89 %	1.84 %	84.8	40.2	4.8	76.8 %	58.5 %	0.133	- 1.4 %
	± 0.85 %	± 0.52 %	± 7.1	± 3.4	± 0.2	± 1.5 %	± 2.5 %		
TQ	**4.74 %**	1.62 %	84.8	40.2	4.8	76.8 %	95.6 %	0.016	1.6 %
	± 0.75 %	± 0.51 %	± 7.1	± 3.4	± 0.2	± 1.5 %	± 0.8 %		
Äquidistante	5.21 %	2.08 %	82.3	36.3	3.8	81.1 %	–	0.010	- 8.1 %
Diskretisierung	± 0.86 %	± 0.61 %	± 7.5	± 3.4	± 0.2	± 1.8 %	–		
Keine	4.94 %	1.70 %	88.5	43.6	5.0	76.2 %	–	0.244	- 2.6 %
Gewichtsfaktoren	± 0.69 %	± 0.32 %	± 6.9	± 2.8	± 0.2	± 1.5 %	–		
Keine CSFS	**4.76 %**	1.74 %	84.8	40.2	19.0	65.6 %	–	0.324	1.2 %
	± 0.97 %	± 0.47 %	± 7.1	± 3.4	± 0.0	± 1.8 %	–		

Tabelle B.6: Ergebnisse ExpA für Benchmark Temp.

Alternative	$\widehat{Q}_T$	$\widehat{Q}_L$	K	K_{Red}	ø m	Covered	ø TQ	P_{H0}	Vergl.
Default	**1.436**	1.048	55.3	27.6	1.4	86.3 %	–		
	± 0.381	± 0.277	± 6.5	± 4.4	± 0.1	± 1.8 %	–		
Keine	2.177	1.833	55.3	27.6	1.4	86.3 %	–	0.000	- 51.7 %
Neuberechnung c	± 0.436	± 0.393	± 6.5	± 4.4	± 0.1	± 1.8 %	–		
4σ-Normierung	**1.421**	1.043	55.3	27.5	1.4	86.3 %	–	0.002	1.0 %
	± 0.376	± 0.271	± 6.4	± 4.4	± 0.1	± 1.9 %	–		
d_{avr}-Normierung	**1.386**	1.055	55.4	27.5	1.4	86.2 %	–	0.002	3.5 %
	± 0.383	± 0.276	± 6.4	± 4.4	± 0.1	± 1.9 %	–		
Einfacher	1.943	1.555	37.9	22.5	1.4	93.5 %	87.1 %	0.002	- 35.3 %
Vereinigungstest	± 0.912	± 0.779	± 3.3	± 3.1	± 0.1	± 2.4 %	± 2.0 %		
MTB	**1.436**	1.048	55.3	27.6	1.4	86.3 %	–	1.000	0.0 %
	± 0.381	± 0.277	± 6.5	± 4.4	± 0.1	± 1.8 %	–		
Laplace korrigierte	1.464	1.075	55.3	27.6	1.4	86.3 %	37.3 %	0.000	- 2.0 %
TQ	± 0.390	± 0.274	± 6.5	± 4.4	± 0.1	± 1.8 %	± 2.5 %		
Konfidente TQ	1.467	1.076	55.3	27.6	1.4	86.3 %	45.9 %	0.000	- 2.2 %
	± 0.391	± 0.275	± 6.5	± 4.4	± 0.1	± 1.8 %	± 3.7 %		
TQ	1.437	1.047	55.3	27.6	1.4	86.3 %	97.5 %	0.121	- 0.1 %
	± 0.380	± 0.276	± 6.5	± 4.4	± 0.1	± 1.8 %	± 0.8 %		
Äquidistante	**1.434**	1.048	55.3	27.5	1.4	86.2 %	–	0.343	0.1 %
Diskretisierung	± 0.373	± 0.274	± 6.5	± 4.5	± 0.1	± 1.8 %	–		
Keine	1.440	1.037	54.7	27.6	1.4	86.5 %	–	0.333	- 0.3 %
Gewichtsfaktoren	± 0.363	± 0.260	± 6.2	± 4.1	± 0.1	± 1.8 %	–		
Keine CSFS	**1.414**	1.074	55.3	27.6	3.0	82.7 %	–	0.153	1.5 %
	± 0.367	± 0.283	± 6.5	± 4.4	± 0.0	± 2.4 %	–		

Anhang C

t-Test für gepaarte Ereignisse

Der t-Test für gepaarte Ereignisse[1] [PTVF93] (S. 618)[2] gibt Auskunft darüber, ob die Mittelwerte zweier Größen Q_a und Q_b signifikant voneinander verschieden sind. Die Ereignisse der beiden Größen müssen dabei miteinander *gepaart* sein, also beispielsweise den Meßwerten von zwei verschiedenen Sensoren zur Bewertung ein und desselben Objektes entsprechen. Hier sind Q_a und Q_b die Prognosegüten zweier Modelle, die sich bei den durchgeführten N_R-Aufteilungen des betrachteten Benchmarks in Lern- und Teststichproben ergeben. Der Wert der t-Statistik berechnet sich zu

$$t = \frac{\overline{Q}_a - \overline{Q}_b}{s} \tag{C.1}$$

mit

$$s = \sqrt{\frac{\sigma^2_{Q_a} + \sigma^2_{Q_b} - 2\,Cov(Q_a, Q_b)}{N_R}} \tag{C.2}$$

und

$$Cov(Q_a, Q_b) = \frac{1}{N_R - 1} \sum_{w=1}^{N_R} (Q_{a_w} - \overline{Q}_a)(Q_{b_w} - \overline{Q}_b), \tag{C.3}$$

wobei $\overline{Q}_a$ bzw. $\overline{Q}_b$ den arithmetischen Mittelwert und $\sigma^2_{Q_a}$ bzw. $\sigma^2_{Q_b}$ die Varianz der jeweiligen Größe bezeichnen. Die Wahrscheinlichkeit der Null-Hypothese P_{H0} wird aus dem ermittelten Wert der t-Statistik mit einem Freiheitsgrad von $\nu = N_R - 1$ mit den Quantilen der Studentschen t-Verteilung bei zweiseitiger Fragestellung ausgewertet zu

$$P_{H0} = I_{\frac{\nu}{\nu+t^2}}\left(\frac{\nu}{2}, 0.5\right) \tag{C.4}$$

mit der sogenannten *Incomplete Beta Function* [PTVF93] (S. 226ff.)

$$I_x(a,b) = \frac{1}{B(a,b)} \int_0^x t^{a-1}(1-t)^{b-1} dt \tag{C.5}$$

und der sogenannten *Beta Function* [PTVF93] (S. 215f.)

$$B(a,b) = \int_0^1 t^{b-1}(1-t)^{a-1} dt. \tag{C.6}$$

[1] engl.: *Student's t-test in the case of paired samples*

[2] In dem angegeben Werk sind die in der Programmiersprache *C* geschriebenen Quelltexte für den Test und die dabei benötigten Funktionen zu finden.

Anhang D

Symbole, Abkürzungen und Indizes

Tabelle D.1: Abkürzungen.

Abkürzung	Bedeutung
COD	Curse of Dimensionality
COG	Center of Gravity
CSFS	Context-Sensitive Feature-Selection
HEOM	Heterogeneous-Euclidean-Overlap-Metric
HVDM	Heterogeneous-VDM
IBL	Instance-Based-Learning
IVDM	Interpolated-VDM
LOOCV	Leave-One-Out Cross-Validation
MAE	Mean Absolute Error
MCE	Mean Classification Error
MFC	Most Frequent Class
MND	Mutual-Neighbor-Distance
MOM	Mean of Maximum
MTB	Mass Tie Breaking
MDLP	Minimum Description Length Principle
MSE	Mean Square Error
MMC	Mean Misclassification Costs
OCR	Optimizing Conflict Resolution
ROSA	Rule Oriented Statistical Analysis
TQ	Trefferquote
TSK	Takagi-Sugeno-Kang *Modell*
VDM	Value-Difference-Metric
ZGF	Zugehörigkeitsfunktion

Tabelle D.2: Lernalgorithmen.

Abkürzung	Bedeutung
BNGE	Batch-NGE
FCM	Fuzzy-C-Means
FR	Fuzzy-ROSA
GK	Gustafson-Kessel
GG	Gath-Geva
*k*NN	*k*-Nearest-Neighbor
NGE	Nearest-Generalized-Exemplar
NN	Nearest-Neighbor
OBNGE	Overlapping-Batch-NGE
PNC 2	Positive and Negative Example Based Clustering
RISE	Rule Induction from a Set of Exemplars
SMBC	Supervised Model Based Clustering

Tabelle D.3: Spezielle Begriffe.

Begriff	Bedeutung
Stichprobe	Menge von Datenpunkten
Datenpunkt	Im Kontext der datenbasierten Modellierung ist ein Datenpunkt ein Tuple aus einem Eingangsvektor und einer Ausgangsgröße. Im Kontext unüberwachter Clusterverfahren hingegen handelt es sich nur um einen Datenvektor.
Eingangsvektor	Vektor mit m Eingangsgrößen – Im Kontext der Anwendung eines Modells zur Prognose ohne Bezug zu einer konkreten Stichprobe wird auch der Begriff *Eingangsraumposition* verwendet.
Datenvektor	Vektor mit m Größen

Tabelle D.4: Spezielle Zähler und Indizes.

Zähler	Index	Verwendungskontext	Beispiel
N	i	Datenpunkte	Datenpunkt P_i
m	j	(Eingangs-)Größen	(Eingangs-)Größe x_j
K	t	Cluster	Cluster C_t
S_j, S_x	w, z	Symbole nominaler oder diskretisierter Eingangsgrößen	Symbol w bzw. z
S_y	s	Symbole nominaler oder diskretisierter Ausgangsgrößen	Symbol s
–	T, L	Test- bzw. Lernstichprobe	Prognosegüte Q_T und Q_L
–	a, b	zwei Elemente einer Menge	Datenpunkte P_a und P_b
–	w, z	allgemeine Indizes für Matrizenelemente o.ä.	

Tabelle D.5: Weitere global verwendete Symbole.

Symbol	Bedeutung
$\mathbf{b}$	Bitstring mit den von einem Hyperquader bezüglich einer nominalen Eingangsgröße abgedeckten Symbolen
C	Cluster
c	Clusterausgangsgrößenwert oder Konklusion
c^*	mittels der Pseudo-Inversen neu berechneter Clusterausgangsgrößenwert (SMBC-Algorithmus)
$\mathbf{B}$	Fehlklassifizierungskostenmatrix mit den Elementen $b_{w,z}$
e	Prognosefehler
$\mathcal{C}$	Clusterpopulation: Menge mit K Clustern
d	Abstand zweier Datenpunkte bzw. Cluster
d_{avr}	durchschnittlicher komponentenweiser Abstand einer Eingangsgröße
d_j	komponentenweiser Abstand zweier Datenpunkte bzw. Cluster bezüglich der Eingangsgröße x_j
E	Exemplar (NGE-Algorithmus)
$E(y)$	Entropie der Ausgangsgröße
$f_{tf}(\mathbf{x})$	zugrundeliegender funktionaler Zusammenhang (engl.: *true function*)
$g(\mathbf{x})$	Teilfunktion bei RBF-Netz
H	Hyperquader (PNC 2-Algorithmus)
$I(x, y)$	Transinformation einer Eingangsgröße zur Ausgangsgröße
l	linke Grenze einer Hyperquaderkomponente (kontinuierliche Eingangsgrößen)
k	Parameter beim kNN-Algorithmus
K_{Red}	Anzahl von Clustern nach Reduktion (PNC 2-Algorithmus)
N_{Bins}	Anzahl Diskretisierungsintervalle für kontinuierliche Eingangsgrößen
N_R	Anzahl Wiederholungen/Kreuz-Validierungen bei Ermittlung der Prognosegüte $\widehat{Q}$
N_{Tune}	Anzahl Wiederholungen oder Kreuz-Validierungen beim Parametertuning
$P(\mathbf{x}, y)$	Datenpunkt
P_{H0}	Wahrscheinlichkeit der Null-Hypothese

$P(c\|p)$, $\widehat{P}(c\|p)$	bedingte Auftrittswahrscheinlichkeit der Konklusion bei erfüllter Prämisse
p	Clustermasse bzw. Anzahl positiver Lernbeispiele eines Clusters (PNC 2-Algorithmus) oder Regelprämisse
$\mathcal{P}$	Stichprobe: Menge von N Datenpunkten
n	Anzahl negativer Lernbeispiele eines Clusters (PNC 2-Algorithmus)
$\mathbf{O}$	Konfusionsmatrix mit den Elementen $o_{w,z}$
$\mathbf{Q}$	Geometrieparametermatrix eines Clusters (SMBC-Algorithmus)
Q^*, $\widehat{Q}$	tatsächliche bzw. geschätzte Güte eines Lernverfahrens
Q_P^*, $\widehat{Q}_P$	tatsächliche bzw. geschätzte Güte eines Modells
r	rechte Grenze einer Hyperquaderkomponente (kontinuierliche Eingangsgrößen)
$\mathbf{U}$	Partitions- bzw. Zuteilungsmatrix bestehend aus den Elementen $u_{i,t}$
V^α	Konfidenzintervall zu gegebener Irrtumswahrscheinlichkeit α
$\mathbf{v}$	Clusterzentrum (SMBC-Algorithmus)
W_{Kernel}	Parameter PNC 2-Algorithmus
$W_{Kernel\,Min}$	Parameter PNC 2-Algorithmus
$\mathbf{x}$	Eingangsvektor oder Datenvektor
$\overline{x}$	arithmetischer Mittelwert einer Eingangsgröße
$\widetilde{x}$	durch Normierung transformierte Eingangsgröße
x_{max}	maximaler Wert einer Eingangsgröße
x_{min}	minimaler Wert einer Eingangsgröße
x_{range}	Spannweite einer Eingangsgröße
y	Ausggangsgrößenwert
$\widehat{y}$	zu gegebener Eingangsraumposition prognostizierter Ausgangsgrößenwert
α	Irrtumswahrscheinlichkeit der konfidenten Trefferquote oder Parameter SMBC-Algorithmus
β	Parameter SMBC-Algorithmus
γ	Parameter SMBC-Algorithmus
δ	Normierungsfaktor einer Eingangsgröße
η	Parameter PNC 2-Algorithmus
θ	Schwellwert bei auf der Trefferquote basierenden Regeltests
$\mu(\mathbf{x})$	Erfülltheitsgrad der Prämisse einer Fuzzy-Regel
$\mu(\mathbf{x}, y)$	ausgangsseitiges Empfehlungsgebirge
$\mu(y)$	Erfülltheitsgrad der Konklusion einer Fuzzy-Regel
$\mu_{tf}(\mathbf{x})$	Verteilungsfunktion der Eingangsraumpositionen
ν	Anzahl der Freiheitsgrade
ξ	Abdeckungsgrad einer Hyperquaderkomponente (PNC 2-Algorithmus)
ρ	Parameter der Minkowski-Metrik
ϱ	Relevanz/Trefferquote eines Clusters bzw. einer Regel
σ_Q	Standardabweichung Prognosegüte
σ_Q^*	Standard-Fehler der Schätzung der Prognosegüte
σ_j	Standardabweichung der Eingangsgröße x_j
σ_y	Standardabweichung der Ausgangsgröße
σ_{ZGF}	Parameter PNC 2-Algorithmus
υ	Parameter PNC 2-Algorithmus
ω_{COD}	Parameter PNC 2-Algorithmus
Ω	Modifikationsmatrix bestehend aus den Elementen $\omega_{i,t}$ (SMBC-Algorithmus)
ω	Gewichtsfaktor einer Eingangsgröße

Literaturverzeichnis

[Aha90] Aha, David W. *A study of instance-based learning algorithms for supervised learning tasks: Mathematical, empirical, and psychological evaluations.* 1990

[Aha95a] Aha, David: An implementation and experiment with the nested generalized exemplars algorithm / Naval Research Laboratory. 1995 (AIC–95–003). – Forschungsbericht

[Aha95b] Aha, David W.: Machine learning. In: *5th International Workshop on Artificial Intelligence & Statistics.* Ft. Lauderdale, FL, January 1995. – Tutorial slides

[Aha98] Aha, David W. *Feature weighting for lazy learning algorithms.* 1998

[AHK01] Aggarwal, Charu C. ; Hinneburg, Alexander ; Keim, Daniel A.: On the surprising behavior of distance metrics in high dimensional space. In: *Lecture Notes in Computer Science* 1973 (2001)

[And73] Anderberg, M. R.: *Cluster analysis for applications.* New York : Academic Press Inc., 1973

[Bab98] Babuska, Robert: *Fuzzy modeling for control.* Boston : Kluwer Academic Publisher, 1998

[Bab02] Babuska, Robert: *Fuzzy systems, modeling and identification.* 2002. – Lecture Notes, Delft University of Technology, Department of Electrical Engineering, The Netherlands

[BD67] Ball, G.H. ; D.J., Hall: A clustering technique for summarizing multivariate data. In: *Behavioral Science* (1967), Nr. 12, S. 153–155

[Bel61] Bellman, R. E.: *Adaptive control processes: A guided tour.* Princeton : Princeton University Press, 1961

[Bez81] Bezdek, J.C.: *Pattern recognition with fuzzy objective function algorithms.* New York, USA : Plenum Press, 1981

[BGC97] Burdsall, Ben ; Giraud-Carrier, Christophe: Evolving fuzzy prototypes for efficient data clustering. In: *Proc. 2nd International ICSC Symposium on Fuzzy Logic and Applications (ISFL)*, ICSC Academic Press, February 1997, S. 217–223

[BK99] Bauer, Eric ; Kohavi, Ron: An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. In: *Machine Learning* 36 (1999), Nr. 1–2, S. 105–139

[BM98] Blake, C.L. ; Merz, C.J.: UCI Repository of machine learning databases / Department of Information and Computer Science, University of California. Irvine, 1998. – Forschungsbericht

[BR93] Banfield, J.D. ; Raftery, Adrian E.: Model-based gaussian and non-gaussian clustering. In: *Biometrics* 49 (1993), Nr. 3, S. 803–821

[CH67] Cover, T.M. ; Hart, P.E.: Nearest neighbor pattern classification. In: *IEEE Transactions on Information Theory* Bd. 13, 1967, S. 21–27

[Das91] Dasarathy, Belur V.: Nearest neighbor NN norms: NN pattern classification techniques. In: *IEEE Computer Society Press*. Los Alamitos, 1991, S. 388–397

[DH73] Duda, Richard ; Hart, Peter: *Pattern recognition and scene analysis*. New York : John Wiley and Sons, 1973

[Die98] Dietterich, Thomas G.: Machine-learning research: Four current directions. In: *The AI Magazine* 18 (1998), Nr. 4, S. 97–136

[DKS95] Dougherty, James ; Kohavi, Ron ; Sahami, Mehran: Supervised and unsupervised discretization of continuous features. In: *Proc. International Conference on Machine Learning*, 1995, S. 194–202

[DM98] Deng, Kan ; Moore, Andrew: On the greediness of feature selection algorithms. In: *Proc. International Conference on Machine Learning (ICML)*, 1998

[Dom95] Domingos, Pedro: Rule induction and instance-based learning: A unified approach. In: *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, 1995, S. 1226–1232

[Dom96] Domingos, Pedro: Unifying instance-based and rule-based induction. In: *Proc. International Conference on Machine Learning* Bd. 24, 1996, S. 141–168

[Dom97] Domingos, Pedro: *A unified approach to concept learning*, Department of Information and Computer Science, University of California, Irvine, Diss., 1997

[Eve93] Everitt, B. S.: *Cluster analysis*. London, UK : Edward Arnold Ltd., 1993

[FBY92] Frakes, William B. (Hrsg.) ; Baeza-Yates, Ricardo A. (Hrsg.): *Information retrieval: Data structures & algorithms*. Upper Saddle River, NJ : Prentice-Hall, 1992

[FI89] Fayyad, Usama M. ; Irani, Keki B.: Multi-interval discretization of continuous-valued attributes for classification learning. In: *Proc. 13th International Joint Conference on Artificial Intelligence (IJCAI)*, 1989, S. 800–805

[Fle96] Flexer, Arthur: Statistical evaluation of neural network experiments: Minimum requirements and current practice. In: *Proc. 13th European Meeting on Cybernetics and Systems Research* Bd. 2. A–1010 Vienna, Austria, 1996, S. 1005–1008

[FR98] Fraley, Chris ; Raftery, Adrian E.: How many clusters? Which clustering method? Answers via model-based cluster analysis. In: *The Computer Journal* 41 (1998), Nr. 8, S. 578–588

[Fri96] Fritsch, Martin: *Baumorientierte Regel-Induktionsstrategie für das ROSA-Verfahren zur Modellierung komplexer dynamischer Systeme*, Universität Dormund, Fakultät für Elektrotechnik, Diss., 1996. – VDI Verlag Düsseldorf, Fortschritt-Berichte VDI, Reihe 8, Nr. 565

[Fri97] Fritzke, Bernd: Incremental neuro-fuzzy systems. In: *Applications of Soft Computing, SPIE International Conference*, 1997

[GG89] Gath, I. ; Geva, A.B.: Unsupervised optimal fuzzy clustering. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (1989), July, S. 773–781

[GK79] Gustafson, D. ; Kessel, W.: Fuzzy clustering with a fuzzy covariance matrix. In: *Proc. IEEE CDC* (1979), S. 761–766

[Goo65] Good, I. J.: *The estimation of probabilities: An essay on modern bayesian methods.* Cambridge : MIT Press, 1965

[GSJ97] Gómez-Skarmeta, A.F. ; Jiménez, F.: Generating and tuning fuzzy rules using hybrid systems. In: *Proc. 6th International Conference on Fuzzy Systems (FUZZ-IEEE).* Barcelona, Spain, 1997, S. 247–252

[Hae01a] Haendel, Lars: Data-based lerning of fuzzy-rules using an agglomerative cluster algorithm. In: *Proc. European Symposium on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems (Eunite).* Puerto de la Cruz, Tenerife, Spain, December 2001

[Hae01b] Haendel, Lars: Datenbasierte Generierung von Fuzzy-Regeln mittels eines agglomerativen Clusterverfahrens. In: *Proc. 11. Workshop Fuzzy-Control des GMA-FA 5.22*, 2001

[HAK00] Hinneburg, Alexander ; Aggarwal, Charu C. ; Keim, Daniel A.: What is the nearest neighbor in high dimensional spaces? In: *The VLDB Journal* (2000), S. 506–515

[HC99] Howe, Nicholas ; Cardie, Claire: Weighting unusual feature types / Cornell University. 1999 (TR99–1735). – Forschungsbericht

[HEK95] Hartung, J. ; Elpelt, B. ; Kloesener, K.-H.: *Statistik.* München : Oldenbourg, 1995

[HK00] Höppner, Frank ; Klawonn, Frank: Obtaining interpretable fuzzy models from fuzzy clustering and fuzzy regression. In: *Proc. KES00.* Brighton, UK, 2000, S. 162–165

[HKK97] Höppner, Frank ; Klawonn, Frank ; Kruse, Rudolf: *Fuzzy-Clusteranalyse: Verfahren für die Bilderkennung, Klassifizierung und Datenanalyse.* Braunschweig; Wiesbaden : Vieweg, 1997

[HL96] Hoffbeck, Joseph P. ; Landgrebe, David A.: Covariance matrix estimation and classification with limited training data. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* Bd. 18, 1996, S. 763–767

[HLTM99] Hussain, F. ; Liu, H. ; Tan, C. L. ; M., Dash: Discretization: An enabling technique / School of Computing, National University of Singapore. 1999 (TRC6/99). – Forschungsbericht

[IV94] Imam, Ibrahim ; Vafaie, Haleh: An empirical comparison between global and greedy-like search for variable selection. In: *Florida AI Research Symposium (FLAIRS)*, 1994

[Jes00] Jessen, Holger: *Test- und Bewertungsverfahren zur regelbasierten Modellierung und Anwendung in der Lastprognose*, Universität Dormund, Fakultät für Elektrotechnik, Diss., 2000. – VDI Verlag Düsseldorf, Fortschritt-Berichte VDI, Reihe 8, Nr. 836

[JKP94] John, George H. ; Kohavi, Ron ; Pfleger, Karl: Irrelevant features and the subset selection problem. In: *Proc. International Conference on Machine Learning*, 1994. – Journal version in AIJ, S. 121–129

[JMF00] Jain, A.K. ; Murty, M.N. ; Flynn, P.J.: Data clustering: A review. In: *ACM Computing Surveys* 31 (2000)

[JP73] Jarvis, R.A. ; Patrick, E.A.: Clustering using a similarity measure based on shared near neighbors. In: *IEEE Transaction on Computers* C-22 (1973), November, Nr. 11

[KH02] Kiendl, Harro ; Haendel, Lars: Datenbasierte Regelgenerierung mit modellbasierten Clusterverfahren. In: *Proc. 12. Workshop Fuzzy-Control des GMA-FA 5.22*, 2002

[KHDM98] Kittler, J. ; Hatef, M. ; Duin, R. P. ; Matas, J.: On combining classifiers. In: *Proc. IEEE Transactions on Pattern Analysis and Machine Intelligence* Bd. 20, 1998

[KHJ97] Kantrowitz, Mark ; Horstkotte, Erik ; Joslyn, Cliff: *Answers to frequently asked questions about fuzzy logic and fuzzy expert systems.* 1997. – Official FAQ of the Usenet newsgroup `comp.ai.fuzzy` (ftp.cs.cmu.edu:/user/ai/pubs/faqs/fuzzy/fuzzy.faq)

[Kie97] Kiendl, Harro: *Fuzzy-Control methodenorientiert.* München : Oldenbourg Verlag, 1997

[Kie98] Kiendl, Harro: Self-organising adaptive moment-based clustering. In: *Proc. 7th International Conference on Fuzzy Systems (FUZZ–IEEE)* Bd. 2. Piscataway, NJ : IEEE Press, 1998, S. 1470–1475

[Kin67] King, B.: Step-wise clustering procedures. In: *Journal of the American Statistical Association* 69 (1967), S. 86–101

[KJ97] Kohavi, Ron ; John, George H.: Wrappers for feature subset selection. In: *Artificial Intelligence* 97 (1997), Nr. 1–2, S. 273–324

[KK89] Kiendl, H. ; Krabs, M.: Ein Verfahren zur Generierung regelbasierter Modelle für dynamische Systeme. In: *at – Automatisierungstechnik* 37 (1989), Nr. 11, S. 423–430

[KK97] Klawonn, Frank ; Keller, Annette: Fuzzy clustering and fuzzy rules. In: *Proc. 7th International Fuzzy Systems Association World Congress IFSA'97* Bd. 1, Academica Prague, 1997, S. 193–198

[Koh95] Kohavi, Ron: A study of cross-validation and bootstrap for accuracy estimation and model selection. In: *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, 1995, S. 1137–1145

[Kra01] Krause, Peter: *Datenbasierte Generierung von transpareneten und genauen Fuzzy-Modellen für mehrdeutige Daten und komplexe Systeme*, Universität Dormund, Fakultät für Elektrotechnik, Diss., 2001. – VDI Verlag Düsseldorf, Fortschritt-Berichte VDI, Reihe 10, Nr. 691

[Kro99] Krone, Angelika: *Datenbasierte Generierung von relevanten Fuzzy-Regeln zur Modellierung von Prozesszusammenhängen und Bedienstrategien*, Universität Dormund, Fakultät für Elektrotechnik, Diss., 1999. – VDI Verlag Düsseldorf, Fortschritt-Berichte VDI, Reihe 10, Nr. 615

[KW00] Kojadinovic, Ivan ; Wottka, Thomas: Comparision between a filter and a wrapper approach to variable subset selection in regression problems. In: *Proc. European Symposium on Intelligent Techniques (ESIT)*, 2000

[LWTB97] Liu, W. Z. ; White, A. P. ; Thompson, S. G. ; Bramer, M. A.: Techniques for dealing with missing values in classification. In: *Lecture Notes in Computer Science* 1280 (1997)

[McQ67] McQueen, J.: Some methods for classification and analysis of multivariate observations. In: *Proc. 5th Berkely Symposium on Mathematical Statistics and Probability*, 1967, S. 281–297

[MG77] Mackey, M. ; Glass, L.: Oscillation and Chaos in Physiological Control Systems. In: *Science* 197 (1977), S. 287–289

[MG81] Mamdani, E. H. ; Gaines, B. R.: *Fuzzy reasoning and its applications.* London : Academic Press, 1981

[Mur83] Murtagh, F.: A survey of recent advances in hierarchical clustering algorithms. In: *The Computer Journal* 26 (1983), Nr. 4

[Nad64] Nadaraya, E. A.: On estimating regression. In: *Theory of Probability and Its Application* 10 (1964), S. 186–190

[Nag68] Nagy, G.: State of the art in pattern recognition. In: *Proceedings IEEE* Bd. 56, 1968, S. 836–862

[Nib87] Niblett, T.: Constructing decision trees in noisy domains. In: *Proc. 2nd European Working Session on Learning.* Bled, Yogoslavia, 1987, S. 67–78

[NK97] Nauck, Detlef ; Kruse, Rudolf: What are neuro-fuzzy classifiers? In: *Proc. 7th International Fuzzy Systems Association World Congress IFSA'97* Bd. 4, Academica Prague, 1997, S. 228–233

[Orr96] Orr, Mark J. L.: Introduction to radial basis function networks / Centre for Cognitive Science, University of Edinburgh. 1996. – Forschungsbericht

[PF00] Palmer, Christopher R. ; Faloutsos, Christos: Density biased sampling: an improved method for data mining and clustering. In: *Proc. ACM International Conference on Management of Data (SIGMOD).* Dallas, Texas, United States, 2000, S. 82–92

[PTVF93] Press, W.H. ; Teukolsky, S.A. ; Vetterling, W.T. ; Flannery, B.P.: *Numerical Recipes in C*. Cambridge University Press, Cambridge, UK, 1993. – 2nd Edition

[Qui86] Quinlan, J. R.: Induction of decision trees. In: *Machine Learning* 1 (1986), S. 81–106

[Qui89] Quinlan, J. R.: Unknown attribute values in induction. In: *Proc. 6th International Machine Learning Workshop*, 1989, S. 164–168

[Rez94] Reza, Fazlollah M.: *An introduction to information theory*. New York, Dover, 1994

[RNH+96] Rasmussen, C. E. ; Neal, R. M. ; Hinton, G. E. ; van Camp, D. ; Revow, M. ; Ghahramani, Z. ; Kustra, R. ; Tibshrani, R: The DELVE manual / University of Toronto. 1996. – Forschungsbericht

[RPP99] R. P. Paiva, A. Dourado e B. D.: Applying subtractive clustering for neuro-fuzzy modelling of a bleaching plant. In: *Proc. European Control Conference ECC*, 1999

[RSA00] Roubos, J. ; Setnes, M. ; Abonyi, J.: Learning fuzzy classification rules from data. In: *Proc. RASC Conference*. Leichester, UK, 2000

[Sal91] Salzberg, Steven: A nearest hyperrectangle learning method. In: *Proc. 6th International Conference on Machine Learning*, 1991, S. 251–276

[Sal97] Salzberg, Steven: On comparing classifiers: Pitfalls to avoid and a recommended approach. In: *Data Mining and Knowledge Discovery* 1 (1997), Nr. 3, S. 317–328

[Sal99] Salzberg, Steven: *On comparing classifiers: A critique of current research and methods*. Boston : Kluwer Academic Publishers, 1999

[Sar02] Sarle, Warren S.: *Neural Network FAQ*. 2002. – Official FAQ of the Usenet newsgroup `comp.ai.neural-nets` (ftp://ftp.sas.com/pub/neural/FAQ.html)

[SBBG02] Savaresi, Sergio M. ; Boley, Daniel L. ; Bittani, Sergio ; Gazzaniga, Giovanni: Cluster selection in divisive clustering algorithms. In: *Proc. 2nd SIAM International Conference on Data Mining*, 2002

[Sha48] Shannon, C.E.: A mathematical theory of communication. In: *Bell Systems Technical Journal* 27 (1948), S. 379–423 and 623–656

[Sla01] Slawinski, Timo: *Analyse und effiziente Generierung von relevanten Fuzzy-Regeln in hochdimensionalen Suchräumen*, Universität Dormund, Fakultät für Elektrotechnik, Diss., 2001. – VDI Verlag Düsseldorf, Fortschrittberichte VDI, Reihe 10, Nr. 686

[SNK01] Schauten, Daniel ; Nicolaus, Barbara ; Kiendl, Harro: Evolutionäres Konzept zur Selektion relevanter Merkmalssätze für die datenbasierte Fuzzy-Modellierung. In: *Proc. 11. Workshop Fuzzy-Control des GMA-FA 5.22*, 2001

[SS73] Sneath, P. H. A. ; Sokal, R. R.: *Numerical taxonomy*. San Francisco : W. H. Freeman, 1973

[SW92] Stanfill, Craig ; Waltz, David L.: *Statistical methods, artificial intelligence, and information retrieval*. Lawrence Erlbaum Associates, 1992. – 215–226 S

[TS85] Takagi, T. ; Sugeno, M.: Fuzzy Identification of Systems and its Application to Modeling and Control. In: *IEEE Transactions on Systems, Man, and Cybernetics* 15 (1985), Nr. 1, S. 116–132

[VDI02] VDI/VDE. *Richtlinie VDI/VDE 3550: Fuzzy-Logik und Fuzzy Control, Begriffe und Definitionen.* 2002

[WAM97] Wettschereck, Dietrich ; Aha, David W. ; Mohri, Takao: A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. In: *Artificial Intelligence Review* 11 (1997), Nr. 1–5, S. 273–314

[War63] Ward, J. H.: Hierarchical grouping to optimize an objective function. In: *Journal of American Statistical Association* 58 (1963), S. 236–245

[Wat64] Watson, G.S.: Smooth regression analysis. In: *Sankhya: The Indian Journal of Statsitics Series A* 26 (1964), S. 359–372

[WD95] Wettschereck, Dietrich ; Dietterich, Thomas G.: An experimental comparison of the nearest-neighbor and nearest-hyperrectangle algorithms. In: *Proc. International Conference on Machine Learning* Bd. 19, 1995, S. 5–27

[Wet94] Wettschereck, Dietrich: A hybrid nearest-neighbor and nearest-hyperrectangle algorithm. In: *Proc. European Conference on Machine Learning* Bd. 784, 1994, S. 323–335

[WM96] Wilson, D. R. ; Martinez, Tony R.: Value difference metrics for continuously valued attributes. In: *Proc. Iternational Conference on Artificial Intelligence, Expert Systems and Neural Networks*, 1996, S. 11–14

[WM97a] Wilson, D. R. ; Martinez, Tony R.: Improved heterogeneous distance functions. In: *Journal of Artificial Intelligence Research* 6 (1997), S. 1–34

[WM97b] Wilson, D. R. ; Martinez, Tony R.: Instance pruning techniques. In: *Proc. 14th International Conference on Machine Learning*, Morgan Kaufmann, 1997, S. 403–411

[WM00a] Wilson, D. R. ; Martinez, Tony R.: An integrated instance-based learning algorithm. In: *Computational Intelligence* 16 (2000), Nr. 1, S. 1–28

[WM00b] Wilson, D. R. ; Martinez, Tony R.: Reduction techniques for instance-based learning algorithms. In: *Machine Learning* 38 (2000), Nr. 3, S. 257–286

[Zad65] Zadeh, Lotfi A.: Fuzzy Sets. In: *Information Control* 8 (1965), S. 338–353

[Zad68] Zadeh, Lotfi A.: Probability measures of fuzzy events. In: *Journal of Mathematical Analysis and Applications* 23 (1968), S. 421–427

Selbständigkeitserklärung

Hiermit erkläre ich, die vorliegende Arbeit selbständig ohne fremde Hilfe verfaßt zu haben und nur die angegebene Literatur und Hilfsmittel verwendet zu haben.