Diplomarbeit

Diagnostik von Telekommunikationsanlagen mit Methoden des maschinellen Lernens

Wilhelm Leibel

Betreuer:

Prof. Dr. Ing. Rudolf Schehrer
Lehrstuhl für Elektronische Systeme und Vermittlungstechnik
Fakultät für Elektrotechnik und Informationstechnik
Universität Dortmund

Prof. Dr. Katharina Morik
Lehrstuhl für Künstliche Inteligenz
Fachbereich Informatik
Universität Dortmund

7. Dezember 2001

Inhaltsverzeichnis

1 Einleitung **9**

2 Einführung in das maschinelle Lernen **13**
- 2.1 Wie kann Lernen definiert werden? . . . 13
- 2.2 Lernen aus Beispielen . . . 14
- 2.3 Statistische Lerntheorie . . . 15
- 2.4 Support Vector Machine . . . 19
  - 2.4.1 Lineare Support Vector Machine . . . 20
  - 2.4.2 Nichtlineare Support Vector Machine . . . 25
- 2.5 Bewertung des Lernerfolgs . . . 28
  - 2.5.1 Fehlerrate . . . 28
  - 2.5.2 Recall, Precision und F_1 . . . 28
  - 2.5.3 Klassifikation einer Testmenge . . . 29
  - 2.5.4 Kreuzvalidierung . . . 29
  - 2.5.5 $\xi\alpha$-Schätzer . . . 29
- 2.6 Fuzzy-Clusteranalyse . . . 30
  - 2.6.1 Clusteranalyse . . . 31
  - 2.6.2 Bewertungsfunktion und Kriterium . . . 33
  - 2.6.3 Fuzzy-c-Means-Algorithmus . . . 35

3 Herkunft und Aufbereitung der Daten **39**
- 3.1 Beschreibung der Daten . . . 39
  - 3.1.1 Grundlagen des ISDN . . . 39
  - 3.1.2 Das LAP-D-Protokoll . . . 40
  - 3.1.3 Ablauf des Informationsaustausches . . . 42
  - 3.1.4 Schicht 3 des D-Kanals . . . 43
- 3.2 Aufzeichnung der Daten . . . 44
  - 3.2.1 Szenario 1: Emulator alleine gegen Anlage . . . 45
  - 3.2.2 Szenario 2: Vermitteln der Rufe durch Anlage . . . 48
  - 3.2.3 Szenario 3: Automatische Annahme der Rufe . . . 51
  - 3.2.4 Szenario 4: Erkennung der Anlage . . . 53
- 3.3 Auswahl und Repräsentation der Daten . . . 54
- 3.4 Vorverarbeitung der Daten . . . 58
  - 3.4.1 Ohne Vorverarbeitung . . . 59
  - 3.4.2 Absolute Häufigkeit in Intervallen . . . 59
  - 3.4.3 Relative Häufigkeit in Intervallen . . . 61
  - 3.4.4 Koordinaten der Cluster . . . 61
  - 3.4.5 Gewicht von Clustern in Intervallen . . . 61

4 Entwicklung der Analyse-Software **63**
4.1 Einlesevorgang 64
4.2 Anwendung der Fuzzy-Clusteranalyse 66
4.3 Anwendung der SVM 71
4.4 Integration in bedienungsfreundliche Oberfläche 72

5 Untersuchungsdurchführung **73**
5.1 Durchführung der Experimente 73
5.1.1 Vorverarbeitung der Ausgangsdaten 73
5.1.2 Umfang der Beispiele 88
5.1.3 Skalierung der Zeitachse 89
5.1.4 Einstellungen der SVM 90
5.2 Auswertung 92

6 Ausblick **95**

Abbildungsverzeichnis

2.1 Modell: Lernen aus Beispielen 15
2.2 Schätzung eines unbekannten Zusammenhangs 17
2.3 Untermengen sortiert nach der VC-Dimension 19
2.4 Prinzip der strukturellen Risikominimierung 19
2.5 Varianten der allgemeinen SVM 20
2.6 Varianten der linearen SVM 20
2.7 Trennende Hyperebene für linear separierbare Daten 22
2.8 Trennende Hyperebene bei nicht linear separierbaren Daten . . . 24
2.9 Verfahren der nichtlinearen SVM 27
2.10 Einfluss der Skalierung auf die Clustereinteilung 32
2.11 Flussdiagramm zum Verfahren der Fuzzy-Clusteranalyse 38

3.1 Rahmenaufbau LAP-D 40
3.2 Ablauf des Informationsaustausches 42
3.3 Ablauf eines Verbindungsaufbaus 44
3.4 Hardware-Komponenten von EasyTrace 44
3.5 Versuchsaufbau Szenario 1 46
3.6 Versuchsaufbau Szenario 2 48
3.7 Screenshot von CPU-Last 50
3.8 Versuchsaufbau Szenario 3 51
3.9 Schichtenarchitektur 54

4.1 Anwendungsfalldiagramm 64
4.2 Graph der logarithmischen Skalierungsfunktion 65
4.3 Fuzzy-Clusteranalyse-Methode 68
4.4 Zweidimensionale Ansicht der Cluster 70
4.5 Ansicht der Verteilung entlang einer Zeitachse 70
4.6 Dreidimensionale Ansicht der Cluster 71
4.7 Screenshot der Benutzungsoberfläche der Software 72

5.1 Vergleich der Leistungsmaße 78
5.2 Clustereinteilung im Szenario 2 81
5.3 Vergleich der Verfahren 87

Tabellenverzeichnis

3.1 Festlegung der SAPI-Werte 41
3.2 Rahmentypen im ISDN 41
3.3 Auszug aus dem Trace im Szenario 1 46
3.4 Rufabstände Szenario 1 47
3.5 Auszug aus dem Trace im Szenario 2 49
3.6 Rufabstände Szenario 2 50
3.7 Versuchsreihen im Szenario 2 51
3.8 Auszug aus dem Trace im Szenario 3 52
3.9 Einteilung der Klassen im Szenario 3 53
3.10 Auszug aus dem Trace im Szenario 4 54
3.11 Abbildung des Rahmentyps auf numerischen Bereich 58

5.1 Parametrierung der Versuche 74
5.2 Testergebnis im Szenario 1 ohne Vorverarbeitung 75
5.3 Testergebnis im Szenario 2 ohne Vorberarbeitung 75
5.4 Testergebnis im Szenario 3 ohne Vorberarbeitung 76
5.5 Testergebnis im Szenario 4 ohne Vorberarbeitung 77
5.6 Fehlerrate Szenario 1 bei absoluter Häufigkeit 79
5.7 Fehlerrate Szenario 2 bei absoluter Häufigkeit 79
5.8 Fehlerrate Szenario 3 bei absoluter Häufigkeit 80
5.9 Fehlerrate Szenario 4 bei absoluter Häufigkeit 82
5.10 Fehlerrate Szenario 1 bei relativer Häufigkeit 82
5.11 Fehlerrate Szenario 2 bei relativer Häufigkeit 83
5.12 Fehlerrate Szenario 3 bei relativer Häufigkeit 83
5.13 Fehlerrate Szenario 4 bei relativer Häufigkeit 83
5.14 Fehlerrate Szenario 1 beim Verfahren Cluster-Koordinaten 84
5.15 Fehlerrate Szenario 2 beim Verfahren Cluster-Koordinaten 84
5.16 Fehlerrate Szenario 3 beim Verfahren Cluster-Koordinaten 85
5.17 Fehlerrate Szenario 4 beim Verfahren Cluster-Koordinaten 85
5.18 Fehlerrate Szenario 1 beim Verfahren Cluster-Gewicht 85
5.19 Fehlerrate Szenario 2 beim Verfahren Cluster-Gewicht 86
5.20 Fehlerrate Szenario 3 beim Verfahren Cluster-Gewicht 86
5.21 Fehlerrate Szenario 4 beim Verfahren Cluster-Gewicht 86
5.22 Fehlerrate bei unterschiedlichen Zeitfenstern 88
5.23 Fehlerrate bei unterschiedlicher Skalierung 89
5.24 Fehlerrate bei unterschiedlichen Kernfunktionen 90
5.25 Fehlerrate bei unterschiedlichen Kapazitäten 91
5.26 Optimale Konfiguration des Analyseverfahrens 92

Kapitel 1

Einleitung

Diese Diplomarbeit beschreibt die Diagnostik von Telekommunikationsanlagen mit Methoden aus der künstlichen Intelligenz. Dabei werden Verfahren aus dem Gebiet des maschinellen Lernens angewandt, um Kommunikationsanlagen zu bewerten und zu klassifizieren.

Die heutigen Telekommunikationsanlagen setzen vollständig auf die Digitaltechnik auf. Die Forderung nach mehr Funktionalität und Flexibilität und nicht zuletzt der Kostenfaktor haben den Einzug von rechnergesteuerten Systemen in modernen Telekommunikationsanlagen unterstützt. Diese Systeme bestehen aus hochintegrierten Mikroelektronik-Bausteinen und entsprechender Kommunikationssoftware. Diese Software orientiert sich an die von Rechnernetzen bekannten Prinzipien und Methoden und enthält einen Protokollstack. Der Einsatz von Software erhöht die Flexibilität der Kommunikationssysteme und bietet die Möglichkeit, neue Funktionalitäten und Dienste durch Updates ohne Änderung der bestehenden Hardware zur Verfügung zu stellen. In heutigen Kommunikationsananlagen befinden sich inzwischen riesige Software-Komponenten, die in der Zukunft noch weiter anwachsen werden. Gleichzeitig wird aber eine extrem hohe Zuverlässigkeit und Robustheit der Dienste erwartet. Deswegen ist es besonders wichtig, fehlerfreies Verhalten in allen Situationen zu gewährleisten. Die Vielzahl der Funktionen in modernen Anlagen macht es unmöglich, alle Funktionalitäten systematisch zu testen. Zudem können Überlastzustände, Besonderheiten im Protokollablauf und Inkompatibilitäten zwischen den Anlagenherstellern mögliche Ursachen für Fehlverhalten und Ausfälle der Anlagen sein.

Formale Methoden sind entwickelt worden, um diese Probleme zu bewältigen und zu überprüfen, ob das reale System den Spezifikationen entspricht. Allerdings decken die Spezifikationen oft nicht die tatsächlichen Anforderungen der Kommunikationssysteme ab. Das Testen ist immer noch sehr schwierig und zeitaufwändig, da einige Fehler nur in bestimmten Situationen auftreten, z.B. bei Überlastzuständen oder bei der Nutzung besonderer Dienste. Dazu kommt, dass einige Fehler nur die Performanz reduzieren und nur in Spitzenlastsituationen zu fehlerhaftem Verhalten führen. Um das Testen zu verbessern, sind daher intelligente Verfahren zur Diagnostik von Kommunikationsanlagen wünschenswert. In der vorliegenden Arbeit werden Verfahren und Erkenntnisse aus der künstlichen Intelligenz eingesetzt, um ISDN-Kommunikationsanlagen zu bewerten und zu klassifizieren. Dabei wird untersucht, ob und in wie weit es anhand von aufgezeichneten Datenströmen der Anlage möglich ist, eine automatische Klassifikation vorzunehmen. Dafür werden Verfahren vorgestellt, die auf aufgezeichnete Testszenarien angewendet werden. Mit Hilfe dieser Szenarien wird eruiert, welches Verfahren am besten geeignet ist und wie es optimal konfiguriert wird.

Grundlage dabei ist die Untersuchung von Matthias Barnutz, der in seiner Diplomarbeit bereits das Verhalten von Telekommunikationsanlagen mit Methoden der künstlichen Intelligenz erforscht hat (siehe [2]). Die Ergebnisse haben gezeigt, dass man auf sehr anschauliche Weise charakteristische Eigenschaften im Verhalten einer Telekommunikationsanlage erkennen kann. Darauf aufbauend werden in der vorliegenden Arbeit die gewonnenen Erkenntnisse seiner Untersuchung weiter vertieft und ergänzt. Dabei wird versucht ein maschinelles Lernverfahren anzuwenden, um eine Klassifikationsregel für die Bewertung einer Anlage zu finden. Hierbei kommt eine Implementierung der Support Vector Machine (SVM) zum Einsatz, die aus der Forschung auf dem Gebiet der statistischen Lerntheorie entstand. Die SVM wurde bereits auf anderen Problembereichen der Mustererkennung erfolgreich eingesetzt. In dieser Arbeit wird die Anwendbarkeit der SVM auf ISDN-Datenströmen von Kommunikationsanlagen untersucht. Die Lernmaschine extrahiert aus einer Menge von Trainingsmustern Strukturen, die ihr die Klassifikation neuer Beispiele erlauben, d.h., dass unbekannte Beispiele in eine Klasse oder Kategorie eingeordnet werden.

Die einzigen Informationen, die von den Kommunikationssystemen zur Verfügung stehen, sind die gemessenen und aufgezeichneten Datenströme auf dem Medium. Aus dieser riesigen Informationsflut müssen Informationen extrahiert werden, die ein Muster für das Verhalten der Kommunikation darstellen. Dabei wird untersucht, ob und wie das Verhalten einer Kommunikation mit wenigen Merkmalen repräsentiert werden kann. Das Kommunikationsverhalten zweier Aufzeichnungen ist unterschiedlich, falls sich die Merkmale unterscheiden, die das Muster definieren. Für eine erfolgreiche Erkennung ist sowohl eine Merkmalsextraktion und als auch eine geeignete Repräsentation der Datenströme notwendig. Um dem Lernalgorithmus die Erkennung der Muster zu ermöglichen bzw. zu erleichtern, werden Verfahren zur Vorverarbeitung der Merkmale entwickelt. Die Vorverarbeitung dieser Merkmale ist eine wesentliche Aufgabe, um auf gegebenen Daten erfolgreich lernen zu können. Die vorgestellten Verfahren werden in einer Software implementiert und auf Testdaten angewendet, um deren Leistungsfähigkeit zu testen.

Die vorliegende Arbeit soll die zentrale Frage beantworten, ob und in wie weit es möglich ist, das Kommunikationsverhalten mit wenigen Informationen zu charakterisieren und darauf Muster zu erkennen, die es ermöglichen, eine Klassifizierung und Bewertung von Kommunikationssystemen vorzunehmen. Die Systeme werden daraufhin überprüft, ob sie den Erwartungen eines Benutzers entsprechen. Es sollen Aussagen möglich werden, die einer Kommunikationsverbindung eine Zuordnung zu Klassen nach verschiedenen Kriterien ermöglichen. Das Verhalten von Anlagen könnte somit z.B. entweder in die Klasse „alles in Ordnung“ oder „etwas seltsam“ eingeordnet werden.

Aus den Ergebnissen der Untersuchung werden Schlussfolgerungen gezogen, die die Bewertung der Anlagen ermöglichen. Die Schlüsse sollen bei der Suche nach Software-Fehlern behilflich sein und können dazu dienen, drohende Probleme aufzudecken. Softwareentwickler erhalten hiermit eine Möglichkeit, signifikante Änderungen des Protokollverhaltens neuer Software-Releases zu erkennen.

Bei der Qualitätssicherung kann automatisch getestet werden, ob das Verhalten innerhalb bestimmter Grenzen liegt. Somit kann die Stabilität der Software bei immer kürzer werdenden Testphasen neuer Software-Updates sichergestellt werden. Auch im laufenden Betrieb könnte die Diagnostik eingesetzt werden, um als Frühwarnsystem bei Überschreiten bestimmter Schranken zu alarmieren, so dass entsprechende Maßnahmen durchgeführt werden können.

Die Arbeit gliedert sich grob in die Kapitel, die die Grundlagen für die Analyse erläutern, und in einen Teil, der die durchgeführten Untersuchungen beschreibt. Das zweite Kapitel enthält eine Einführung in das maschinelle Lernen. Im dritten Kapitel werden die Daten und ihre Herkunft und Repräsentation beschrieben. Daran schließt sich eine Erläuterung der entwickelten Software an, die notwendig ist, um die Untersuchungen durchzuführen. Schließlich wird im Kapitel 5 die Durchführung der Analysen beschrieben.

Diese interdisziplinäre Arbeit entstand in Kooperation mit dem Lehrstuhl für Künstliche Intelligenz und dem Lehrstuhl für Elektronische Systeme und Vermittlungstechnik. Daher fallen die Grundlagenkapitel etwas ausführlicher aus.

Kapitel 2

Einführung in das maschinelle Lernen

Dieses Kapitel stellt eine Einführung in das maschinelle Lernen dar. Zunächst wird *Lernen* allgemein definiert. Danach erfolgt eine Beschreibung des *Lernens aus Beispielen*, wobei wichtige Begriffe erläutert werden. Daran schließt sich eine Ausführung über die Grundlagen der statistischen Lerntheorie und eine detaillierte Beschreibung der Support Vector Machine an. Das nachfolgende Kapitel befasst sich mit der Bewertung des Lernerfolgs einer trainierten SVM. Am Ende des Kapitels wird schließlich die Fuzzy-Clusteranalyse vorgestellt.

2.1 Wie kann Lernen definiert werden?

In der Literatur sind mehrere Definitionen bekannt, die einem System Lernfähigkeit anerkennen. Simon hat eine bekannte Definition angegeben (siehe [29]):

> *Lernen ist jede Veränderung eines Systems, die es ihm erlaubt, eine Aufgabe bei der Wiederholung derselben Aufgabe oder einer Aufgabe derselben Art besser zu lösen.*

Diese Definition ist nicht sehr exakt, da sie auch Erscheinungen einbezieht, die nicht als Lernen bezeichnet werden, und da sie nicht alle Phänomene abdeckt, die dem Lernen zugerechnet werden. So versteht man z.B. unter Lernen die Erkenntnis, dass man mit einem schärferen Messer das Schneiden verbessern kann. Nach Simons Definition wäre Lernen allerdings auch die zufällige Verwendung eines schärferen Messers. Michalski gab sogar ein extremes Gegenbeispiel an, bei dem eine Leistungssenkung als Lernen zu verstehen ist (siehe [23]): Lernen kann man Zwangsarbeitern zusprechen, die einen Weg finden, weniger zu arbeiten und doch gleich beschäftigt auszusehen. Michalski weist auf die Abhängigkeit des Leistungsbegriffes hin.

Scott argumentiert jedoch bei der Definition des Lernens gegen die Leistungsmessung (siehe [27]). Demzufolge lernt z.B. ein Fußgänger in einer unbekannten Stadt, der an einer Bibliothek vorbeikommt, ob und wo es eine Bibliothek gibt. Er lernt unabhängig davon, ob ihn ein Passant nach dem Weg fragt. Lernen ist also nicht davon abhängig, ob man getestet wird. Scotts Definition vom Lernen lautet:

> *Lernen ist ein Prozess, bei dem ein System eine abrufbare Repräsentation von vergangenen Interaktionen mit seiner Umwelt aufbaut.*

Michalski gibt eine vergleichbare Definition an:

> *Lernen ist das Konstruieren oder Verändern von Repräsentationen von Erfahrungen.*

Die Diskussion über die unterschiedlichen Definitionen zeigt, dass Lernen schwierig zu fassen und exakt zu definieren ist. Letztlich bestimmt unser intuitives Verständnis, was wir unter Lernen verstehen (siehe [24]).

2.2 Lernen aus Beispielen

Computer werden oft eingesetzt, um komplexe Probleme zu lösen. Das übliche Vorgehen dabei ist, dass die gewünschte Ausgabe von einer Menge von Eingaben explizit berechnet wird. Dann ist es die Aufgabe des Softwareentwicklers, das Verfahren zur Lösung des Problems in eine Sequenz von Instruktionen zu übersetzen. Bei komplexeren Problemen kann es jedoch vorkommen, dass entweder kein Verfahren bekannt ist oder dass die Berechnung sehr aufwändig ist. Diese Aufgaben können somit nicht in herkömmlicher Weise gelöst werden.

Hier bietet sich die alternative Vorgehensweise an, dass der Computer selber aus einer Menge von Ein- und Ausgaben die Funktion lernt, ähnlich wie Kinder lernen bestimmte Buchstaben zu erkennen. Diese Lernmethode, bei der Ein- und Ausgabe-Paare gegeben sind, bezeichnet man als *überwachtes Lernen* (supervised learning). Die Ein- und Ausgabe-Paare werden als *Beispiele der Trainingsmenge* bezeichnet (siehe [5]). Die Beispiele spiegeln einen funktionalen Zusammenhang wider, der als Zielfunktion bezeichnet wird. Dennoch können in einigen Fällen die Beispiele durch Rauschen verzerrt sein. Die Schätzung dieser *Zielfunktion* ist die Ausgabe des Lernalgorithmus und wird als Lösung des Lernproblems bezeichnet. Bei der Klassifikation spricht man oft von der *Entscheidungsfunktion*. Die Lösung wird aus einer Menge von Funktionen ausgewählt. Die Funktionen der Menge werden dabei als Hypothesen bezeichnet. Entscheidungsbäume, konstruiert aus binären Bäumen mit Entscheidungen an den inneren Knoten und Werten an den Blättern, sind beispielsweise Hypothesen. Daher ist die Wahl der Menge der Hypothesen oder des Hypothesenraums ein wichtiges Merkmal der Lernstrategie. Üblicherweise wird zunächst versucht eine Menge oder Klasse von Funktionen zu wählen, die den Eingaberaum auf die Ausgabemenge abbildet. Anschließend wählt der Lernalgorithmus mit Hilfe der Trainingsdaten eine Hypothese aus dem Hypothesenraum aus (siehe [5]).

Das allgemeine Modell des überwachten Lernens besteht dabei aus drei Komponenten (siehe Abbildung 2.1):

1. einem Generator, der zufällige Vektoren $\vec{x} \in \mathbb{R}^N$ erzeugt, die einer festen, aber unbekannten Wahrscheinlichkeitsverteilung unterliegen,

2. einem Supervisor, der einen Ausgabewert y zu jedem Eingabevektor $\vec{x}$ mit einer Wahrscheinlichkeit von $P(y/\vec{x})$ zurückgibt und

3. einer Lernmaschine, die in der Lage ist, aus einer Menge von Funktionen $f_\alpha(\vec{x}), \alpha \in \Lambda$ eine Funktion auszuwählen, wobei Λ eine Menge von Parametern ist.

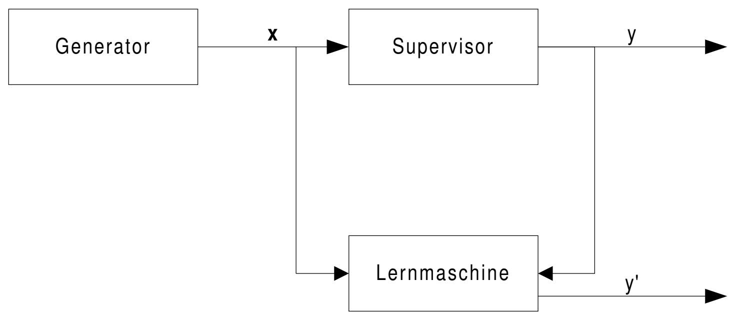


Abbildung 2.1: Modell: Lernen aus Beispielen

Das Problem des Lernens bedeutet nun, aus der Menge $\{f_\alpha : \alpha \in \Lambda\}, f_\alpha : \mathbb{R}^N \to Y$ diejenige Funktion auszuwählen, die am besten der Ausgabe des Supervisors entspricht (siehe [30]).

Bei dem Lernverfahren „Buchstaben erkennen" handelt es sich um ein typisches Klassifikationsproblem. Haben die Ausgabewerte nur zwei Klassen, so bezeichnet man es als *binäres Klassifikationsproblem*. Enthält das Problem mehr als zwei Klassen aber endlich viele, so spricht man von *Multi-Klassen-Klassifikation*, das auf binäre Klassifikation zurückgeführt werden kann. Eine Klasse ist dabei eine Untermenge der Trainingsdatensätze und wird durch eine symbolische Repräsentation (Bezeichner, Zahl, ...) gekennzeichnet. Probleme mit reellwertigen Ausgaben sind als *Regression* bekannt geworden.

Im Fall des *unüberwachten Lernens* (unsupervised learning) werden den Beispielen keine Ausgabewerte oder Klassen zugeordnet. Das Lernverfahren versucht dann nur mit den Eingabedaten Muster oder Klassen zu erkennen. Dies führt somit zu einer abstrahierten Zusammenfassung der Muster, die allen Objekten innerhalb der Daten zugrunde liegen. Das Fuzzy-Clustering ist ein Beispiel für ein unüberwachtes Lernverfahren und wird im Kapitel 2.6 beschrieben.

2.3 Statistische Lerntheorie

In diesem Abschnitt erfolgt eine Einführung in die statistische Lerntheorie. Sie beinhaltet die mathematischen Grundlagen des Lernens aus Beispielen. Als Basis für das Lernen stehen die Trainingsbeispiele (Beobachtungen) zur Verfügung, die einer unbekannten Wahrscheinlichkeitsverteilung $P(\vec{x}, y)$ unterliegen:

$$(\vec{x_1}, y_1), ..., (\vec{x_l}, y_l) \in \mathbb{R}^N \times Y. \tag{2.1}$$

Um eine möglichst gute Annäherung an die Antwort des Supervisors zu erreichen, muss die Abweichung messbar gemacht werden. $L(y, f_\alpha(\vec{x}))$ bezeichnet den Verlust zwischen den Antworten y des Supervisors und den Ausgaben $f_\alpha(\vec{x})$ der Lernmaschine. Das Risiko berechnet sich dann aus

$$R(\alpha) = \int L(y, f_\alpha(\vec{x}))\, dP(\vec{x}, y). \tag{2.2}$$

Das Ziel ist es, die Funktion $f_{\alpha_0}(\vec{x})$ zu finden, die die Risiko-Funktion $R(\alpha)$ minimiert, wobei die Verteilungsfunktion $P(\vec{x}, y)$ unbekannt ist. Die Trainingsbeispiele sind die einzigen Informationen, die die Lernmaschine bekommt (siehe [30]).

Dieses mathematische Modell ist sehr allgemein gehalten und wird für die drei Hauptprobleme spezialisiert: Problem der Mustererkennung, Regressionsabschätzung und Dichteabschätzung. In dieser Arbeit ist nur das Problem der Mustererkennung relevant, daher wird dieses Problem näher beschrieben.

Die Ausgabe y des Supervisors kann bei der binären Klassifikation nur zwei Werte annehmen, -1 oder 1. Die Trainingsdaten sind dann ein Tupel aus einem Muster $\vec{x_i}$ und einer Markierung $y \in \{-1, 1\}$,

$$(\vec{x}_1, y_1), ..., (\vec{x}_l, y_l) \in \mathbb{R}^N \times \{-1, 1\}\,. \tag{2.3}$$

Die Entscheidungsfunktion der Lernmaschine kann daher nur zwei Werte annehmen:

$$f_\alpha : \mathbb{R}^N \to \{-1, 1\}\,. \tag{2.4}$$

Als Verlustfunktion[1] ergibt sich dann:

$$L(y, f_\alpha(\vec{x})) = \begin{cases} 0 & \text{für} \quad y = f_\alpha(\vec{x}) \\ 1 & \text{für} \quad y \neq f_\alpha(\vec{x}) \end{cases} \tag{2.5}$$

Die Verlustfunktion wird auch als *Loss* bezeichnet. Bei einer korrekten Vorhersage stimmt die Antwort der Lernmaschine mit der des Supervisors überein und der Verlust ist 0. Die Differenz zwischen $f_\alpha(\vec{x})$ und y ist in diesem Fall ebenfalls 0. Von einem *Klassifikationsfehler* spricht man, wenn die Antworten unterschiedlich sind. Dann ist der Verlust eins und der Betrag der Differenz zwischen $f_\alpha(\vec{x})$ und y ist zwei. Beide Fälle der Gleichung 2.5 lassen sich daher zu folgender Verlustfunktion zusammenfassen:

$$L(y, f_\alpha(\vec{x})) = \frac{1}{2}|f_\alpha(\vec{x}) - y| \tag{2.6}$$

Wird diese Verlustfunktion in Gleichung 2.2 eingesetzt, so erhält man für das Risiko:

$$R(\alpha) = \int \frac{1}{2}|f_\alpha(\vec{x}) - y| dP(\vec{x}, y). \tag{2.7}$$

Das Problem ist es nun, eine Funktion f_α (siehe Gleichung 2.4) zu finden, die die Wahrscheinlichkeit für einen Klassifikationsfehler möglichst gering hält, wobei die Verteilung $P(\vec{x}, y)$ unbekannt ist und nur die Trainingsdaten zur Verfügung stehen. Gesucht ist also eine Funktion f_α, die das Risiko $R(\alpha)$ minimiert.

Die Gleichung 2.7 ist zwar wohldefiniert, aber meist nicht berechenbar, da $P(\vec{x}, y)$ nicht bekannt ist. Um das Risiko mit einer unbekannten Verteilungsfunktion zu minimieren, wird ein induktives Prinzip angewandt: Aus den endlich vielen Trainingsdaten wird auf die Allgemeinheit geschlossen (Generalisierung). Das Risiko wird durch das sogenannte *empirische Risiko* ersetzt, das nur aus den Beobachtungen ausgewertet wird (siehe [4]):

$$R_{emp}(\alpha) = \frac{1}{2l} \sum_{i=1}^{l} |y_i - f_\alpha(\vec{x}_i)|\,. \tag{2.8}$$

[1] Bei der Regression könnte es sich bei $L(y, f_\alpha(\vec{x}))$ um den quadratischen Fehler $(y - f_\alpha(\vec{x}))^2$ handeln.

Dieses Prinzip wird als *Empirische Risikominimierung* bezeichnet.

Es genügt nun nicht, das empirische Risiko zu minimieren, also einen niedrigen Trainingsfehler zu erreichen. Genauso wichtig ist die Beachtung der *Komplexität der Lernmaschine*, um eine möglichst gute Generalisierung auf unbekannte Testdaten zu erreichen.

Ein einfaches Beispiel soll dies verdeutlichen: Gegeben ist eine Menge von Trainingsbeispielen $\{\vec{x}_1, \ldots, \vec{x}_l\}$ und eine beliebige Klassifikationsfunktion $f : \mathbb{R}^N \rightarrow \{\pm 1\}$. Die Lernmaschine wird nun mit einer Menge von neuen Beispielen $\{\vec{x}_1^*, \ldots, \vec{x}_m^*\}$ mit der Eigenschaft $\{\vec{x}_1, \ldots, \vec{x}_l\} \cap \{\vec{x}_1^*, \ldots, \vec{x}_m^*\} = \emptyset$ getestet. Die Funktionswerte der beiden disjunkten Mengen sind völlig unabhängig voneinander. Daher lässt sich offenbar eine zweite Funktion $f^{'}$ angeben, die auf der Trainingsmenge dieselben Funktionswerte liefert wie f, aber auf der Testmenge trotzdem unterschiedliche Vorhersagen macht. Lernen ist hier also nicht möglich, da allein aufgrund der Trainingsdaten nicht entschieden werden kann, welche der beiden Funktionen besser gelernt hat. Dies liegt daran, dass die Lernmaschine die Klassifikationsfunktion aus der Menge aller Funktionen auswählen darf. Zum erfolgreichen Lernen muss also die Menge der Funktionen, die eine Lernmaschine realisieren kann, eingeschränkt werden.

Für eine Lernmaschine ist es immer möglich, das empirische Risiko auf null zu bringen, also keine Fehler auf der Trainingsmenge zu machen. Eine solche Funktion stammt jedoch meist aus einer Funktionenklasse hoher Komplexität. So können z.B. l Punkte durch ein Polynom vom Grad $l-1$ beschrieben werden. Der Trainingsfehler ist zwar null, aber man kann nicht unbedingt davon ausgehen, dass die Lernmaschine gut gelernt hat. Dies wird an Hand eines Beispiels erläutert (siehe [26]).

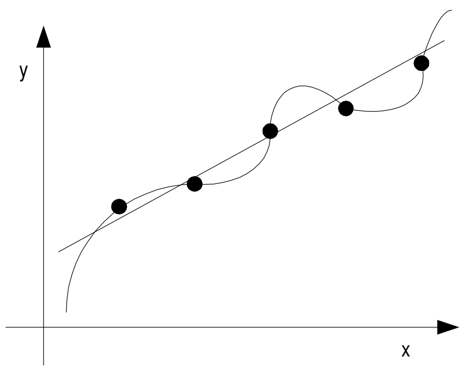


Abbildung 2.2: Schätzung eines unbekannten Zusammenhangs von einer Geraden und von einem Polynom vierten Grades

Nehmen wir an, die Abbildung 2.2 stellt empirische Messdaten dar, die durch Rauschen verzerrt sind. Dann wird man der Geraden mehr Vertrauen schenken als dem Polynom vierten Grades, obwohl es einen Trainingsfehler von null erreicht. Gibt die Lernmaschine in diesem Fall die Gerade als Entscheidungsfunktion zurück, so hat sie besser gelernt als wenn sie das Polynom zurückgibt. Es genügt also nicht, die Punkte der Trainingsmenge möglichst gut zu „erklären“. Die Komplexität der Funktionen-Menge muss gering gehalten werden. Zum erfolgreichen Lernen muss also die Menge der Funktionen, aus der eine Lernmaschine eine Schätzung auswählt, eingeschränkt werden. Schafft man es, mit einer Funktionenmenge mit kleiner Komplexität auf den Trainingsdaten ein geringes empirisches Risiko zu erreichen, so hat man mit hoher Wahrscheinlichkeit den tatsächlichen funktionalen Zusammenhang gefunden (siehe [26]).

Dieser intuitive Komplexitätsbegriff wird nun durch die VC (Vapnik-Chervonenkis)-Dimension von einer Menge von Funktionen formalisiert. Die VC-Dimension bildet den Kern der statistischen Lerntheorie. Bei einer gegebenen Menge von l Punkten gibt es bei der binären Klassifikation 2^l Möglichkeiten der Markierung dieser Punkte. Die VC-Dimension h einer Menge von Funktionen gibt die maximale Anzahl von Punkten an, die in allen Möglichkeiten in zwei Klassen von den Funktionen der Menge getrennt werden können. Werden Hyperebenen in $\mathbb{R}^N$ als Funktionen betrachtet, so können maximal $N+1$ Punkte in allen Variationsmöglichkeiten aufgeteilt werden. Die VC-Dimension ist dementsprechend $N+1$. Wählt man einen Punkt als Ausgangspunkt, dann müssen die Ortsvektoren zu den restlichen Punkten linear unabhängig sein. Der Beweis ist in [4] nachzulesen.

Vapnik formulierte mit Hilfe der VC-Dimension eine obere Schranke für das Risiko, die bei gegebenen η mit $0 \leq \eta \leq 1$ mit einer Wahrscheinlichkeit von $1-\eta$ eingehalten wird (siehe [30]):

$$R(\alpha) \leq R_{emp}(\alpha) + \phi\left(\frac{h}{l}, \frac{\log(\eta)}{l}\right). \tag{2.9}$$

Der Parameter h ist die VC-Dimension der zugrundeliegenden Menge von Funktionen. Der Ausdruck ϕ wird dabei als VC-Sicherheit (VC confidence) bezeichnet und ist wie folgt definiert:

$$\phi\left(\frac{h}{l}, \frac{\log(\eta)}{l}\right) = \sqrt{\frac{h\left(\log\frac{2l}{h}+1\right) - \log\frac{\eta}{4}}{l}}. \tag{2.10}$$

Für eine endliche Anzahl von Trainingsdaten kann das Risiko durch die zwei Größen $R_{emp}(\alpha)$ und die VC-Dimension h kontrolliert werden. Das empirische Risiko hängt von der gewählten Funktion ab und kann durch die Wahl von α variiert werden. Die VC-Dimension h hängt dagegen von der Menge der Funktionen ab, aus der die Lernmaschine auswählen kann. Für die Lösung des Problems wird das Prinzip der *strukturellen Risikominimierung* angewandt (siehe [30]). Hierbei werden Untermengen $S_n := \{f_\alpha : \alpha \in \Lambda_n\}$ aus $\{f_\alpha : \alpha \in \Lambda\}$ gebildet, so dass gilt:

$$S_1 \subset S_2 \subset \ldots S_n \subset \ldots, \tag{2.11}$$

wobei für die zugehörigen VC-Dimensionen h_n

$$h_1 \leq h_2 \leq \ldots \leq h_n \leq \ldots < \infty \tag{2.12}$$

gilt. Abbildung 2.3 stellt die Untermengen beispielhaft dar.

Für eine gegebene Trainingsmenge $(\vec{x}_1, y_1), \ldots, (\vec{x}_l, y_l)$ sucht nun das Prinzip der strukturellen Risikominimierung diejenige Funktion $f_{\alpha_l^n}$ aus der Untermenge S_n, für die die obere Schranke des Risikos minimiert wird. Bei steigender

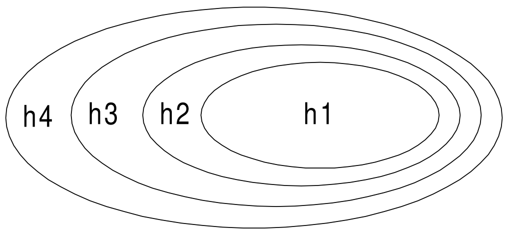


Abbildung 2.3: Untermengen sortiert nach der VC-Dimension

VC-Dimension sinkt zwar der Trainingsfehler, jedoch wächst der Sicherheitsterm ϕ an. Das Prinzip der strukturellen Risikominimierung wägt zwischen Trainingsfehler und VC-Sicherheit ab (siehe Abbildung 2.4).

Die Lösung kann berechnet werden, indem für jede Untermenge eine Lernmaschine das empirische Risiko minimiert und zur entsprechenden VC-Sicherheit addiert. Anschließend kann das Minimum aus den Summen bestimmt werden.

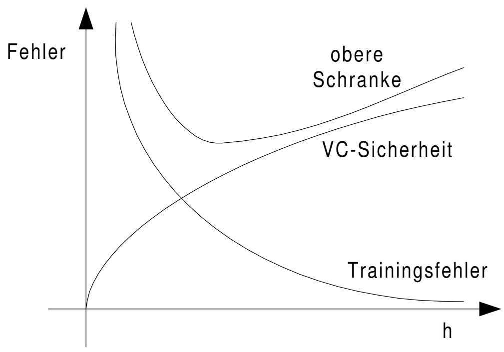


Abbildung 2.4: Prinzip der strukturellen Risikominimierung. Grafische Darstellung des Zusammenhangs aus Gleichung 2.9.

2.4 Support Vector Machine

Die Support Vector Machine (SVM) basiert auf den Erkenntnissen der statistischen Lerntheorie. Zunächst wird die lineare SVM vorgestellt, die auf linear und nicht linear separierbare Daten angewendet werden kann. Dabei wird der zuerst der Fall untersucht, bei dem die Trainingsdaten linear separierbar sind. Da jedoch in der Praxis die Daten oft nicht linear zu trennen sind, kann diese SVM nicht angewendet werden. Die Erkenntnisse aus dem linearen Fall werden auf den allgemeinen Fall mit nicht linear separierbaren Daten übertragen. Bei unbekannten Daten ist der Zusammenhang meist nicht vorher bekannt, so dass man bei der Anwendung von nicht linear trennbaren Daten ausgeht. Die

Beschreibung der SVM auf linear separierbaren Daten dient hier als Herleitung für den allgemeinen Fall mit nicht linear trennbaren Daten.

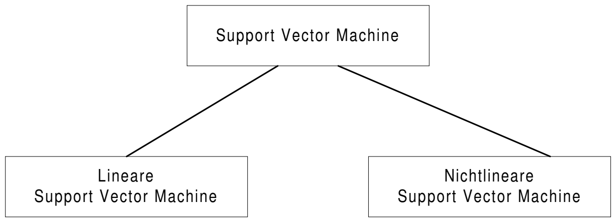


Abbildung 2.5: Varianten der allgemeinen SVM

Da der Zusammenhang von empirischen Daten oft nicht mit einer linearen Entscheidungsfunktion wiedergegeben werden kann, wird die lineare SVM zu einer SVM mit nichtlinearer Klassifikationsfunktion erweitert. Die nichtlineare SVM basiert nach einer Transformation in einen höherdimensionalen Raum auf einer linearen SVM. In diesem Raum können die Daten wieder linear oder nicht linear separierbar sein. In Abbildung 2.5 werden die Varianten der SVM dargestellt. Die Ausführungen dieses Kapitels beziehen sich vorwiegend auf Literatur von Vapnik [30], Burges [4] und Schölkopf [26].

2.4.1 Lineare Support Vector Machine

Die lineare Support Vector Machine verwendet eine lineare Entscheidungsfunktion. Zunächst wird der Fall für separierbare Daten betrachtet, der dann verallgemeinert wird für nicht separierbare Daten. Abbildung 2.6 veranschaulicht die Varianten der linearen SVM.

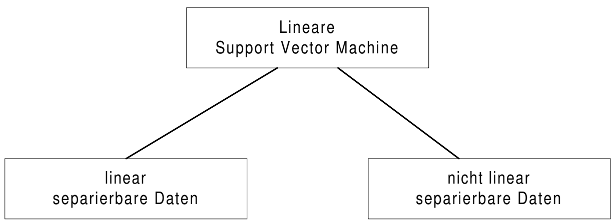


Abbildung 2.6: Varianten der linearen SVM

Linear separierbare Daten

Es ist eine Menge von Punkten gegeben, die jeweils einer von zwei disjunkten Mengen angehören. Dies entspricht wieder der binären Klassifikation. Die Aufgabe der Lernmaschine ist es, diejenige lineare Entscheidungsfunktion zu finden, die die Daten richtig separiert. Gesucht ist demnach eine Ebenengleichung $g = \vec{w}\vec{x} + b$, wobei $\vec{w}$ der Normalenvektor der Hyperebene ist. Für die Punkte $\vec{x}$, die auf der Hyperebene liegen, ist die Gleichung $\vec{w}\vec{x} + b = 0$ erfüllt. Der Abstand der Ebene zum Ursprung beträgt $|b|/||w||$. Zwischen den positiven und den negativen Beispielen existiert ein Grenzbereich, in dem sich keine Punkte befinden. Die Hyperebene trennt den Raum in zwei Halbräume, die die Klassen der Trainingsdaten bilden. Daraus ergibt sich die zu suchende Entscheidungsfunktion

$$f_{\vec{w},b}(\vec{x}) = \operatorname{sgn}(\vec{w}\cdot\vec{x}+b). \tag{2.13}$$

Die Hyperebene separiert die Datenmenge in zwei Klassen, daher wird gefordert, dass folgende Bedingungen gelten:

$$\vec{w}\cdot\vec{x}_i+b \geq +1 \quad \forall i \quad \text{mit} \quad y_i=+1 \tag{2.14}$$

$$\vec{w}\cdot\vec{x}_i+b \leq -1 \quad \forall i \quad \text{mit} \quad y_i=-1. \tag{2.15}$$

Die Bedingungen 2.14 und 2.15 können durch Multiplikation mit y_i zu einer Bedingung zusammengefasst werden:

$$y_i(\vec{w}\cdot\vec{x}_i+b)-1 \geq 0 \quad \forall i. \tag{2.16}$$

Man sucht nun diejenige Hyperebene, die einen möglichst großen Abstand zu den Punkten der beiden Mengen hat. d_+ (d_-) sei der minimale Abstand zu den positiven (negativen) Beispielen, dann wird der Grenzbereich (*margin*) der Hyperebene wie folgt definiert:

$$margin = d_+ + d_i \tag{2.17}$$

Die Punkte, für die die Bedingung 2.14 zu einer Gleichung wird, liegen auf der Hyperebene H1 : $\vec{w}\vec{x}+b=1$ mit dem Normalenvektor $\vec{w}$ und einem Abstand $|1-b|/||\vec{w}||$ vom Ursprung. Genauso gibt es auf der gegenüberliegenden Seite eine Hyperebene H2 : $\vec{w}\vec{x}+b=-1$ mit einem Abstand $|-1-b|/||\vec{w}||$. Daraus ergibt sich für den Grenzbereich $margin = 2/||\vec{w}||$ und für $d_+ = d_- = 1/||\vec{w}||$. In Abbildung 2.7 werden diese Zusammenhänge grafisch veranschaulicht. Die beiden Hyperebenen H1 und H2 haben denselben Normalenvektor und sind deshalb auch parallel zueinander. Zwischen diesen beiden Hyperebenen befinden sich in diesem Fall keine Punkte. Um die optimal trennende Hyperebene zu bestimmen, muss der Grenzbereich maximiert werden durch Minimierung von $||\vec{w}||$ unter Einhaltung der Bedingungen 2.16 (vgl. [4]).

Die Lage der Hyperebenen H1 und H2 werden nur durch die Vektoren festgelegt, die genau auf einer der beiden Hyperebenen liegen (siehe Abbildung 2.7). Diese Punkte werden als Support Vektoren bezeichnet. Eine Änderung der Lage der Support Vektoren könnte auch die trennende Hyperebene verändern und somit die Lösung des Problems. Die anderen Punkte tragen dagegen nicht zu der Lösung des Problems bei.

Um die Lösung zu bestimmen, muss das folgende konvexe Optimierungsproblem gelöst werden:

$$\text{minimiere} \quad \tau(\vec{w}) = \frac{1}{2}||\vec{w}||^2 \tag{2.18}$$

unter den Nebenbedingungen

$$y_i(\vec{w}\cdot\vec{x}_i+b) \geq 1, \quad i=1,\ldots,l. \tag{2.19}$$

Das Extremwertproblem lässt sich mit Hilfe des Lagrange-Ansatzes lösen, mit dem zwei Vorteile erreicht werden. Zum einen sind die Nebenbedingungen einfach zu handhaben und zum anderen erscheinen die Daten der Trainingsmenge

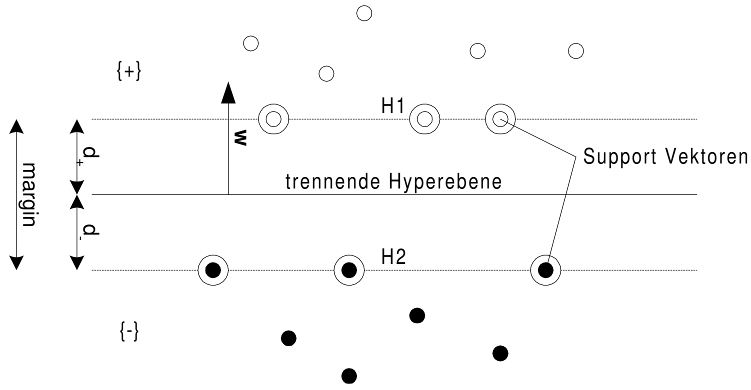


Abbildung 2.7: Trennende Hyperebene für linear separierbare Daten

nur in Skalarprodukten. Der zweite Vorteil ist eine grundlegende Eigenschaft, die es erlaubt, den linearen Fall auf den nichtlinearen zu übertragen (siehe [4]). Es werden nun positive Lagrange-Multiplikatoren $\alpha_i > 0,\ i = 1, \ldots, l$ für jede Nebenbedingung eingeführt. Man erhält dann die Gleichung

$$L_P = \frac{1}{2}||w||^2 - \sum_{i=1}^{l} \alpha_i y_i (\vec{x}_i \cdot \vec{w} + b) + \sum_{i=1}^{l} \alpha_i \tag{2.20}$$

L_P stellt das primale Optimierungsproblem dar und muss nun bezüglich $\vec{w}$ und b minimiert werden, wobei gleichzeitig die Ableitung nach α_i null sein muss. Man sucht also den sogenannten Sattelpunkt. Dies ist ein konvexes quadratisches Optimierungsproblem. Äquivalent dazu ist das duale Problem: maximiere L_P unter den Nebenbedingungen, dass der Gradient von L_P bezüglich $\vec{w}$ und b null ist und $\alpha_i \geq 0$. Dies führt zu den Gleichungen

$$\frac{\partial}{\partial b} L_P = \sum_{i=1}^{l} \alpha_i y_i = 0, \quad \text{und} \tag{2.21}$$

$$\frac{\partial}{\partial w} L_P = \vec{w} - \sum_{i=1}^{l} \alpha_i y_i \vec{x}_i = 0 \tag{2.22}$$

$$\Rightarrow \vec{w} = \sum_{i=1}^{l} \alpha_i y_i \vec{x}_i. \tag{2.23}$$

Man sieht, dass $\vec{w}$ eine Linearkombination aus den Eingabedaten ist. Werden diese Gleichungen in 2.20 eingesetzt, so erhält man die duale Form des Problems:

$$L_D = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j. \tag{2.24}$$

Die Aufgabe der linearen Support Vector Machine ist es nun, L_D zu maximieren unter den Nebenbedingungen

$$\alpha_i \quad \geq \quad 0, \quad i = 1, \ldots, l, \tag{2.25}$$

$$\sum_{i=1}^{l} \alpha_i y_i = 0. \tag{2.26}$$

Für die Support Vektoren gilt dabei $\alpha_i > 0$ und für alle anderen Punkte $\alpha_i = 0$ (siehe [4]).

Karush-Kuhn-Tucker-Bedingungen Bei den beschränkten Optimierungsproblemen spielen die Karush-Kuhn-Tucker- (KKT-)Bedingungen eine zentrale Rolle. Zu dem primalen Optimierungsproblem in 2.20 sind folgende KKT-Bedingungen äquivalent (siehe [11]):

$$\frac{\partial}{\partial w_\nu} L_P = w_\nu - \sum_i \alpha_i y_i x_{i\nu} = 0 \quad \nu = 1, \ldots, d \tag{2.27}$$

$$\frac{\partial}{\partial b} L_P = -\sum_i \alpha_i y_i = 0 \tag{2.28}$$

$$y_i(\vec{x}_i \cdot \vec{w} + b) - 1 \geq 0 \quad i = 1, \ldots, l \tag{2.29}$$

$$\alpha_i \geq 0 \quad \forall i \tag{2.30}$$

$$\alpha_i(y_i(\vec{w} \cdot \vec{x}_i + b) - 1) = 0 \quad \forall i \tag{2.31}$$

Diese KKT-Bedingungen sind sowohl notwendig als auch hinreichend dafür, dass $\vec{w}$, b und $\vec{\alpha}$ eine Lösung sind, da das Problem konvex ist (siehe [11]). Das bedeutet, dass das Lösen der KKT-Bedingungen äquivalent ist zur Bestimmung der Lösung des primalen Problems. Während $\vec{w}$ in diesem Fall explizit bestimmt wird, muss der Abstand b noch nachträglich berechnet werden, da er nur implizit dargestellt ist. b kann jedoch einfach berechnet werden, indem ein i mit der Bedingung $\alpha_i \neq 0$ gewählt und in die KKT-Bedingung 2.31 eingesetzt wird.

Nicht linear separierbare Daten

In der Praxis sind die Daten oft nicht linear separierbar und somit nicht durch eine Hyperebene zu trennen. Damit die lineare Support Vector Machine trotzdem auf diese Daten eingesetzt werden kann, wurden von Cortes und Vapnik sog. Schlupfvariablen eingeführt, mit der Eigenschaft

$$\xi_i \geq 0, \quad i = 1, \ldots, l, \tag{2.32}$$

die die Bedingungen 2.16 abschwächen ((siehe [6]):

$$y_i(\vec{w} \cdot \vec{x}_i + b) \geq 1 - \xi_i, \quad i = 1, \ldots, l. \tag{2.33}$$

Dies erlaubt Fehlklassifikationen, die jedoch gering bleiben sollen (siehe Abbildung 2.8). Bei einem Fehler wird, nach Bedingung 2.33, die entsprechende Schlupfvariable $\xi_i > 1$. Daher ist die obere Schranke für die Anzahl von Trainingsfehlern $\sum_i \xi_i$. Das Optimierungsproblem 2.18 wird nun um diese Schlupfvariablen erweitert:

$$\text{minimiere} \quad \tau(\vec{w}, \xi) = \frac{1}{2}||\vec{w}||^2 + C \sum_{i=1}^{l} \xi_i \tag{2.34}$$

unter den Nebenbedingungen 2.32 und 2.33. Zur ursprünglichen Zielfunktion werden also „Extrakosten“ hinzuaddiert, wobei der Parameter C steuert, wie

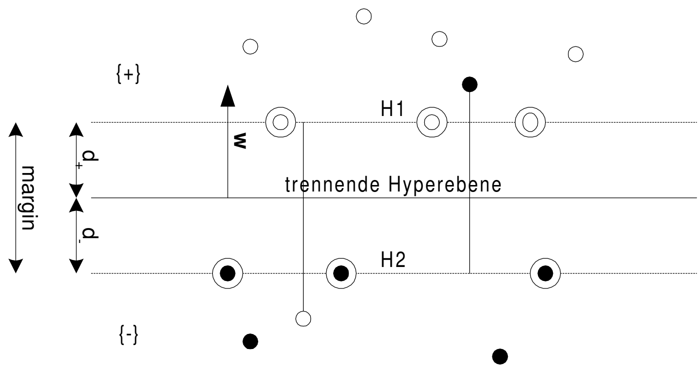


Abbildung 2.8: Trennende Hyperebene bei nicht linear separierbaren Daten

hoch Fehlklassifikationen bestraft werden. Der Parameter C ist eine praktische Implementierung der strukturellen Risikominimierung (siehe [26]). Es liegt hier wieder ein konvexes Optimierungsproblem vor. Über den Lagrange-Ansatz erhält man das primale Optimierungsproblem:

$$L_P = \frac{1}{2}||w||^2 + C\sum_i \xi_i - \sum_i \alpha_i(y_i(\vec{w}\cdot\vec{x}_i + b) - 1 + \xi_i) - \sum_i \mu_i\xi_i. \tag{2.35}$$

Hierbei wurden μ_i als Lagrange-Multiplikatoren eingeführt, um sicherzustellen, dass die Variablen ξ_i positiv sind. Als KKT-Bedingungen ergeben sich dann:

$$\frac{\partial L_P}{\partial w_\nu} = w_\nu - \sum_{i=1}^{l} \alpha_i y_i x_i \nu \quad = \quad 0 \tag{2.36}$$

$$\frac{\partial L_P}{\partial b} = -\sum_{i=1}^{l} \alpha_i y_i \quad = \quad 0 \tag{2.37}$$

$$\frac{\partial L_P}{\partial \xi_i} = C - \alpha_i - \mu_i \quad = \quad 0 \tag{2.38}$$

$$y_i(\vec{w}\cdot\vec{x}_i + b) - 1 + \xi_i \quad \geq \quad 0 \tag{2.39}$$

$$\xi_i \quad \geq \quad 0 \tag{2.40}$$

$$\alpha_i \quad \geq \quad 0 \tag{2.41}$$

$$\mu_i \quad \geq \quad 0 \tag{2.42}$$

$$\alpha_i(y_i(\vec{w}\cdot\vec{x}_i + b) - 1 + \xi_i) \quad = \quad 0 \tag{2.43}$$

$$\mu_i\xi_i \quad = \quad 0. \tag{2.44}$$

Das dazu duale Optimierungsproblem lautet:

$$\text{maximiere} \quad L_D = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i\alpha_j y_i y_j \vec{x}_i \cdot \vec{x}_j \tag{2.45}$$

unter den Nebenbedingungen

$$\begin{aligned} 0 \leq \alpha_i &\leq C, \quad i = 1, \ldots, l, && (2.46) \\ \sum_i \alpha_i y_i &= 0. && (2.47) \end{aligned}$$

Die Lösung ist wieder gegeben durch

$$\vec{w} = \sum_{i=1}^{N_S} \alpha_i y_i \vec{x}_i, \tag{2.48}$$

wobei N_S die Anzahl der Support Vektoren ist. Der einzige Unterschied zum Fall mit linear trennbaren Daten ist, dass die α_i durch obere Schranke C begrenzt werden.

Das Lösen dieses Optimierungsproblems bestimmt also die noch beste Hyperebene bei Unterstellung eines linearen Zusammenhangs bezüglich einer ungewichteten Summe der Fehler ξ_i. Ein Kritikpunkt dieser Methode ist, dass eine Abweichung reicht, um die Lage der Hyperebene zu verfälschen.

Nachdem die Lösung des Problems gefunden wurde, können unbekannte Testdaten klassifiziert werden. Es muss lediglich bestimmt werden, auf welcher Seite der gelernten Hyperebene der Testvektor $\vec{x}$ liegt. Dafür substituiert man den Ausdruck 2.23 in der allgemeinen Entscheidungsfunktion 2.13 und erhält

$$f(\vec{x}) = \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i y_i (\vec{x} \cdot \vec{x}_i) + b\right). \tag{2.49}$$

Die lineare Support Vector Machine hat den Nachteil, dass keine nichtlineare Entscheidungsfunktion eingesetzt werden kann. Hat man es mit nichtlinear trennbaren Daten zu tun, die nicht nur durch Rauschen zustande gekommen sind, so stößt man auf die Grenzen der linearen SVM.

2.4.2 Nichtlineare Support Vector Machine

In der Praxis treten häufig Klassifikationsprobleme auf, die nichtlineare Zusammenhänge aufweisen. Die nichtlineare Support Vector Machine ist in der Lage, nichtlinear separierbare Daten zu trennen. Boser, Guyon und Vapnik [3] zeigen, dass dafür ein bereits von Aizerman [1] entdeckter Trick angewendet werden kann. Die Grundidee ist dabei, die Trainingsdaten über eine nichtlineare Abbildung

$$\begin{aligned} \Phi : \mathbb{R}^N &\rightarrow F && (2.50) \\ \vec{x} &\mapsto \Phi(\vec{x}) && (2.51) \end{aligned}$$

in einen hochdimensionalen Raum F zu transformieren, in dem die transformierten Daten dann linear separierbar sind. Das Lernproblem wird jetzt für die Daten

$$(\Phi(\vec{x}_1), y_1), \ldots, (\Phi(\vec{x}_l), y_l) \in F \times Y \tag{2.52}$$

betrachtet. In dem Optimierungsproblem 2.45 - 2.47 fällt auf, dass die Trainingsdaten nur in Form von Skalarprodukten $\vec{x}_i \cdot \vec{x}_j$ auftreten. Existiert nun eine sogenannte *Kernfunktion* K, so dass gilt

$$K(\vec{x}_i, \vec{x}_j) = \Phi(\vec{x}_i) \cdot \Phi(\vec{x}_j) \quad \text{für} \quad i, j \in 1, \ldots, l, \tag{2.53}$$

dann wird in dem Optimierungsproblem nur die Funktion K benötigt und man befreit sich von der expliziten Bestimmung der Funktion Φ. Somit erhält man eine nichtlineare Entscheidungsfunktion, indem die allgemeine Funktion 2.13 um die Kernfunktion K erweitert wird zu

$$f(\vec{x}) = \operatorname{sgn}\left(\sum_{i=1}^{l} \alpha_i y_i K(\vec{x}, \vec{x}_i) + b\right). \tag{2.54}$$

Dazu wird ein kleines Beispiel im zweidimensionalen Raum ($N = 2$) mit einem Polynom zweiten Grades ($d = 2$) vorgestellt (siehe [26]). Für die Abbildung

$$C_2 : (x_1, x_2) \mapsto (x_1^2, x_2^2, x_1 x_2, x_2 x_1) \tag{2.55}$$

nehmen die Skalarprodukte die Form

$$C_2(\vec{x}) \cdot C_2(\vec{y}) = x_1^2 y_1^2 + x_2^2 y_2^2 + 2 x_1 x_2 y_1 y_2 = (\vec{x} \cdot \vec{y})^2 \tag{2.56}$$

an. Dann ist die Kernfunktion K das Quadrat der Skalarprodukte im Eingangsraum $\mathbb{R}^2$. Dies lässt sich verallgemeinern für beliebige $N, d \in \mathbb{N}$:

$$K(\vec{x}, \vec{y}) = C_d(\vec{x}) \cdot C_d(\vec{y}) = (\vec{x} \cdot \vec{y})^d. \tag{2.57}$$

Beweis: Man kann direkt ausrechnen:

$$\begin{aligned} C_d(\vec{x}) \cdot C_d(\vec{y}) &= \sum_{j_1,\ldots,j_d=1}^{N} x_{j_1} \cdot \ldots \cdot x_{j_d} \cdot y_{j_1} \cdot \ldots \cdot y_{j_d} && (2.58) \\ &= \left(\sum_{j=1}^{N} x_j \cdot y_j\right)^d = (\vec{x} \cdot \vec{y})^d. && (2.59) \end{aligned}$$

Jedoch existiert nur für diejenige Kernfunktion K eine Abbildung Φ, für die die Bedingung von *Mercer* erfüllt wird (siehe [30] und [7]):

> Es existiert eine Abbildung Φ und eine Kernfunktion
>
> $$K(\vec{x}, \vec{y}) = \sum_i \Phi(\vec{x})_i \Phi(\vec{y})_i \tag{2.60}$$
>
> genau dann, wenn für alle $g(x)$ mit
>
> $$\int g(x)^2 dx \quad \text{ist endlich} \tag{2.61}$$
>
> gilt:
>
> $$\int K(\vec{x}, \vec{y}) g(x) g(y)\, dx dy \geq 0. \tag{2.62}$$

Mit Hilfe Mercers Bedingung kann man zwar beweisen, ob eine Kernfunktion ein Skalarprodukt in einem Raum darstellt, aber man erfährt nicht, wie man zur Abbildung Φ oder gar zum Raum F gelangt. Es lassen sich jedoch eine Vielzahl von Lernmaschinen bei Anwendung von Kernfunktionen erzeugen, die Mercers Bedingung erfüllen. Im folgenden werden drei Kernfunktionen vorgestellt, die auch von der SVM$^{\text{light}}$ (Thorsten Joachims) unterstützt werden:

Polynom vom Grad d

$$K(\vec{x},\vec{y}) = (\vec{x} \cdot \vec{y})^d \tag{2.63}$$

Radialbasisfunktion

$$K(\vec{x},\vec{y}) = \exp\left(-\frac{||\vec{x}-\vec{y}||^2}{c}\right) \tag{2.64}$$

Sigmoidfunktion (Neuronales Netzwerk)

$$K(\vec{x},\vec{y}) = \tanh(\kappa \cdot (\vec{x} \cdot \vec{y}) + \Theta) \tag{2.65}$$

Zusammenfassend kann das Verfahren der (nichtlinearen) Support Vector Machine folgendermaßen beschrieben werden (siehe Abbildung 2.9): Die Trainingsdaten werden durch eine nichtlineare Abbildung Φ in einen höherdimensionalen Merkmalsraum F transformiert. Dort wird mit Hilfe des Support Vector

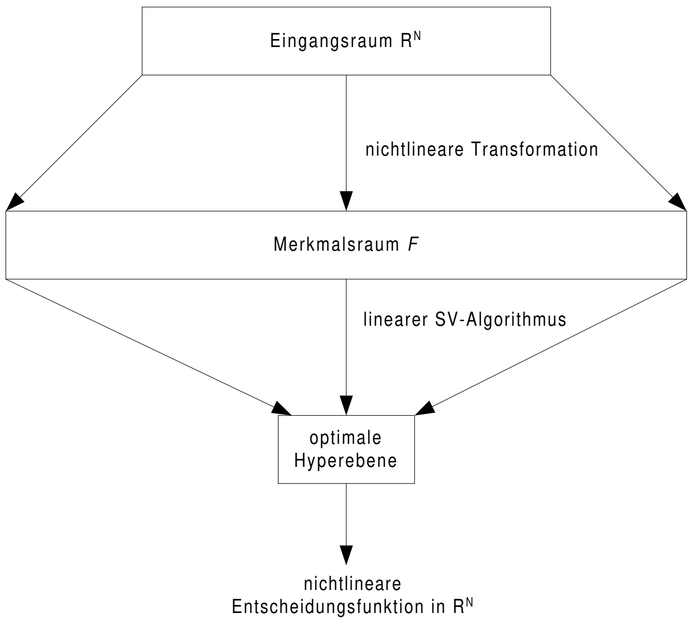


Abbildung 2.9: Verfahren der nichtlinearen SVM

Learning-Algorithmus aus Abschnitt 2.4.1 die optimale Hyperebene bestimmt, die die Trainingsdaten in zwei Klassen trennt. Fundamental ist dabei, dass die Eingangsdaten nur in Form von Skalarprodukten auftreten und somit eine Kernfunktion eingesetzt werden kann. Damit lässt sich eine nichtlineare Entscheidungsfunktion in F angeben.

2.5 Bewertung des Lernerfolgs

Nach dem Trainieren der SVM ist man daran interessiert, wie erfolgreich die Lernmaschine gelernt hat. Man möchte gerne wissen, wie gut unbekannte Beispiele klassifiziert werden, d.h. wie gut die Lernmaschine generalisiert. Damit ist es dann möglich, unterschiedliche Versuchsreihen miteinander zu vergleichen und somit die optimale Methode und Parametrierung zu finden. Dadurch kann auch entschieden werden, ob diese Lernmethode überhaupt in der Praxis eingesetzt werden kann, da viele Anwendungen eine maximale Fehlerwahrscheinlichkeit voraussetzen.

Zunächst werden einige Leistungsmaße vorgestellt, die mit unterschiedlichen Methoden gemessen werden können. Anschließend werden Verfahren zur Messung dieser Leistungsmaße beschrieben. Die vorgestellten Bewertungsmethoden werden in den Experimenten in Kapitel 5.1 eingesetzt.

2.5.1 Fehlerrate

Zur Ermittlung der Fehlerrate werden Testbeispiele von einer traininerten SVM klassifiziert, wobei die Antworten der Lernmaschine mit den richtigen Antworten verglichen werden. Die (absolute) Anzahl von Fehlern f_{abs} stellt dabei den Verlust nach Gleichung 2.5 dar. Setzt man diese Anzahl von Fehlern in Bezug zur Anzahl der Beispiele aus der Testmenge T_m, so erhält man den relativen Fehler bzw. die Fehlerrate:

$$f_{rel} = \frac{f_{abs}}{|T_m|}. \tag{2.66}$$

2.5.2 Recall, Precision und F_1

Neben der Fehlerrate existieren noch weitere Bewertungsmaße für Mustererkennungsprobleme. Die Fehlerrate liefert kein gutes Leistungsmaß, wenn die Anzahl der Beispiele aus der einen Klasse zahlenmäßig stark überwiegen. Dies wird durch ein kleines Beispiel verdeutlicht:

99% aller Trainingsbeispiele befinden sich in der negativen Klasse. Eine Lernmaschine, die alle Testbeispiele negativ klassifiziert, hat eine Fehlerrate von 1%, da mit einer Wahrscheinlichkeit von 1% ein positives Beispiel untersucht wird, das immer fehlerhaft erkannt wird. Die Fehlerrate scheint hier einen hohen Lernerfolg zu versprechen, man kann jedoch nicht davon ausgehen, dass die Lernmaschine gut gelernt hat. Hier versprechen andere Bewertungsmaße bessere Abschätzungen.

Als *Recall* $Rec(h)$ einer Klassifikationsregel h bezeichnet man ein Leistungsmaß, dass die Wahrscheinlichkeit für eine korrekte Klassifikation eines positiven Beispiels angibt. Dagegen versteht man unter der *Precision* $Prec(h)$ die Wahrscheinlichkeit, dass ein positiv klassifiziertes Beispiel tatsächlich positiv ist. Die Lernmaschine aus dem obigen Beispiel hat demnach einen Recall und eine Precision von 0%.

Sei n_+ die Anzahl aller positiven Beispiele der Testmenge und a_+ (a_-) die Anzahl von positiven (negativen) Beispielen, die positiv erkannt werden, so lassen sich die beiden Gütemaße wie folgt definieren:

$$Rec(h) = \frac{a_+}{n_+} \tag{2.67}$$

$$Prec(h) = \frac{a_+}{a_+ + a_-}. \tag{2.68}$$

Zwischen einem hohen Wert für Recall und einem hohen Wert für Precision muss abgewägt werden. Ein Maß, das beide Metriken kombiniert, ist unter den Namen F_1 bekannt und stellt das geometrische Mittel zwischen Recall und Precision dar:

$$F_1 = \sqrt{Rec(h)^2 + Prec(h)^2} \tag{2.69}$$

2.5.3 Klassifikation einer Testmenge

Eine einfache Methode zur Messung der Leistungsfähigkeit einer SVM besteht darin, eine Testmenge T_m aus der zur Verfügung stehenden Trainingsmenge S_n auszuwählen, und auf der verbleibenden Menge $S_n \setminus T_m$ die SVM zu trainieren. Die Beispiele $(\vec{x}, y) \in T_m$ werden dann von der trainierten SVM klassifiziert. Aus den Fehlklassifikationen lassen sich die oben beschriebenen Leistungsmaße berechnen.

Bei dem Verfahren muss vorgegeben werden, wie hoch der Anteil der Testmenge zu der gesamten Menge sein soll. Problematisch bei diesem Verfahren ist, dass zufällig eine „gute" Testmenge ausgewählt werden könnte, die keine allgemeingültige Aussage über den Lernerfolg hat. Für eine schnelle Bewertung kann diese Methode jedoch herangezogen werden, da die Lernmaschine nur einmal trainiert werden muss.

2.5.4 Kreuzvalidierung

Die Kreuzvalidierung vermeidet den Nachteil der zuvor beschriebenen Methode, da sie nacheinander mehrere disjunkte Testmengen gleicher Größe auswählt. Die Beispiele der Testmenge werden wieder von einer SVM klassifiziert, die auf der Restmenge trainiert wurde. Anschließend kann aus den Ergebnissen ein Mittelwert gebildet werden. Dieses Verfahren ist stabiler als das vorhergehende, da alle Beispiele einmal klassifiziert werden. Der Rechenaufwand ist allerdings wesentlich höher, da mehrere Trainingsdurchläufe notwendig sind. Besteht beispielsweise die Testmenge aus 20% der gesamten Daten, so sind 5 Lerndurchgänge erforderlich. Je kleiner die Testmenge T_m wird, desto größer wird die Ausführungszeit.

Wird die Testmenge so verkleinert, dass sie nur noch ein Beispiel enthält, so führt dies zur *Leave-One-Out*-Schätzung von Lunts und Brailovskiy [22]. Aus der Trainingsmenge wird nacheinander jeweils ein Beispiel, ausgewählt. Auf der verbleibenden Menge $S_n \setminus (\vec{x}_i, y_i)$ wird die SVM trainiert. Das ausgewählte Beispiel wird nun von der SVM klassifiziert. Wird ein Beispiel falsch klassifiziert, so spricht man von einem Leave-One-Out-Fehler. Die Anzahl aller Fehler dividiert durch die Anzahl der Beispiele ergibt die Leave-One-Out-Schätzung des Generalisierungsfehlers (vgl. [20]). Man erkennt sofort, dass die Berechnung der Schätzung sehr ineffizient ist, da n Lerndurchläufe erforderlich sind. Für praktische Anwendungen ist dieser Rechenaufwand zu hoch.

2.5.5 $\xi\alpha$-Schätzer

Die von Joachims in [20] vorgestellten $\xi\alpha$-Schätzer basieren auf der Leave-one-out-Schätzung, beanspruchen jedoch um Größenordnungen weniger Rechenzeit, da kein wiederholtes Trainieren der SVM erforderlich ist. Die Schätzer können direkt nach dem Trainieren der Lernmaschine berechnet werden. Als Eingaben für die Berechnung sind lediglich die Vektoren $\vec{\alpha}$ des dualen Optimierungsproblems und $\vec{\xi}$ des primalen Problems notwendig. $\vec{\xi}$ und $\vec{\alpha}$ stehen nach dem Lernen ohne zusätzliche Berechnung zur Verfügung. Der $\xi\alpha$-Schätzer für die Fehlerrate wird dabei folgendermaßen definiert (siehe [20]):

$$Err^n_{\xi\alpha}(h_L) = \frac{d}{n} \quad \text{mit } d = |\{i : (\rho\alpha_i R^2_\triangle + \xi_i) \geq 1\}| \tag{2.70}$$

Die Größe d zählt hierbei die Anzahl von Trainingsbeispielen, für die die Ungleichung

$$\rho\alpha_i R^2_\triangle + \xi_i \geq 0 \tag{2.71}$$

erfüllt wird. Wenn ein Trainingsbeispiel $(\vec{x}_i, y_i)$ von der SVM, die auf der Submenge $S_n \setminus (\vec{x}_i, y_i)$ trainiert wurde, fehlerhaft klassifiziert wird, dann muss die Ungleichung 2.71 erfüllt sein. Daraus folgt, dass d eine obere Schranke für die Anzahl von Leave-One-Out-Fehlern ist. Den Beweis dafür liefert Joachims in [20]. Der $\xi\alpha$-Schätzer tendiert allerdings dazu, die tatsächliche Fehlerrate zu überschätzen.

Joachims zeigt auch, wie man mit Hilfe der $\xi\alpha$-Schätzer die Leistungsmaße Precision und Recall für eine SVM abschätzen kann. Diese Gütemaße werden folgendermaßen definiert:

$$\begin{aligned} Rec^n_{\xi\alpha}(h_L) &= 1 - \frac{d_+}{n_+} && (2.72)\\ Prec^n_{\xi\alpha}(h_L) &= \frac{n_+ + d_+}{n_+ - d_+ + d_-} && (2.73) \end{aligned}$$

mit

$$\begin{aligned} d_+ &= |\{i : y_i = 1 \wedge (\rho\alpha_i R^2_\triangle + \xi_i) \geq 1\}| && (2.74)\\ d_- &= |\{i : y_i = -1 \wedge (\rho\alpha_i R^2_\triangle + \xi_i) \geq 1\}| && (2.75)\\ n_+ &= |\{i : y_i = 1\}|. && (2.76) \end{aligned}$$

d_+ (d_-) ist dabei die Anzahl der positiven (negativen) Trainingsbeispiele, für die die Ungleichung 2.71 erfüllt wird. n_+ ist die gesamte Anzahl der positiven Beispiele der Trainingsmenge. Die Leistungsmaße $Rec(h_L)$ und $Prec(h_L)$ sind im Mittel kleiner als die tatsächlichen Werte (siehe [20]).

2.6 Fuzzy-Clusteranalyse

Bisher wurden ausschließlich überwachte Lernverfahren beschrieben. Dieses Kapitel befasst sich mit einem unüberwachten Lernverfahren, bei dem die Beispiele nicht von einem Supervisor markiert werden. Das Lernverfahren ordnet selber die Beispiele in anonyme Teilmengen ein.

Im Kapitel 5.1.1 wird die Fuzzy-Clusteranalyse als Methode für die Datenvorverarbeitung verwendet. Daher werden hier die Grundlagen der Theorie erläutert. Eine Einführung in dieses Thema bieten Höppner, Klawonn und Kruse in [14] sowie Barnutz in [2].

2.6.1 Clusteranalyse

Die Clusteranalyse befasst sich allgemein mit dem Auffinden von Strukturen und Gruppierungen von Daten. Mit der Einführung der Fuzzy-Menge durch Zadeh wurde ein Objekt definiert, das die mathematische Modellierung unscharfer Aussagen zulässt (siehe [32]). Die *Fuzzy*-Clusteranalyse verzichtet auf eine eindeutige Zuordnung der Daten zu Klassen oder Clustern und berechnet stattdessen Zugehörigkeitswerte, die ausdrücken, wie stark ein Punkt einem Cluster angehört. Das allgemeine Ziel ist, eine gegebene Menge von Daten oder Objekten in Cluster (Teilmengen, Gruppen, Klassen) einzuteilen, wobei folgende Eigenschaften gelten sollen:

- Homogenität innerhalb der Cluster und
- Heterogenität zwischen den Clustern.

Daraus folgt, dass die Daten eines Clusters möglichst ähnlich sein sollen. Der Begriff der Ähnlichkeit muss dabei noch präzisiert werden.

Ähnlichkeitsmaße

In vielen Anwendungsfällen sind die Daten reellwertige Vektoren, daher kann der euklidische Abstand zwischen den Punkten als Maß für die Unähnlichkeit verwendet werden. Die Ähnlichkeits- bzw. Distanzfunktion $d(x_k, x_l)$ misst den Abstand zwischen zwei Objekten x_k und x_l im N-dimensionalen Raum und muss folgende Voraussetzungen erfüllen:

$$d(x_k, x_l) = d_{kl} \geq 0 \tag{2.77}$$
$$d_{kl} = 0 \Leftrightarrow x_k = x_l \tag{2.78}$$
$$d_{kl} = d_{lk}. \tag{2.79}$$

Ist das Distanzmaß ein metrisches Maß, so muss zusätzlich noch die Dreiecksungleichung erfüllt werden:

$$d_{kl} \leq d_{kj} + d_{jl}. \tag{2.80}$$

Häufig werden L_p-Distanzen als Ähnlichkeitsmaße verwendet:

$$d_{kl}^p = \left(\sum_{j=1}^{N} |x_{kj} - x_{lj}|^p \right)^{\frac{1}{p}}, \quad \text{für } p > 0, \tag{2.81}$$

wobei sich für $p = 2$ die bekannte euklidische Distanz

$$d_{kl}^E = \sqrt{\sum_{j=1}^{N} |x_{kj} - x_{lj}|^2} \tag{2.82}$$

ergibt.

Skalierung

Eine geeignete Skalierung der Daten ist wichtig für die Einteilung der Cluster. Eine Änderung der Skalierung kann dazu führen, dass sich die Zuordnung der

Punkte zu Clustern ebenfalls verändert. Ein Beispiel zur Veranschaulichung dieses Zusammenhangs wird in Abbildung 2.10 dargestellt. Dieses kleine Beispiel könnte in vereinfachter Form einen Ausschnitt aus den Analyse-Daten darstellen. Wie man in Abbildung 2.10 sieht, werden im linken Diagramm zwei Cluster

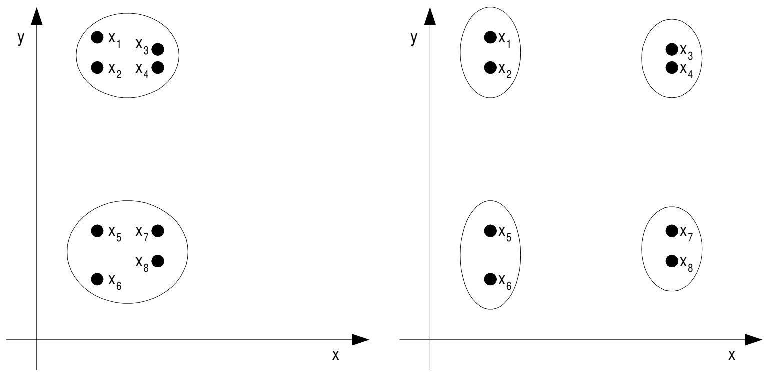


Abbildung 2.10: Einfluss der Skalierung auf die Clustereinteilung

gefunden. Die Punkte x_1, x_2, x_3, x_4 und x_5, x_6, x_7, x_8 werden jeweils zu einem Cluster zusammengefasst. Wird nun die Skalierung der x-Achse verändert, so dass die Punkte einen größeren Abstand bezüglich x bekommen, dann wird eine andere Clustereinteilung gefunden.

Es ist also bei der Skalierung der Wertebereiche der Vektorkomponenten darauf zu achten, dass keine unerwünschten Clustereinteilungen zustande kommen. So könnte eine logarithmische Einteilung eines Wertebereichs dazu führen, dass im Bereich kleinerer Werte mehrere Cluster gefunden werden als bei einer linearen Skalierung, da kleine Werte stärker voneinander getrennt werden.

Techniken der Clusteranalyse

Es gibt in der Literatur zahlreiche klassische Clustering-Verfahren, die jedoch nicht alle aufgeführt werden können. Es werden hier lediglich die unterschiedlichen Techniken der Clusteranalyse vorgestellt (siehe [14]):

- *unvollständige Clusteranalyseverfahren:* Mehrdimensionale Daten werden einer Dimensionsreduktion unterzogen, um sie zwei- oder dreidimensional graphisch darzustellen. Die Clustereinteilung erfolgt dann manuell durch augenscheinliche Betrachtung der Daten.

- *deterministische Clusteranalyseverfahren:* Bei diesem Verfahren wird jedes Datum genau einem Cluster zugeordnet.

- *überlappende Clusteranalyseverfahren:* Hier wird jedes Datum mindestens einem Cluster zugeordnet.

- *probabilistische Clusteranalyseverfahren:* Für jedes Datum wird eine Wahrscheinlichkeitsverteilung über den Clustern bestimmt, die angibt, mit welcher Wahrscheinlichkeit ein Datum einem Cluster zugeordnet wird. Die Summe der Wahrscheinlichkeiten für ein Datum beträgt eins.

- *possibilistische Clusteranalyseverfahren:* Jedem Datum wird ein Zugehörigkeitsgrad zugeordnet, der ausdrückt, inwieweit das Datum zum betreffenden Cluster gehört. Die Nebenbedingung der probabilistischen Clusteranalyseverfahren, dass die Summe der Wahrscheinlichkeiten eines Datums eins ergeben muss, wird hier fallengelassen.
- *hierarchische Clusteranalyseverfahren:* Diese Verfahren unterteilen die Daten in mehreren Schritten von einem einzigen Cluster in immer feinere Cluster oder fassen umgekehrt kleine Klassen stufenweise zu gröberen zusammen.
- *Objective-Function Clusteranalyseverfahren:* Den Objective-Function Methoden liegt eine zu optimierende Ziel- oder Bewertungsfunktion zugrunde, die jeder möglichen Clustereinteilung einen Güte- oder Fehlerwert zuordnet.

Die meisten Fuzzy-Clustering-Verfahren zählen zu den Objective-Function Methoden. Daher werden im folgenden die Grundlagen von Bewertungsfunktionen für die Fuzzy-Clusteranalyse erklärt.

2.6.2 Bewertungsfunktion und Kriterium

Um die vielen möglichen Lösungen von der oder den richtigen Lösungen zu unterscheiden, benötigt man ein *Bewertungskriterium*, das in Form einer *Bewertungsfunktion b* eingeführt wird. Diese Funktion wird im allgemeinen dazu herangezogen, um unterschiedliche Lösungen desselben Problems zu vergleichen. Durch die Angabe einer Bewertungsfunktion lässt sich je nach Problemstellung die Lösung durch Bestimmung des Maximums bzw. Minimums der Funktion b definieren. Man erhält damit ein Optimierungsproblem.

Eine Abbildung $f : X \to K$ repräsentiert das Ergebnis einer Datenanalyse durch die Abbildung einer speziellen, vorliegenden Datenmenge $X \subseteq D$ auf ein mögliches Ergebnis $K \subseteq E$. Man nennt D Datenraum und E Ergebnisraum.

Eine oft verwendete Bewertungsfunktion ergibt sich aus folgender Grundfunktion (Verallgemeinerung der kleinsten Fehlerquadrate):

$$b(f) = \sum_{x \in X} \sum_{k \in K} f^w(x)(k) \cdot d^2(x,k). \tag{2.83}$$

Der Parameter w stellt dabei den Fuzzyparamter[2] dar und steuert den Einfluss kleiner Zugehörigkeitswerte. Bei $w = 1$ geht der Algorithmus in seinen Vorgänger, den Hard-c-Means, über. Bei größeren Werten werden die Clustergrenzen unschärfer, bis schließlich im Grenzfall ($w = \infty$) alle Punkte einem Cluster angehören.

Liegt ein Punkt nah an einem Cluster, so ist der Zugehörigkeitsgrad zu dem Cluster ungefähr eins und die Distanz nahe null. Zu entfernt liegenden Clustern ist dagegen die Zugehörigkeit sehr klein und die Distanz wird größer. Durch die Produktbildung $f^w(x)(k) \cdot d^2(x,k)$ werden bei der Summe somit in beiden Fällen kleine Werte ermittelt, weil einmal der Abstand klein und das andere Mal die Zugehörigkeit gering ist. Damit ist auch klar, wie das Kriterium an die Bewertungsfunktion lautet, sie muss für die Lösung ein Minimum annehmen.

[2] auch Fuzzifier genannt

Bei der probabilistischen Clustereinteilung müssen zusätzlich folgende Bedingungen für die Zugehörigkeitsgrade gelten:

$$\forall x \in X : \sum_{k \in K} f(x)(k) = 1, \tag{2.84}$$

$$\forall k \in K : \sum_{x \in X} f(x)(k) > 0. \tag{2.85}$$

Satz: Nimmt die Bewertungsfunktion für alle probabilistischen Clustereinteilungen ein Minimum an, so gilt $\forall x \in X$ und $\forall k \in K$ (siehe [14]):

$$f(x)(k) \mapsto \begin{cases} \frac{1}{\sum_{j \in K} \left(\frac{d^2(x,k)}{d^2(x,j)}\right)^{\frac{1}{w-1}}} & \text{für } I_x = \emptyset \\ \sum_{i \in I_x} f(x)(i) = 1 & \text{für } I_x \neq \emptyset, k \in I_x \\ 0 & \text{für } I_x \neq \emptyset, k \notin I_x, \end{cases} \tag{2.86}$$

wobei $I_x := \{j \in K | d(x,j) = 0\}$.

Der Beweis dafür soll hier nur skizziert werden: Setze $u := f(x)$ und $u_k := f(x)(k)$. Ein notwendiges Kriterium für ein Minimum ist eine Nullstelle in der Ableitung der Bewertungsfunktion. Die partiellen Ableitungen sind also gleich null zu setzen. Die Gültigkeit der Nebenbedingung 2.84 lässt sich durch die Einführung eines Lagrange-Multiplikators λ ebenfalls durch eine Nullstelle in der partiellen Ableitung nach λ ausdrücken. Die partiellen Ableitungen sind dann:

$$\frac{\partial}{\partial \lambda} b_x(\lambda, u) = \left(\sum_{k \in K} u_k\right) - 1 \tag{2.87}$$

$$\frac{\partial}{\partial u_k} b_x(\lambda, u) = w \cdot u_k^{w-1} \cdot d^2(x,k) - \lambda. \tag{2.88}$$

Aus der Gleichung 2.88 ergibt sich durch Nullsetzen und Auflösen nach u_k:

$$u_k = \left(\frac{\lambda}{w \cdot d^2(x,k)}\right)^{\frac{1}{w-1}}. \tag{2.89}$$

Setzt man die Gleichung 2.87 gleich null und substituiert u_k, so erhält man:

$$1 = \sum_{j \in K} \left(\frac{\lambda}{w \cdot d^2(x,j)}\right)^{\frac{1}{w-1}} \tag{2.90}$$

$$= \left(\frac{\lambda}{w}\right)^{\frac{1}{w-1}} \sum_{j \in K} \left(\frac{1}{d^2(x,j)}\right)^{\frac{1}{w-1}} \tag{2.91}$$

Durch Umstellen bekommt man:

$$\left(\frac{\lambda}{w}\right)^{\frac{1}{w-1}} = \frac{1}{\sum_{j \in K} \left(\frac{1}{d^2(x,j)}\right)^{\frac{1}{w-1}}}. \tag{2.92}$$

Setzt man dies in Gleichung 2.89 ein, so ergibt sich:

$$u_k = \frac{1}{\sum_{j \in K} \left(\frac{d^2(x,k)}{d^2(x,j)} \right)^{\frac{1}{w-1}}}. \tag{2.93}$$

Es bleibt noch offen, dass $d(x,j) = 0$ für mindestens ein $j \in K$ gilt. In diesem Fall liegt das Datum x genau im Cluster j. Hier ist das Minimierungsproblem leicht zu lösen, denn $b_x(\lambda, u) = 0$ und Gleichung 2.84 gelten, falls $u_k = 0$ für alle $k \notin I_k$. Der ausführliche Beweis ist in [14] nachzulesen.

2.6.3 Fuzzy-c-Means-Algorithmus

Nach den Vorüberlegungen über Bewertungsfunktion und Kriterium wird nun ein Algorithmus zur Berechnung der Cluster eingeführt. Der Algorithmus gehört zu den probabilistischen Clusteranalyseverfahren. Dabei werden in einem iterativen Vorgang die Clusterzentren so verschoben und die Zugehörigkeiten so verändert, dass die Bewertungsfunktion minimiert wird.

Der Fuzzy-c-Means-Algorithmus erkennt hyperkugelförmige Punktwolken im p-dimensionalen Raum. Die Entwicklung dieses Algorithmus war die erste aller Clustering-Verfahren. Zuerst wurde ein Hard-c-Means-Algorithmus bekannt, der eine harte Einteilung der Cluster vornimmt. Die Fuzzy-Variante wurde anschließend von Dunn vorgestellt und von Bezdek schließlich durch Einführung des Fuzzifiers zur endgültigen Variante verallgemeinert.

Die Zugehörigkeit der Punkte zu den Clustern wird in einer *Fuzzy-Partitionsmatrix* dargestellt. Dabei werden die Zugehörigkeitswerte μ_{ik} der m Vektoren zu den c Clustern in einer $c \times m$-Matrix angegeben:

$$U = \begin{pmatrix} \mu_{11} & \cdots & \mu_{1m} \\ \vdots & \ddots & \vdots \\ \mu_{c1} & \cdots & \mu_{cm}. \end{pmatrix} \tag{2.94}$$

Die Cluster[3] $v_1, \ldots, v_c$ werden bei diesem Algorithmus als gleich groß angenommen und durch ihren Mittelpunkt dargestellt. Als Distanzmaß wird der euklidische Abstand (siehe Gleichung 2.82) verwendet. Der FCM-Algorithmus verwendet die Objective-Function-Methode, um ein (lokales) Optimum zu berechnen. Es wird ein Iterationsverfahren eingesetzt, bei dem die Grundfunktion 2.83 als Bewertungsfunktion schrittweise minimiert wird.

Die Anzahl der Cluster ist vorab nicht bekannt und muss ebenfalls bestimmt werden. Der Algorithmus läuft dabei folgendermaßen ab:

Gegeben sei eine Datenmenge $X = \{x_1, x_2, x_3, \ldots, x_m\}$.

1. Wähle die Clusteranzahl c
2. Wähle den Fuzzyparameter w
3. Wähle eine Anfangsbelegung der Fuzzy-Partitionsmatrix U
4. Wähle eine Abbruchbedingung ϵ

[3] auch als Prototypen bezeichnet

5. Setze den Schleifenzähler $s = 0$

6. Berechne die Prototypen unter Verwendung der Partitionsmatrix:

$$v_i^{(s)} = \frac{\sum_{k=1}^{m} \left(\mu_{ik}^{(s)}\right)^w x_k}{\sum_{k=1}^{m} \left(\mu_{ik}^{(s)}\right)^w} \quad \forall i = 1, \dots, c. \tag{2.95}$$

   In diesem Iterationsschritt werden im allgemeinen die Prototypen so bestimmt, dass die Bewertungsfunktion b minimal wird. Dieser Algorithmus benutzt eine Form der verallgemeinerten Mittelwertbildung[4], woher der Algorithmus auch seinen Namen *Fuzzy-c-Means* hat. Wird b bezüglich allen probabilistischen Clustereinteilungen bei gegebenen Zugehörigkeiten minimiert, so gilt die Gleichung 2.95. Der Beweis kann in [14] nachgelesen werden.

7. Sei $I_k = \{i \in \{1, \dots, c\} | d(x_k, v_l) = 0\}$ und r_k die Anzahl der Elemente in I_k. Berechne die neue Fuzzy-Partitionsmatrix nach Satz 2.86:

   falls $I_k = \emptyset \quad \forall\, k = 1, \dots, m$:

$$\mu_{ik}^{(s+1)} = \frac{\left(\frac{1}{d(x_k, v_i)^{(s)}}\right)^{\frac{1}{w-1}}}{\sum_{j=1}^{c} \left(\frac{1}{d(x_k, v_j)^{(s)}}\right)^{\frac{1}{w-1}}} \tag{2.96}$$

   falls $I_k \neq \emptyset\ \forall k = 1, \dots, m$:

$$\mu_{ik}^{(s+1)} = \begin{cases} 0 & \text{für } \forall i \in I_k^C \\ \frac{1}{r_k} & \text{für } \forall i \in I_k. \end{cases} \tag{2.97}$$

8. Breche den Algorithmus ab, falls

$$\left|\left|U^{(s+1)} - U^{(s)}\right|\right| \leq \epsilon, \tag{2.98}$$

   ansonsten erhöhe den Schleifenzähler s um eins und gehe zu Schritt 6.

Der FCM-Algorithmus besitzt sehr gute Konvergenzeigenschaften, findet aber unter Umständen nur ein lokales Minimum. Ein weiterer Nachteil des vorgestellten Algorithmus ist, dass die Anzahl der Cluster angegeben werden muss, aber oft nicht vorher bekannt ist. Trotz dieser Einschränkungen stellte Barnutz empirisch fest, dass das Verfahren sehr gute Ergebnisse in der Praxis liefert (siehe [2]).

Bestimmung der optimalen Clusteranzahl

Für die Bestimmung der optimalen Anzahl von Clustern bietet sich die Möglichkeit an, den FCM-Algorithmus mit unterschiedlichen Clusteranzahlen durchlaufen zu lassen und die Ergebnisse zu bewerten. Dazu ist eine obere Schranke c_{max} zu schätzen und für jedes $c \in \{2, 3, \dots, c_{max}\}$ eine Clustereinteilung nach dem FCM-Algorithmus durchzuführen und ein Gütemaß zu berechnen. Die optimale

[4] Der einzige Unterschied zur Berechnung des Standard-Mittelwerts ist die Gewichtung der Zugehörigkeitswerte.

Anzahl c von Clustern liegt bei der Clustereinteilung vor, bei der das Gütemaß optimal ist. Somit wird der FCM-Algorithmus $c_{max} - 1$-mal durchgeführt.

Es wird somit ein globales Gütemaß benötigt, das die gesamte Clustereinteilung bewerten kann. Es gibt unterschiedliche Gütemaße, allerdings stellte Barnutz hier fest, dass nur das Verfahren *Compactness and Separation* sinnvoll eingesetzt werden kann. Daher wird auch nur dieses Gütemaß näher erläutert. Dieses Maß wurde von Xie und Beni [31] vorgestellt und wird wie folgt definiert:

$$S(f) = \frac{\sum_{i=1}^{c} \sum_{k=1}^{m} \mu_{i,k}^2 ||v_i - x_k||^2}{m \min_{i,j} ||v_i - v_j||^2}. \tag{2.99}$$

Die bekannte Grundfunktion für b (siehe 2.83) wird in Bezug gesetzt zum m-fachen Minimum des quadratischen Abstands zweier Cluster. Dieses Gütemaß versucht dadurch, die Kompaktheit der Cluster und die Trennung der einzelnen Cluster untereinander ganzheitlich zu bewerten. Kleinere Werte geben dabei kompaktere und besser getrennte Cluster an.

Initialisierung

Bevor der FCM-Algorithmus durchgeführt werden kann, muss die ursprüngliche Lage der Zentren der Cluster angegeben werden. Man benötigt also vor dem ersten Iterationsschritt eine Belegung der Prototypen p_i. Ist die Lage der Cluster unbekannt, kann hierfür eine zufällige Anfangsbelegung gewählt werden, die die Bedingungen für eine Fuzzy-Partitionsmatrix einhalten. Die Cluster werden dadurch irgendwo im Raum zufällig verteilt.

Eine andere Möglichkeit besteht darin, die Cluster in gleichen Abständen anzuordnen. Das Verfahren wird als Zentrumsinitialisierung bezeichnet. Dabei wird um den Datenraum der kleinstmögliche quaderförmige Raum gelegt. Danach werden die Cluster in gleichem Abstand entlang einer Diagonalen durch diesen Raum positioniert. Die Zentren der Cluster stellen die Anfangsbelegung für den FCM-Algorithmus dar.

Abbildung 2.11 stellt abschließend in einem Flussdiagramm den gesamten Ablauf des Fuzzy-Clustering-Verfahrens dar. Die Ergebnisse der Analyse können letztlich noch zwei- bzw. dreidimensional grafisch visualisiert werden, um sie besser auswerten zu können.

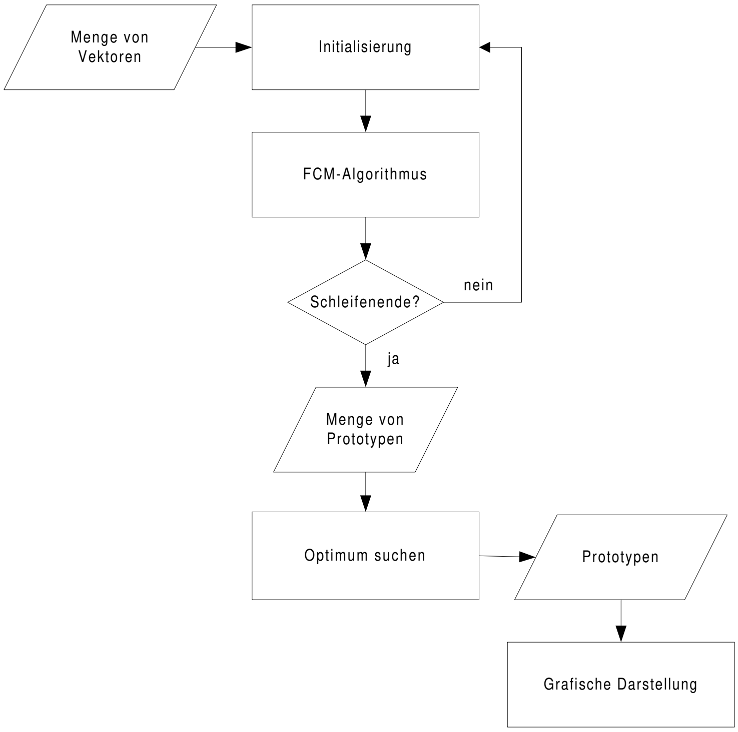


Abbildung 2.11: Flussdiagramm zum Verfahren der Fuzzy-Clusteranalyse

Kapitel 3

Herkunft und Aufbereitung der Daten

Dieses Kapitel beschreibt die zu analysierenden Daten. Dazu werden zunächst die erforderlichen Grundlagen der ISDN-Technik erklärt. Anschließend wird die Aufzeichnung von Testszenarien vorgestellt. Daran schließt sich eine Darstellung der Aufbereitung der Daten an. Es wird erläutert, wie die Daten zu den Ausgangsdaten für das Lernverfahren weiterverarbeitet werden.

3.1 Beschreibung der Daten

In dieser Arbeit wird die Kommunikation im ISDN[1] betrachtet. Der Begriff ISDN wurde von der ITU-T[2] (ehemals CCITT[3]) geprägt und beschreibt ein Telekommunikations-Service-Paket (siehe [12]).

3.1.1 Grundlagen des ISDN

Der Teilnehmeranschluss sieht einen gesonderten Kanal für Signalisierungsinformationen vor, der als *D-Kanal* bezeichnet wird (out band signalling). Beim ISDN-Basisanschluss stehen zwei B-Kanäle zu je 64kBit/s für Nutzdaten und ein D-Kanal zu 16kBit/s zur Verfügung. Hier stellt die Teilnehmerleitung die S_0-Schnittstelle zur Verfügung, an die die Endsysteme angeschlossen werden. Weiterhin existiert ein Primärmultiplexanschluss (S_{2m}-Schnittstelle), der 30 B-Kanäle und einen D-Kanal bereitstellt mit jeweils 64kBit/s.

Der Austausch von vermittlungstechnischen Informationen zwischen Endsystemen und Netz sowie innerhalb des Netzes wird als *Signalisierung* oder *Zeichengabe* bezeichnet. Die Aufgabe der Signalisierung besteht vor allem in der Bereitstellung aller notwendigen Informationen für den Auf- und Abbau von Verbindungen (siehe [21]).

Die Zeichengabe ist nach den gleichen Prinzipien und Methoden aufgebaut, wie sie in Rechnernetzen gelten, da die Steuerung des ISDN nahezu ausschließlich durch Rechner erfolgt. Dementsprechend basieren die Spezifikationen auf

[1] Intergrated Services Digital Network
[2] International Telecommunication Union
[3] Comité Consultatif International Télégraphic et Téléphonique

dem Referenzmodell für offene Systeme (Reference Model for Open Systems Interconnection, ISO-OSI-Model, siehe [15]).

3.1.2 Das LAP-D-Protokoll

Im D-Kanal wurde auf Protokollschicht 2 des Referenzmodells (Datensicherungsschicht) das Protokoll LAP-D[4] entwickelt, das eine spezielle Variante des HDLC[5] ist. Die Aufgabe des LAP-D ist es, die Informationen der Schicht 3 gesichert zu übertragen. Dieses Protokoll wurde von der ITU-T in der Empfehlung Q.920 [16] und Q.921 [18] standardisiert und von der ETSI in der Norm ETS 300 125 [8] übernommen.

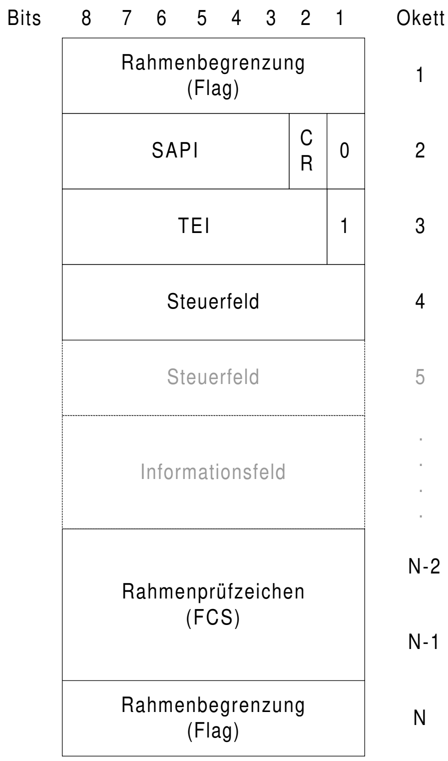


Abbildung 3.1: Rahmenaufbau LAP-D

Alle Daten der Datensicherungsschicht werden in Rahmen übertragen gemäß Abbildung 3.1 (siehe [8]). Diese Rahmen werden durch ein einheitliches Oktett[6] (01111110) begrenzt. Diese Rahmenbegrenzung kommt nicht innerhalb eines Rahmens vor, da ein Bitstopfen-Verfahren angewendet wird, das nach jeder fünften eins eine null einfügt. Diese zusätzlichen Bits werden vom Empfänger wieder entfernt.

[4]Link Access Procedure on D-channel
[5]High-level Data Link Control (ein bitorientiertes Protokoll)
[6]sog. Flag

Die Adresse einer logischen Schicht-2-Verbindung setzt sich aus SAPI (Service Access Point Identifier) und TEI (Terminal Endpoint Identifier) zusammen, wodurch eindeutig ein Verbindungspunkt deklariert wird. Der SAPI kennzeichnet einen Dienstzugangspunkt (SAP, Service Access Point) über den die Schicht 2 ihren Dienst der Schicht 3 zur Verfügung stellt. Mit den sechs Bits im Adressfeld für den SAPI lassen sich 64 verschiedene Werte definieren, es sind jedoch in der Richtlinie Q.921 erst vier SAPI-Werte festgelegt (vgl. Tabelle 3.1). Innerhalb eines Dienstzugangspunktes können ein oder mehrere Verbindungs-

SAPI	Funktion
0	Zeichengabeprozeduren (s-SAPI)
1	Reserviert für Zeichengabeprozeduren
16	Prozeduren für Paketkommunikation nach X.25
63	D2-Schicht Managementfunktionen (z.B. TEI-Vergabe)
sonst	Reserviert für zukünftige Anwendungen

Tabelle 3.1: Festlegung der SAPI-Werte

endpunkte bestehen. Für eine Punkt-zu-Punkt-Verbindung wird durch einen TEI eine logische Verbindung zwischen den Partnerinstanzen gekennzeichnet. Eine Ausnahme ist der Broadcast-TEI 127, mit dem ein „Rundsenden“ möglich ist. Ein Broadcast wird bei Punkt-zu-Mehrpunkt-Verbindungen grundsätzlich bei ankommenden Rufen verwendet. Die TEI-Werte von 1-63 sind in den Endgeräten fest eingestellt. Bei einem reinen Punkt-zu-Punkt-Anschluss (Anlagenanschluss) wird der Wert null eingesetzt. Die TEI-Werte von 64-126 werden in einer TEI-Vergabeprozedur von der Vermittlungsstelle zugewiesen (vgl. [21]).

Die Informationen des Steuerfeldes kennzeichnen die Art der Protokolldatenelemente. Es werden grundsätzlich drei Rahmentypen unterschieden, die im Steuerfeld codiert sind. Tabelle 3.2 führt die Rahmentypen mit den Protokolldatenelementen auf.

Rahmentyp	Protokolldatenelement
Information Frames	INFO (Information)
Supervisory Frames	RR (Receive Ready)
	RNR (Receive Not Ready)
	REJ (Reject)
Unnumbered Frames	SABME (Set Asynchronous Balance Mode Ext.)
	DM (Discconected Mode)
	UI (Unnumbered Information)
	DISC (Disconnect)
	UA (Unnumbered Acknowledgement)
	FRMR (Frame Reject)
	XID (Exchange Identification)

Tabelle 3.2: Rahmentypen im ISDN

Die Daten der Protokollschicht 3 werden in den Informationsrahmen (INFO) übertragen. Die Supervisory-Rahmen dienen der Steuerung des Datenflusses und der Quittierung und enthalten keine Nutzdaten. In diesen beiden Rahmentypen werden für den Quittungsmechanismus Sende- und Empfangsfolgenummern verwendet. Bei den Supervisory-Rahmen existiert der Rahmen RR (Receive Ready) als Anzeige für die Empfangsbereitschaft einer Station oder als positive Bestätigung für die Ankunft übertragener Rahmen. Die Unnumbered Frames dienen der Kommunikation der Managementinstanzen und werden beispielsweise für den Aufbau der Schicht 2 verwendet. Diese Rahmen verwenden nur ein Oktett für das Adressfeld und enthalten keine Folgenummern (siehe [21]).

Die LAP-D-Rahmen enthalten außerdem ein Kontrollfeld, in dem ein Bit kennzeichnet, wer der Auslöser eines Nachrichtenaustausches ist. Das sogenannte Command/Response-Bit (C/R-Bit) zeigt an, ob es sich um ein Kommando (Command) oder um eine Antwort (Response) auf ein Kommando handelt. Das C/R-Bit tritt in INFO-Rahmen immer als Kommando auf. RR-Rahmen können dagegen in beiden Varianten vorkommen.

3.1.3 Ablauf des Informationsaustausches

In der Informationsaustauschphase werden Schicht-3-Protokolldatenelemente im Informationsfeld quittiert übertragen. Abbildung 3.2 zeigt den prinzipiellen Ablauf des Informationsaustausches in einem Zeitdiagramm.

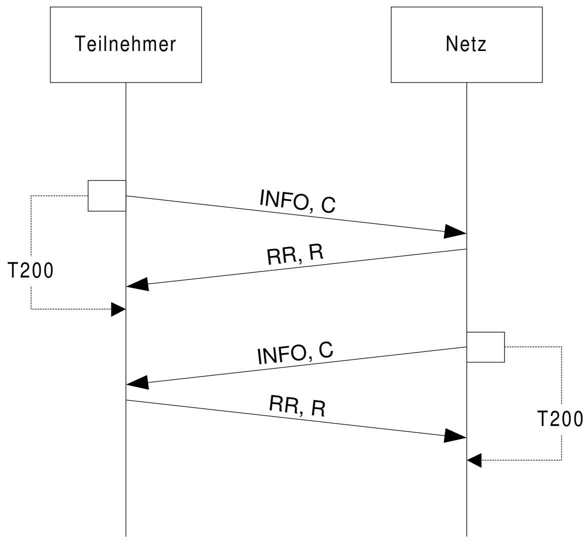


Abbildung 3.2: Ablauf des Informationsaustausches

Um einen Verlust oder ein Vertauschen der Reihenfolge von INFO-Rahmen zu verhindern, werden diese mit Sende- und Empfangsfolgenummern versehen. Steht ein INFO-Rahmen zur Übertragung an, so wird der Überwachungszeitgeber T200 (1s) gestartet, die Sendefolgenummer gesetzt und im Steuerfeld des INFO-Rahmens mit übertragen. Die Partnerinstanz quittiert diesen Rahmen, indem sie die entsprechende Empfangsfolgenummer setzt und im Steuerfeld eines RR-Rahmens übermittelt. Um diese genannten Folgenummern beim Senden und Empfangen aktualisieren zu können, sind für die beteiligten Instanzen lokale Zustandsvariablen vorgesehen, die nach jeder Initialisierungsphase auf null gesetzt werden (siehe [21]).

Die Fenstergröße gibt die maximale Anzahl noch nicht quittierter INFO-Rahmen an. Ist die Grenze erreicht, muss mindestens ein Rahmen bestätigt werden, bevor ein weiterer gesendet werden kann. Dies hat zur Folge, dass für jede Instanz eine Warteschlange existieren muss, um noch nicht quittierte INFO-Rahmen temporär zwischenzuspeichern. Eine lokale Überlast kann dann entstehen, wenn mehr INFO-Rahmen in die Warteschlange eingetragen werden als bereits gesendete INFO-Rahmen quittiert wurden. Die Gefahr einer Überlast steigt umso mehr, je größer die Fenstergröße ist und je geringer die maximale Übertragungsgeschwindigkeit ist. Die maximale Anzahl der Warteplätze muss daher begrenzt werden, um einen Überlauf der Ressourcen zu verhindern (siehe [21]).

3.1.4 Schicht 3 des D-Kanals

Auch wenn die Analyse ausschließlich auf Schicht 2 geschieht, ist zumindest ein Grundverständnis der Schicht 3 erforderlich, um die Abläufe der Signalisierung zu verstehen. Für eine ausführliche Beschreibung ist [21] zu empfehlen.

Das D3-Protokoll ist Teil des Digital Subscriber Signalling System No. 1 (DSS 1) und wird europäisch durch die Normen ETS 300 102 [9] und ETS 300 403 [10] sowie international durch die ITU-T-Empfehlung Q.931 [17] spezifiziert. Zu diesen Normen gibt es noch zahlreiche nationale und firmenspezifische Erweiterungen. Das Protokoll DSS1 wird auch als Euro-ISDN bezeichnet und in Deutschland sowie 19 weiteren europäischen Ländern seit 1993 eingesetzt. Es ermöglicht dem Anwender eine problemlose Kommunikation über die Staatsgrenzen hinweg innerhalb Europas (vgl. [12]). Neue ISDN-Anschlüsse werden schon seit langem ausschließlich gemäß Euro-ISDN-Standard installiert. Daher betrachtet diese Arbeit nur Euro-ISDN.

Die Schicht 3 enthält die Signalisierungsinformationen, die zur Steuerung und Bereitstellung der ISDN-Dienste erforderlich sind. Die Informationen werden in D3-Protokolldatenelementen und deren Parametern übertragen. An dieser Stelle soll lediglich ein typischer Verbindungsaufbau beschrieben werden wie er in den Experimenten (siehe Kapitel 3.2) vorgekommen ist. Die Abbildung 3.3 veranschaulicht ein Beispiel für einen Rufaufbau vom Endsystem aus. Das Übertragen des Protokolldatenelements *setup* leitet die Anmeldung eines Verbindungswunsches ein. In diesem Beispiel liefert der rufende Teilnehmer die Zielrufnummer 11 als auch seine eigene Ursprungsrufnummer 333 jeweils als Parameter im *setup* mit. Das Netz bestätigt diese Nachricht mit einem *setup_ack* (setup acknowledge). Anschließend zeigt das Netz mit *call_proc* (call proceeding) an, dass die Zielinformationen vollständig sind und dass der Ruf im Netz bearbeitet wird. Sobald das Zielendsystem den Ruf bestätigt hat, wird dies dem rufenden Endsystem mit *alert* mitgeteilt. Das rufende Endsystem kann diese Information dann geeignet weitermelden, z.B. durch ein akustisches Signal.

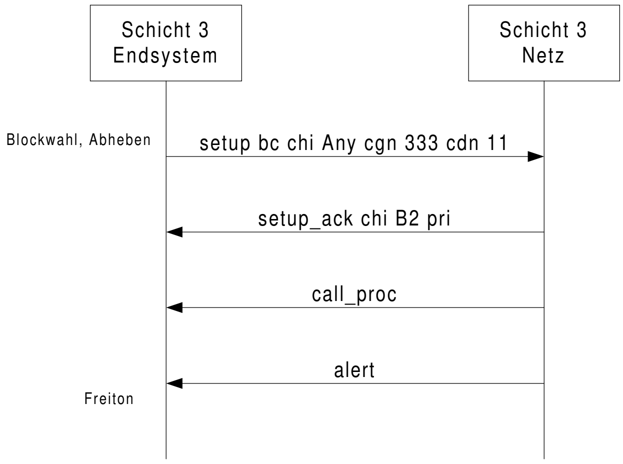


Abbildung 3.3: Ablauf eines Verbindungsaufbaus

3.2 Aufzeichnung der Daten

Um die übertragenen Datenströme auswerten zu können, müssen sie zunächst aufgezeichnet werden. Die Leitungsdaten einer ISDN-Strecke können mit einem speziellem Messinstrument, einem sogenannten *Protokollanalysator*, beobachtet werden. Sämtliche übertragenen Daten werden dabei mitgeschnitten und entsprechend den Protokollen dekodiert. In dieser Diplomarbeit wird der Protokollanalysator *EasyTrace für Windows* von der Firma Herakom GmbH verwendet. Die Hardwarekomponenten des Systems bestehen aus einer PCMCIA-Karte und einer Interfacebox, an die die zu messenden Leitungen angeschlossen werden. Die verwendete Hardware in Abbildung 3.4 bietet 4 S_0-Schnittstellen, sowie eine S_{2m}-, eine V.24- und eine X.21-Schnittstelle.

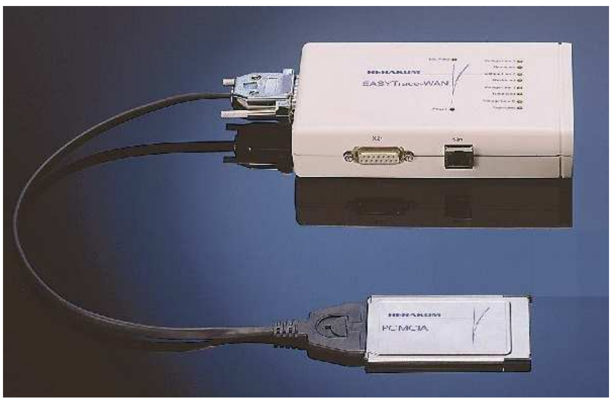


Abbildung 3.4: Hardware-Komponenten von EasyTrace

Die Software bekommt über einen Gerätetreiber die Leitungsdaten geliefert. Die aufgezeichneten Rahmen werden entsprechend den erkannten Protokollen dekodiert und zeilenweise mit Angabe des Datums, der Zeit, der Richtung und des Kanals angezeigt. Zu dem aktuell markierten Rahmen werden zusätzlich eine ausführliche Dekodierung und eine hexadezimale Ansicht dargestellt.

Dieses System kann zudem selber Daten senden und dadurch ein Gerät (Teilnehmer oder Netz) emulieren. Dabei werden gleichzeitig alle Daten auf der Leitung aufgezeichnet. Mehrere Szenarien stehen dabei zur Auswahl. Es können Rufe aufgebaut, angenommen und beendet, Dienstmerkmale getestet oder Daten übertragen werden. Für die ersten Experimente wurden mehrfach abgehende Rufe emuliert und somit Last für eine angeschlossene Telefonanlage erzeugt.

Es wurden Testszenarien durchgeführt, bei denen der Lernalgorithmus zeigen soll, dass nach bestimmten Klassifikationskriterien gelernt werden kann. Dazu müssen die gemessenen Rohdaten durch eine geeignete Merkmalsextraktion repräsentiert werden. Eine zusätzliche Vorverarbeitung dieser Merkmale soll dem Lernalgorithmus die Mustererkennung ermöglichen bzw. verbessern. Die Support Vector Machine ist in der Lage, unbekannte Beispiele als positiv oder negativ zu klassifizieren. Dafür muss definiert werden, nach welchem Kriterium die Daten unterteilt werden. Man muss also festlegen, was man unter positiv oder negativ versteht. Die Einteilung in die beiden Klassen wurde nach verschiedenen Klassifikationskriterien vorgenommen. Die folgenden vier Testszenarien wurden im Hinblick auf unterschiedliche Klassifizierung mit dem Lernalgorithmus erzeugt. Dabei soll untersucht werden, ob das Verhalten einer Kommunikation durch die Merkmale charakterisiert werden kann und in wie weit eine Mustererkennung nach verschiedenen Kriterien möglich ist. Das Ziel der Untersuchung ist die Ermittlung eines geeigneten Verfahrens, mit dem das Verhalten einer Kommunikation dargestellt und erkannt werden kann. Die Verfahren sollen dabei eine möglichst hohe Erkennungsgenauigkeit bei niedriger Laufzeit zur Berechnung der Lösung erreichen.

Um zu untersuchen, ob und in wie weit diese Anforderungen an die Lösung erfüllt werden, wurden unterschiedliche Experimente durchgeführt, die die Leistungsfähigkeit der Verfahren ermitteln. Ein Beispiel für das Lernverfahren ist dabei ein Ausschnitt mit der Dauer eines bestimmten Zeitintervalls. Eine Aufzeichnung wurde daher in mehrere Zeitintervalle unterteilt, sog. Zeitfenster, die als Beispiele für den Lernalgorithmus dienen.

Für die Versuche standen die Telekommunikationsanlagen Siemens Hicom 150 E und AGFEO AC 141 zur Verfügung. Als Protokollanalysator und Lastgenerator wurde das System EasyTrace der Firma Herakom eingesetzt. Die aufgezeichneten Daten der Emulationen wurden jeweils in einer Datei gespeichert, die erst nach der Aufzeichnung analysiert wurde. Auf eine Echtzeitanalyse wurde verzichtet.

3.2.1 Szenario 1: Emulator alleine gegen Anlage

Bei den Daten aus dem Szenario 1 wurde als Klassifikationskriterium die Auslastung des D-Kanals festgelegt. Dabei wurden die Beispiele mit hoher Auslastung der negativen Klasse zugeordnet, wogegen die positive Klasse die Beispiele mit niedriger Last beinhaltet. Die Grenze zwischen den beiden Klassen wurde willkürlich festgelegt. Bei diesem Testszenario soll der Lernalgorithmus anhand der versendeten Daten erkennen, wie hoch der Kanal ausgelastet ist.

Beim Versuchsaufbau wurde der EasyTrace-Emulator als einziges Endgerät an die Telefonanlage angeschlossen (siehe Abbildung 3.5). EasyTrace emulierte

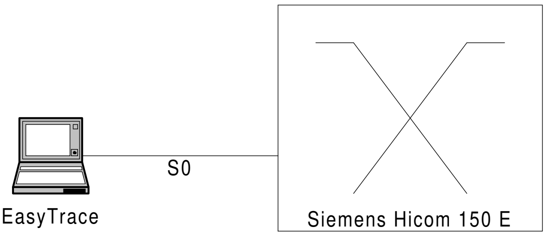


Abbildung 3.5: Versuchsaufbau Szenario 1

dabei einen Teilnehmer, der mehrfach abgehende interne Rufe auf die Mehrfachrufnummer 19 einleitet. Da kein Teilnehmer an der Telefonanlage angeschlossen ist, der diese Rufe entgegen nehmen könnte, werden die Rufe von der Anlage sofort abgeworfen. EasyTrace erzeugte also nur fehlgeschlagene Rufaufbauten. Die Tabelle 3.3 zeigt einen Ausschnitt aus der Aufzeichnung. Man erkennt, dass der Teilnehmer mit *setup* die Rufe einleitet. Die Anlage wirft darauf den Ruf sofort mit *rel_com* (release complete) ab, da kein Teilnehmer erreichbar ist (cause 63 service or option not available).

Zeit	U/N	C/R	SAP	TEI	Layer 2	Layer 3
13:33:03:674,9	U	R	0	65	NR:021 F:1 RR	
13:33:04:268,2	U	C	0	65	NS:021 NR:021 P:0 INFO	O 22 setup bc chi Any cgn 11 cdn 19
13:33:04:275,0	N	R	0	65	NR:022 F:0 RR	
13:33:04:293,1	N	C	0	65	NR:022 NS:021 P:0 INFO	D 22 rel_com cau 63 service or option not available
13:33:04:298,7	U	R	0	65	NR:022 F:0 RR	
13:33:14:253,9	U	C	0	65	NR:022 P:1 RR	
13:33:14:260,1	N	R	0	65	NR:022 F:1 RR	
13:33:17:616,8	U	C	0	65	NS:022 NR:022 P:0 INFO	O 23 setup bc chi Any cgn 11 cdn 19
13:33:17:624,3	N	R	0	65	NR:023 F:0 RR	
13:33:17:642,3	N	C	0	65	NR:023 NS:022 P:0 INFO	D 23 rel_com cau 63 service or option not available
13:33:17:647,9	U	R	0	65	NR:023 F:0 RR	
13:33:26:241,2	U	C	0	65	NS:023 NR:023 P:0 INFO	O 24 setup bc chi Any cgn 11 cdn 19
13:33:26:247,9	N	R	0	65	NR:024 F:0 RR	
13:33:26:266,1	N	C	0	65	NR:024 NS:023 P:0 INFO	D 24 rel_com cau 63 service or option not available
13:33:26:272,0	U	R	0	65	NR:024 F:0 RR	
13:33:36:226,8	U	C	0	65	NR:024 P:1 RR	

Tabelle 3.3: Auszug aus dem Trace im Szenario 1

Es wurden insgesamt zwei Versuchsreihen durchgeführt, bei denen automatisch mehrfach abgehende Rufe eingeleitet wurden. In der ersten Versuchsreihe sind die Rufabstände konstant. Es sind insgesamt 155 Beispiele erzeugt worden, von denen 61 der negativen Klasse angehören. Bei der zweiten Messreihe steuerte ein Zufallsgenerator die Zeitpunkte für die Einleitung der Rufe. Die Verteilung der Anrufabstände war dabei negativ-exponentiell. Empirische Untersuchungen haben gezeigt, dass diese Verteilung in der Praxis oft dem tatsächlich aufkommenden Telefonverkehr entspricht. Die negativ-exponentielle Verteilungsfunktion lässt sich durch eine große Anzahl unabhängiger Teilnehmer hervorrufen. Die Verteilungsfunktion lautet dabei:

$$F_A(t) = 1 - e^{-\lambda t}. \tag{3.1}$$

Dabei wird λ als Anrufrate bezeichnet. Der mittlere Anrufabstand bzw. der

Erwartungswert der Anrufabstände lässt sich aus der Anrufrate bestimmen:

$$E(T_A) = \int_0^\infty t \cdot f_A(t)\, dt. \tag{3.2}$$

Dabei bedeutet $f_A(t)$ die Verteilungsdichtefunktion der Anrufabstände. Die Ausführung der Integration ergibt

$$E(T_A) = \frac{1}{\lambda}. \tag{3.3}$$

Die eingestellten mittleren Anrufabstände der zweiten Versuchsreihe entsprechen dabei den Anrufabständen der ersten Messreihe. Aus der zweiten Versuchsreihe ergeben sich 161 Beispiele, wovon 72 in die negative Klasse gehören. Eine weitere Messreihe enthält alle Beispiele aus den ersten beiden Versuchsreihen. Es wurden pro Versuchsreihe 12 Emulationen durchgeführt. Hinzu kommt noch ein Versuch, bei dem keine Rufe erzeugt wurden, so dass insgesamt 25 Versuche mit einer Dauer von jeweils einer Stunde entstanden. Die Tabelle 3.4 zeigt die dabei eingestellten Rufabstände der durchgeführten Emulationen. Die einstellbare Dauer der Rufe kann hierbei ignoriert werden, da keine Rufe aufgebaut werden konnten.

Klasse	Rufabstand [s]
negativ	0,1
	0,3
	0,5
	0,7
	1
	3
positiv	5
	7
	10
	30
	50
	100
	Leerlauf (keine Rufe)

Tabelle 3.4: Rufabstände Szenario 1

Durch kürzere Rufabstände wird das Datenaufkommen erhöht, da mehr Rahmen pro Zeit gesendet werden. Auf dem D-Kanal wird somit die mittlere Datenrate und die Auslastung erhöht. Die kleinstmögliche Auslastung bei einem angeschlossenen Endgerät auf dem D-Kanal hat man im Leerlaufbetrieb, bei dem keine Rufe eingeleitet werden. Dabei wird jeweils nach Ablauf eines Timers von 10 Sekunden ein „Keep-alive-Polling“ durchgeführt, bei dem RR-Frames ausgetauscht werden. Bei der größtmöglichen Auslastung werden gerade so viele Frames gesendet, wie die physikalische Bandbreite auf dem D-Kanal es zulässt. Die Auslastung wurde bei den Experimenten in diesem weiten Bereich näherungsweise logarithmisch variiert.

Die Versuche sind in zwei Klassen eingeteilt worden, wobei die negative Klasse die Experimente mit hoher Auslastung umfasst. Die Grenze zur positiven Klasse wurde dabei willkürlich auf fünf Sekunden Rufabstand gesetzt, damit beide Klassen gleich viele Beispiele enthalten (siehe Tabelle 3.4). Das Lernverfahren soll aus diesen markierten Beispielen lernen die Auslastung unbekannter Beispiele zu erkennen.

Ein Nachteil dieser Experimente ist, dass die Rufe nicht lange im System verbleiben, da sie sofort von der Anlage abgeworfen werden. Dieses Szenario ist daher in dieser Form nicht besonders realistisch, es werden jedoch Aussagen über die Auslastung des D-Kanals möglich. Auf die Auslastung kann zudem auch durch Beobachten der Anzahl der gesendeten Frames geschlossen werden. Die ersten Experimente mit der SVM sollen jedoch zeigen, ob man die Auslastung des D-Kanals erkennen kann, und welches Verfahren sich dafür am besten eignet.

3.2.2 Szenario 2: Vermitteln der Rufe durch Anlage

Die Experimente dieses Szenarios sollen nicht zur Klassifikation der Auslastung auf dem D-Kanal dienen, sondern zur Erkennung der Auslastung eines angeschlossenen Endgeräts. In den Endsystemen ist ein Mikroprozessor für die Verarbeitung der Nachrichten der Signalisierung verantwortlich. Die Auslastung der CPU soll bei diesen Versuchen von der Lernmaschine erkannt werden. Bei der Untersuchung soll von den gesendeten Daten auf den internen Zustand des Endsystems geschlossen werden.

Dafür wurden neue Emulationen mit EasyTrace durchgeführt, bei denen wieder mehrfach abgehende Rufe erzeugt wurden. Die Telefonanlage vermittelt die eingeleiteten Rufe an ein angeschlossenes Telefon. Es wird hiermit ein Nachteil des ersten Szenarios vermieden, bei dem die Rufe sofort beendet worden sind. Bei diesen Versuchen werden die Rufe zu einem angeschlossenen Telefon durchgeschaltet und bleiben damit etwas länger im System. Das angerufene Endgerät meldet den Verbindungswunsch durch ein Läuten. Die Anrufe werden jedoch nicht vom gerufenen Teilnehmer angenommen. Die emulierten Rufe werden solange gehalten bis der Emulator den Ruf wieder beendet. Der Testaufbau ändert sich, wie in Abbildung 3.6 dargestellt.

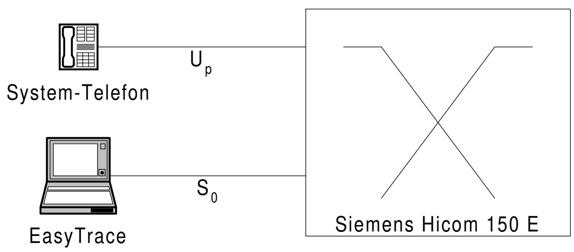


Abbildung 3.6: Versuchsaufbau Szenario 2

In dem beispielhaften Auszug (Tabelle 3.5) aus einem aufgezeichneten Trace sieht man einen Rufaufbau durch ein *setup* mit der Call-Reference 9, bei dem die Ziel- und Ursprungsrufnummer als Parameter mit übermittelt werden. Diesmal wird der Verbindungswunsch vom Netz durch ein *setup_ack* (setup acknowledge) bestätigt. Das darauffolgende *call_proc* (call proceeding) zeigt an, dass die Zielinformationen vollständig sind und dass der Ruf vom Netz bearbeitet wird. Anschließend bekommt der User vom Netz die Nachricht *alert*, die ihm anzeigt,

```
Zeit          U/N C/R SAP  TEI  Layer 2                 Layer 3
--------------------------------------------------------------------------------------------------
10:06:09:860,9  N   R   0   64   NR:020        F:1 RR
10:06:12:392,9  U   C   0   64   NS:020 NR:027 P:0 INFO    O    9 setup bc chi Any cgn 333 cdn 11
10:06:12:399,6  N   R   0   64   NR:021        F:0 RR
10:06:12:430,3  N   C   0   64   NR:021 NS:027 P:0 INFO    D    9 setup_ack chi B2 pri
10:06:12:436,3  U   R   0   64          NR:028 F:0 RR
10:06:12:446,4  N   C   0   64   NR:021 NS:028 P:0 INFO    D    9 call_proc
10:06:12:456,1  U   R   0   64          NR:029 F:0 RR
10:06:12:575,4  N   C   0   64   NR:021 NS:029 P:0 INFO    D    9 alert
10:06:12:581,5  U   R   0   64          NR:030 F:0 RR
10:06:16:512,0  U   C   0   64   NS:021 NR:030 P:0 INFO    O   10 setup bc chi Any cgn 333 cdn 11
10:06:16:519,2  N   R   0   64   NR:022        F:0 RR
10:06:16:549,9  N   C   0   64   NR:022 NS:030 P:0 INFO    D   10 setup_ack chi B1 pri
10:06:16:556,0  U   R   0   64          NR:031 F:0 RR
10:06:16:565,4  N   C   0   64   NR:022 NS:031 P:0 INFO    D   10 call_proc
10:06:16:571,3  U   R   0   64          NR:032 F:0 RR
10:06:16:693,1  N   C   0   64   NR:022 NS:032 P:0 INFO    D   10 prog cau 17 user busy pri
10:06:16:699,1  U   R   0   64          NR:033 F:0 RR
10:06:20:023,9  U   C   0   64   NS:022 NR:033 P:0 INFO    O    9 disc cau 16 normal call clearing
10:06:20:031,4  N   R   0   64   NR:023        F:0 RR
10:06:20:178,9  N   C   0   64   NR:023 NS:033 P:0 INFO    D    9 rel cau 16 normal call clearing fac 0
10:06:20:184,8  U   R   0   64          NR:034 F:0 RR
10:06:20:192,1  U   C   0   64   NS:023 NR:034 P:0 INFO    O    9 rel_com
10:06:20:199,2  N   R   0   64   NR:024        F:0 RR
10:06:20:351,7  U   C   0   64   NS:024 NR:034 P:0 INFO    O   10 disc cau 16 normal call clearing
10:06:20:358,3  N   R   0   64   NR:025        F:0 RR
10:06:20:432,8  N   C   0   64   NR:025 NS:034 P:0 INFO    D   10 rel cau 16 normal call clearing fac 0
10:06:20:438,7  U   R   0   64          NR:035 F:0 RR
10:06:20:445,7  U   C   0   64   NS:025 NR:035 P:0 INFO    O   10 rel_com
10:06:20:453,1  N   R   0   64   NR:026        F:0 RR
```

Tabelle 3.5: Auszug aus dem Trace im Szenario 2

dass das gerufene Endgerät den Ruf bestätigt hat und lokal weitermeldet. Das Telefon zeigt dies durch die Anzeige im Display und durch einen generierten Klingelton an. Währenddessen schickt der Emulator eine weitere Rufeinleitung ab mit der Call-Reference 10. Nach der Bestätigung mit *setup_ack* und der Anzeige der Bearbeitung im Netz mit *call_proc* bekommt das rufende Endsystem aber diesmal die Nachricht, dass das Endsystem auf der anderen Seite besetzt ist. Dieser Fall tritt dann auf, wenn der vorherige Ruf noch nicht abgebaut wurde und bereits ein neuer Rufaufbau stattfindet. Schließlich sieht man, dass der Emulator die Rufe durch *disc* beendet. Daraufhin werden die Rufe wieder abgebaut und die Ressourcen freigegeben.

Um ein Endsystem zu simulieren, bei dem die CPU hoch ausgelastet war, wurde das Programm *CPU-Last* zur Belastung der CPU entwickelt. Dieses Programm startet einen neuen Thread, der eine Endlosschleife ausführt. Diese dient nur dazu, eine vom Scheduler des Betriebssystems zugewiesene Zeitscheibe zu beanspruchen. Dadurch steht den anderen Prozessen weniger Rechenzeit zu, da die gesamte Prozessorzeit auf alle aktiven Prozesse aufgeteilt wird. Die Benutzungsoberfläche, dargestellt in Abbildung 3.7, ermöglicht das Starten und Stoppen des Threads. Zusätzlich kann man noch die Priorität des Threads von sehr niedrig (idle) bis sehr hoch (highest priority) ändern.

Das Tool „Wintop“ von Microsoft wurde auf dem Betriebssystem Windows 98 dazu benutzt, die Verteilung der Rechenzeit auf die laufenden Prozesse und Threads zu beobachten. Demzufolge beanspruchte der Thread bei der höchsten Priorität ca. 95% der gesamten CPU-Zeit. Für den EasyTrace-Prozess verblieben dabei lediglich 5% der Rechenzeit. Dagegen verteilt Windows bei normaler Priorität die gesamte Prozessorzeit gleichmäßig auf die aktiven Prozesse.

Die Experimente wurden auch hier wieder mit unterschiedlichen Rufabständen und Rufdauern durchgeführt. Die Tabelle 3.6 zeigt die Einstellungen der Zeiten. Die Versuche wurden jeweils mit einer niedrig und einer hoch ausgelasteten CPU durchgeführt. Bei dem niedrig ausgelasteten Fall beanspruchte EasyTrace fast die gesamte zur Verfügung stehende Prozessorzeit, da keine anderen

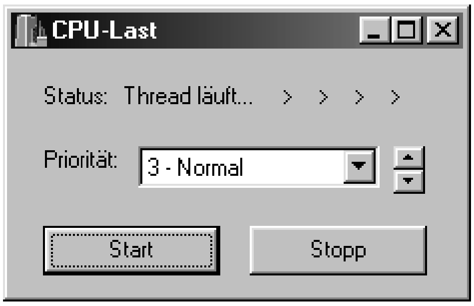


Abbildung 3.7: Screenshot von CPU-Last

Anwendungen aktiv waren. Dagegen wurde bei dem Fall mit hoher Auslastung das Programm CPU-Last mit höchster Priorität gestartet.

Verteilung	Rufabstand	Rufdauer
ohne	6	3
	12	6
negativ-exponentiell	0,1	0,1
	1	1
	6	3
	12	6
	20	20
	30	30
	40	40

Tabelle 3.6: Rufabstände Szenario 2

Die Emulationsszenarien wurden nach Tabelle 3.7 in mehrere Versuchsreihen eingeteilt. Die Versuchsreihen 3 und 4 enthalten nur zwei Emulationen, deren Rufabstände nah beieinander liegen (6 und 12 Sekunden). Die Rufe sind dabei in der dritten Versuchsreihe in festen Abständen und in der vierten negativ-exponentiell verteilt. Eine weitere Testreihe enthält die Vereinigungsmenge der beiden Versuchsreihen. Die fünfte Versuchsreihe enthält dagegen Versuche, deren Rufabstände in einem weitem Bereich auseinander liegen (von 0,1 bis 40 Sekunden), wobei sie wieder einer negativ-exponentiellen Verteilung unterliegen. Durch diese Einteilung der Emulationen in die Versuchsreihen soll der Einfluss der Verteilung und die Streuung der Rufabstände untersucht werden. Die Versuche wurden wieder in eine positive und negative Klasse eingeteilt, wobei die positive Klasse den niedrig ausgelasteten Fall darstellt. Die negative Klasse soll dabei die Versuche widerspiegeln, bei dem das Endsystem künstlich stark belastet wurde. Diese Simulation kann ein in der Praxis vorkommendes System darstellen, das beispielsweise durch Software-Updates neue Funktionen bekommt und dadurch stärker belastet wird. Es soll untersucht werden, ob die Analyse mit der SVM Unterschiede in der Belastung des Rechners erkennen kann.

	Versuchsreihe		
	3	4	5
feste Rufabstände	6 12		
neg.-exp. verteilte Rufabstände		6 12	0,1 1 6 12 20 30 40

Tabelle 3.7: Versuchsreihen im Szenario 2

3.2.3 Szenario 3: Automatische Annahme der Rufe

Bei diesem Szenario soll wieder versucht werden, die Auslastung der CPU eines Endsystems zu erkennen. Es wird also dasselbe Klassifikationskriterium wie in Szenario 2 gewählt, jedoch ändert sich der Versuchsaufbau wie folgt: EasyTrace emulierte zusätzlich ein Endgerät, das die Rufe automatisch annimmt. Dafür wurde der Emulator mit zwei Leitungen an den S_0-Bus angeschlossen (siehe Abbildung 3.8). In EasyTrace wurden dann zwei Emulationen gestartet. Die erste Emulation generiert abgehende Rufe mit einem mittleren Rufabstand von 12 Sekunden bei negativ-exponentieller Verteilung. Das gerufene Endsystem wurde ebenfalls von EasyTrace emuliert, wobei die ankommenden Rufe nach einer Sekunde angenommen und gehalten wurden. Das rufende Endsystem beendet den Ruf mit einer mittleren Rufdauer von 6 Sekunden (bei negativ-exponentieller Verteilung). EasyTrace emuliert gleichzeitig den Rufenden und den Gerufenen, wobei beide Endsysteme dieselbe Mehrfachrufnummer hatten.

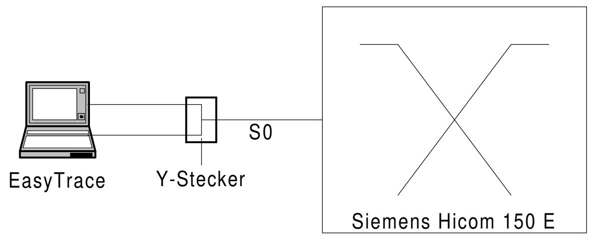


Abbildung 3.8: Versuchsaufbau Szenario 3

Die abgehenden Rufe des Emulators wurden von der Telefonanlage weitervermittelt. Die Anlage erzeugt dabei ihrerseits wieder Rufeinleitungen an demselben D-Kanal, die von dem gerufenen Endsystem angenommen wurden. Dadurch entstehen zwei logische Verbindungen, die anhand der TEI auseinander gehalten werden können. Bei der Analyse macht es jedoch nur Sinn, ausschließlich *eine* logische Verbindung zu betrachten. Daher wird durch eine Filterung nach der TEI jede logische Verbindung isoliert betrachtet. Bei der Filterung muss die TEI 127 mit berücksichtigt werden, da Broadcast-Nachrichten zu jeder logischen

Verbindung gehören.

Es entstanden hierbei zwei einzelne Versuchsreihen. Die Emulationen mit den abgehenden Rufe wurden als Versuchsreihe 6 und die Aufzeichnung der Annahme der Rufe als Versuchsreihe 7 bezeichnet. Zusätzlich wurde die Vereinigungsmenge als Testreihe hinzugenommen.

```
Zeit          U/N C/R SAP  TEI  Layer 2                 Layer 3
------------------------------------------------------------------------------------------------------
12:14:55:681,0  U   C   0    71  NS:000 NR:000 P:0 INFO    O    1 setup bc chi Any cgn 19 cdn 19
12:14:55:687,7  N   R   0    71  NR:001        F:0 RR
12:14:55:714,6  N   C   0    71  NR:001 NS:000 P:0 INFO    D    1 setup_ack chi B2 pri
12:14:55:822,0  N   C   0   127                P:0 UI      O    2 setup bc chi B1 cgn 19 cdn 19
12:14:56:161,0  U   R   0    71           NR:001 F:0 RR
12:14:56:170,5  N   C   0    71  NR:001 NS:001 P:0 INFO    D    1 call_proc
12:14:56:210,8  U   R   0    71         NR:002 F:0 RR
12:14:56:220,3  U   C   0    71  NS:001 NR:002 P:0 INFO    D    2 rel_com cau 16 normal call clearing <?>
12:14:56:226,9  N   R   0    71  NR:002        F:0 RR
12:14:57:261,6  U   C   0    70         NR:000 P:1 RR
12:14:57:267,7  N   R   0    70  NR:000        F:1 RR
12:14:57:281,4  U   C   0    70  NS:000 NR:000 P:0 INFO    D    2 alert chi B1
12:14:57:288,0  N   R   0    70  NR:001        F:0 RR
12:14:57:333,1  N   C   0    71  NR:002 NS:002 P:0 INFO    D    1 alert
12:14:57:339,6  U   R   0    71         NR:003 F:0 RR
12:14:58:250,3  U   C   0    70  NS:001 NR:000 P:0 INFO    D    2 conn
12:14:58:256,9  N   R   0    70  NR:002        F:0 RR
12:14:58:280,7  N   C   0    70  NR:002 NS:000 P:0 INFO    O    2 conn_ack chi B1
12:14:58:288,3  U   R   0    70         NR:001 F:0 RR
12:14:58:341,1  N   C   0    71  NR:002 NS:003 P:0 INFO    D    1 conn dte con 19
12:14:58:346,8  U   R   0    71         NR:004 F:0 RR
12:14:58:353,7  U   C   0    71  NS:002 NR:004 P:0 INFO    O    1 conn_ack
12:14:58:360,3  N   R   0    71  NR:003        F:0 RR
12:15:08:247,1  U   C   0    70         NR:001 P:1 RR
12:15:08:253,2  N   R   0    70  NR:002        F:1 RR
12:15:08:356,4  U   C   0    71         NR:004 P:1 RR
12:15:08:362,4  N   R   0    71  NR:003        F:1 RR
12:15:08:809,7  U   C   0    71  NS:003 NR:004 P:0 INFO    O    2 setup bc chi Any cgn 19 cdn 19
12:15:08:817,1  N   R   0    71  NR:004        F:0 RR
12:15:08:832,3  N   C   0    71  NR:004 NS:004 P:0 INFO    D    2 rel_com cau 34 no circuit/channel available
12:15:08:838,6  U   R   0    71         NR:005 F:0 RR
12:15:16:601,3  U   C   0    71  NS:004 NR:005 P:0 INFO    O    1 disc cau 16 normal call clearing
12:15:16:607,9  N   R   0    71  NR:005        F:0 RR
12:15:16:684,4  N   C   0    70  NR:002 NS:001 P:0 INFO    O    2 disc cau 16 normal call clearing fac 0 pri
12:15:16:725,2  N   C   0    71  NR:005 NS:005 P:0 INFO    D    1 rel cau 16 normal call clearing fac 0
12:15:17:280,8  U   R   0    70         NR:002 F:0 RR
12:15:17:307,0  U   C   0    71  NS:005 NR:006 P:0 INFO    O    1 rel_com
12:15:17:313,6  N   R   0    71  NR:006        F:0 RR
12:15:18:356,0  U   C   0    70         NR:002 P:1 RR
12:15:18:362,1  N   R   0    70  NR:002        F:1 RR
12:15:18:370,0  U   C   0    70  NS:002 NR:002 P:0 INFO    D    2 rel
12:15:18:377,3  N   R   0    70  NR:003        F:0 RR
12:15:18:392,2  N   C   0    70  NR:003 NS:002 P:0 INFO    O    2 rel_com
```

Tabelle 3.8: Auszug aus dem Trace im Szenario 3

Die Tabelle 3.8 zeigt einen typischen Ausschnitt aus den aufgezeichneten Daten. Hierbei hat das rufende Endsystem die TEI 71 und das gerufene die TEI 70. Das rufende Endsystem leitet einen Ruf mit *setup* mit der Call Reference 1 ein, der von der Telefonanlage bestätigt und bearbeitet wird. Die Anlage schickt ihrerseits eine Rufeinleitung mit Call Reference 2 als Broadcast (TEI 127) auf denselben Bus, worauf sich der gerufene Teilnehmer mit *alert* meldet. Die gerufene Seite stellt dann die Verbindung mit *conn* her, wodurch die Anlage wiederum eine Verbindung mit dem rufenden System herstellt. Während der Verbindung kommt es vor, dass der Rufende einen weiteren Verbindungsaufbau einleitet. Dieser wird jedoch vom Netz zurückgewiesen, da kein Nutzkanal mehr zur Verfügung steht. Schließlich leitet der Rufende den Verbindungsabbau mit *disc* ein und veranlasst dadurch auch den Abbau der Verbindung auf der gerufenen Seite. Die benutzten Ressourcen werden dann wieder freigegeben.

Das Emulationsszenario wurde zunächst auf einem nicht ausgelasteten System durchgeführt. Bei den nachfolgenden Emulationen wurde die Prozessorbelastung des emulierenden Systems durch Erhöhung der Prozess-Priorität der Belastungssoftware CPU-Last allmählich gesteigert. Dadurch entstanden vier Testreihen, die in zwei Klassen eingeteilt wurden. Die Tabelle 3.9 zeigt die Zuordnung in Abhängigkeit vom Anteil der Rechenzeit des Emulators (gemessen mit Wintop). Die positive Klasse beinhaltet dabei die Emulationen, bei denen der Lastgenerator schwach belastet war, wobei die negative Klasse die Versuche mit hoher Auslastung des Endsystems darstellt.

Klasse	CPU-Anteil
positiv	$\geq 99\%$
	$\approx 93\%$
negativ	$\approx 49\%$
	$\leq 6\%$

Tabelle 3.9: Einteilung der Klassen im Szenario 3

3.2.4 Szenario 4: Erkennung der Anlage

Bei dem letzten Szenario wurden dieselben Versuchsreihen wie in Szenario 2 durchgeführt, es wurde jedoch anstatt der Hicom von Siemens die Telefonanlage AGFEO AC 141 benutzt. Der Versuchsaufbau und der Ablauf des Szenarios bleiben gleich. Der mittlere Rufabstand beträgt 12 Sekunden und die mittlere Rufdauer 6 Sekunden. Diese Zeiten sind wie im Szenario 2 in festen und in negativ-exponentiell verteilten Rufabständen. Der einzige Unterschied zu Szenario 2 ist also die vermittelnde Telefonanlage. Der Lernalgorithmus soll versuchen einen Unterschied im Verhalten des Protokollablaufs der Anlage von AGFEO zur Anlage von Siemens festzustellen, falls es überhaupt einen Unterschied gibt. Die Telefonanlagen basieren zwar auf demselben Protokoll (Q.921/Q.931), jedoch können zwischen den Anlagen herstellerabhängige Eigenheiten auftreten. Die Normen und Spezifikationen legen viele technische Details im Protokollablauf fest, jedoch gibt es einen Spielraum innerhalb der Spezifikationen, die ein unterschiedliches Verhalten von Hersteller zu Hersteller erlauben.

Die Tabelle 3.10 zeigt einen beispielhaften Ausschnitt aus den Tracedaten der Emulation mit der Anlage von AGFEO. Vergleicht man diese Daten mit der Emulation der Hicom von Siemens, so fällt beim Rufaufbau auf, dass hier keine *call_proc*-Meldungen verschickt werden. Das Senden von Call-Proceeding bestätigt die Vollständigkeit der Zielinformationen und die Verfügbarkeit des angeforderten Dienstes. Nach dem Zustandsübergangsdiagramm in der Richtlinie Q.931 ist es auch möglich, nach einem Setup-Acknowledge direkt ein Alerting oder Connect zu senden. Die Anlage von AGFEO implementiert hier die Variante, auf ein *setup_ack* ein *alert* zu senden, wogegen die Hicom zwischen diesen beiden Nachrichten ein *call_proc* sendet. Es gibt zwar einen Unterschied bei den Tracedaten, aber beide Anlagen verhalten sich trotzdem normgerecht. Die Anlagen durchlaufen sogar andere Zustände im Zustandsdiagramm, landen aber nach erfolgreichem Rufaufbau in demselben Zustand. Beim Rufabbau sind dagegen keine Unterschiede im Ablauf der Kommunikation zu beobachten. Es kann jedoch Abweichungen bei den Zeitdifferenzen zwischen den Nachrichten geben.

```
Zeit           U/N C/R SAP  TEI  Layer 2                 Layer 3
------------------------------------------------------------------------------------------------------------
08:47:01:810,5  U   R   0   64          NR:000 F:1 RR
08:47:09:600,4  U   C   0   64   NS:000 NR:000 P:0 INFO    O    1 setup bc chi Any cgn 31 cdn 11
08:47:09:608,3  N   R   0   64   NR:001        F:0 RR
08:47:09:643,2  N   C   0   64   NR:001 NS:000 P:0 INFO    D    1 setup_ack chi B1 pri
08:47:09:649,1  U   R   0   64          NR:001 F:0 RR
08:47:09:727,2  N   C   0   64   NR:001 NS:001 P:0 INFO    D    1 alert chi B1 pri
08:47:09:732,9  U   R   0   64          NR:002 F:0 RR
08:47:15:651,4  U   C   0   64   NS:001 NR:002 P:0 INFO    O    1 disc cau 16 normal call clearing
08:47:15:659,3  N   R   0   64   NR:002        F:0 RR
08:47:15:692,7  N   C   0   64   NR:002 NS:002 P:0 INFO    D    1 rel
08:47:15:698,9  U   R   0   64          NR:003 F:0 RR
08:47:15:706,1  U   C   0   64   NS:002 NR:003 P:0 INFO    O    1 rel_com
08:47:15:715,2  N   R   0   64   NR:003        F:0 RR
08:47:21:208,8  U   C   0   64   NS:003 NR:003 P:0 INFO    O    2 setup bc chi Any cgn 31 cdn 11
08:47:21:217,0  N   R   0   64   NR:004        F:0 RR
08:47:21:251,4  N   C   0   64   NR:004 NS:003 P:0 INFO    D    2 setup_ack chi B1 pri
08:47:21:257,4  U   R   0   64          NR:004 F:0 RR
08:47:21:334,0  N   C   0   64   NR:004 NS:004 P:0 INFO    D    2 alert chi B1 pri
08:47:21:345,4  U   R   0   64          NR:005 F:0 RR
08:47:27:296,2  U   C   0   64   NS:004 NR:005 P:0 INFO    O    2 disc cau 16 normal call clearing
08:47:27:304,0  N   R   0   64   NR:005        F:0 RR
08:47:27:341,3  N   C   0   64   NR:005 NS:005 P:0 INFO    D    2 rel
08:47:27:347,3  U   R   0   64          NR:006 F:0 RR
08:47:27:360,2  U   C   0   64   NS:005 NR:006 P:0 INFO    O    2 rel_com
08:47:27:369,2  N   R   0   64   NR:006        F:0 RR
```

Tabelle 3.10: Auszug aus dem Trace im Szenario 4

3.3 Auswahl und Repräsentation der Daten

Um auf den aufgezeichneten Daten lernen zu können, müssen aus den Daten geeignete Merkmale herausgefiltert werden. Dazu soll zunächst das schichtenorientierte Kommunikationsmodell betrachtet werden (siehe Abbildung 3.9). Die

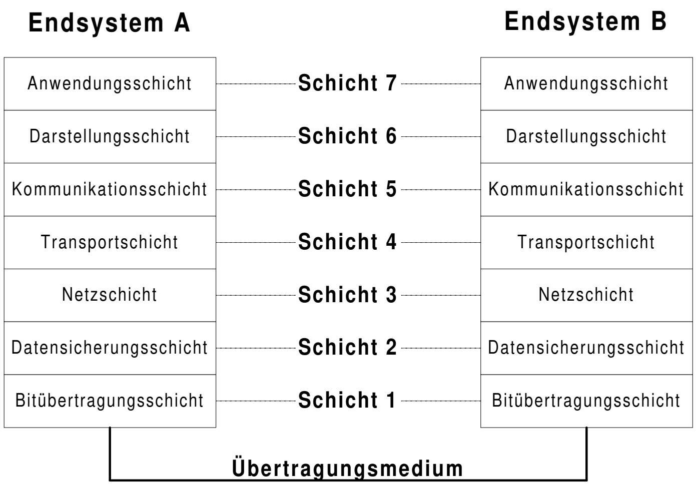


Abbildung 3.9: Schichtenarchitektur

zu sendenden Daten der obersten Kommunikationsschicht werden an die darunter liegende Schicht weitergereicht bis die Bitübertragungsschicht erreicht wird. Auf der Bitübertragungsschicht werden die Daten letztlich über die Leitung übertragen. Auf der Seite des Kommunikationspartners durchlaufen die Daten der Schicht 1 den Protokollstack in umgekehrter Reihenfolge. Die Daten werden von unteren Schichten an die darüber liegenden weitergereicht. Jede Schicht ist dabei für eine bestimmte Aufgabe zuständig und stellt eine Dienstleistung für die darüber liegenden Schichten zur Verfügung. So verlässt sich beispielsweise die Schicht 3 auf die sichere Übertragung der Daten von der Sicherungsschicht.

Die geforderte Lernaufgabe ist die Klassifizierung der Beispiele nach den folgenden Kriterien:

- Auslastung des D-Kanals,
- Prozessorauslastung des Kommunikationssystems,
- herstellerabhängige Implementierung.

Für die Signalisierung im ISDN werden nur die ersten drei Protokollschichten verwendet. Bei der Diagnostik von Telekommunikationsanlagen stellt sich nun die Frage, auf welcher Schicht im Protokollstack die Analyse anzusetzen ist.

Die Schicht 1 ist für die Umsetzung von Datenbits in elektrische Signale zuständig. Diese Signale werden letztlich analog übertragen und können mit einem Oszilloskop aufgezeichnet werden. Die Analyse dieser Signale wäre aber viel zu aufwändig, da man zu viele Informationen daraus nicht benötigt. Die Schicht 1 enthält zwar alle Informationen, die über die physikalische Leitung übertragen werden, jedoch werden längst nicht alle Informationen benötigt. Hinzu kommt, dass die Informationen mit einem Leitungscode codiert wurden. Um an die Daten der Schicht 2 und 3 zu kommen, müsste diese Codierung wieder dekodiert werden, was den Aufwand zusätzlich erheblich erhöhen würde. Die Bitübertragungsschicht ist also für die Diagnostik ungeeignet.

Sämtliche Informationen, die man für die Analyse der Daten benötigt, befinden sich auch in der Sicherungsschicht des Protokollstacks. Diese Schicht verlässt sich auf eine ungesicherte Verbindung auf physikalischer Ebene. Die Schicht 2 sorgt für eine gesicherte Verbindung, indem sie eine Fehlerkorrektur und einen Quittierungsmechanismus hinzufügt. Die darüber liegenden Schichten verlassen sich dann wiederum auf die gesicherte Verbindung der Schicht 2. Als Messgerät lässt sich ein Protokollanalysator einsetzen, der alle Daten auf Schicht 2 anzeigt. Auf Schicht 1 werden dagegen nur Zustandsinformationen angezeigt.

Bei den aufgezeichneten Testszenarien wurden funktionsfähige und ausgereifte Kommunikationsgeräte benutzt. Auch die Verkabelung entspricht den Anforderungen für eine stabile Kommunikation. Daher ist davon auszugehen, dass „grobe“ Fehler während des Informationsaustausches nicht auftreten. Ein grober Fehler wäre z.B. ein fehlerhaft übertragener Rahmen, der nach Ausbleiben der Quittung und Ablauf des Times neu übertragen wird. Bei den aufgezeichneten Szenarien sind jedoch keine Fehler aufgetreten. In den Tracedaten sind ausschließlich korrekte Kommunikationsabläufe zu beobachten.

Beim Übergang der Analyse zu Schicht 3 können einige Informationen verloren gehen. Die Netzwerkschicht im D-Kanal enthält die Signalisierungsinformationen für die Kommunikation. Basiert die Analyse ausschließlich auf den Informationen dieser Schicht, so kann nicht das Quittierungsverhalten auf Schicht 2 beobachtet werden. Daher ist es sinnvoll die Analyse auf der Sicherungsschicht anzusetzen. Die Erkenntnisse der Schicht-2-Analyse können dann auch auf Schicht 3 übertragen werden, um eine weitergehende Diagnose zu betreiben. Diese Diplomarbeit beschränkt sich jedoch auf die Analyse in Schicht 2. Die Übertragung der Erkenntnisse auf andere Schichten erfordert weitere Forschungsarbeit.

LAP-D ist das Protokoll in Schicht 2 im ISDN. Hier gibt es folgende Informationen, die auch mit dem Protokololanalysator EasyTrace gemessen werden können (vgl. Abbildung 3.1):

- SAPI,
- Command/Response-Bit,
- TEI,
- Rahmentyp,
- evtl. eingebettete Schicht-3-Daten,
- absoluter Zeitpunkt des letzten Bytes des Rahmens,
- Richtung.

Daraus müssen die für die Diagnose relevanten Daten ausgewählt werden. Da hier nur die Signalisierung auf Schicht 2 beobachtet werden soll, kommen nur die Rahmen mit SAPI 0 in Frage. Es hat sich außerdem gezeigt, dass die INFO- und RR-Rahmen für die Analyse in Schicht 2 betrachtet werden müssen (vgl. [2]). Die Aufgabe der Datensicherungsschicht ist es gerade, die Schicht-3-Daten sicher zu übertragen. Es soll der Ablauf des Informationsaustausches beobachtet werden, daher werden nur die Rahmen betrachtet, die daran beteiligt sind. Die Daten werden im Informationsfeld eines INFO-Rahmens übertragen und die Quittierungen darauf in RR-Rahmen. Betrachtet man den Rahmentyp in den Aufzeichnungen, so fällt auf, dass während der Emulationen auch nur RR- und INFO-Rahmen gesendet wurden. Andere Rahmentypen kommen nur für den Auf- und Abbau der Schicht 2 vor. Diese Rahmen treten bei den Emulationen am Anfang bzw. am Ende einer Messung auf. Die Rahmen für Auf- und Abbau der Schicht 2 und für die TEI-Vergabeprozedur werden ignoriert. Daher wurden Messdaten, die vor Beginn bzw. nach Beendigung des Test-Szenarios aufgenommen wurden, entfernt. Zu beachten ist, dass INFO-Rahmen ausschließlich als Kommando (Command) auftreten. Sonstige Rahmentypen (RNR-, REJ-, FRMR-, XID-Rahmen) sind bei den Messungen nie aufgetreten.

Da außerdem nur *eine* logische Verbindung analysiert werden soll, muss eine Kommunikationsverbindung anhand der TEI herausgefiltert werden. Die eingebetteten Schicht-3-Daten werden in der Diagnostik nicht näher analysiert. Nach der Filterung der RR- und INFO-Rahmen nach SAPI und TEI bleiben folgende relevanten Informationen übrig: Rahmentyp, Command/Response-Bit, Richtung und Zeitpunkt. Damit lässt sich die Überwachungs- und Steuerungsfunktion einer Kommunikation auf Schicht 2 analysieren.

Ziel der Untersuchung ist es, mit Hilfe dieser Daten, Muster im Protokollverhalten zu erkennen und zu lernen. Um ein Verhalten erkennen zu können, werden aufeinander folgende Rahmen beobachtet. Die naheliegende Vermutung, dass die Lernergebnisse der Testszenarien unabhängig von Datum und Uhrzeit sind, führt dazu, dass nur noch die Zeitdifferenz zwischen unmittelbar nacheinander gesendeten Rahmen notwendig ist. Das Protokollverhalten kann anhand der Zeitdifferenz zweier Rahmen beobachtet werden. Man ist nun daran interessiert, wie sich die Zeitdifferenz zwischen allen möglichen Rahmenpaaren verhält. Dafür werden zunächst die relevanten Kennzeichen der Rahmen notiert und die

Zeitdifferenz dazwischen untersucht. Dies soll Aufschluss darüber geben, wie sich die Reaktionszeiten des Systems verhalten.

Die Kennzeichen der relevanten Rahmen sind Rahmentyp, C/R-Bit und die Richtung der Übertragung. Jedes dieser Kennzeichen kann dabei zwei Werte annehmen. Die beiden Rahmentypen (INFO/RR) können jeweils von den beiden ausgezeichneten Seiten einer Verbindung, der Netz- oder der Teilnehmerseite (Network/User), übertragen werden. Das Response-Bit kann dabei nur in RR-Rahmen vorkommen. Die relevanten Schicht-2-Rahmen können daher in folgenden sechs Kombinationen auftreten:

- INFO Command Network (INFO C/N)
- INFO Command User (INFO C/U)
- RR Command Network (RR C/N)
- RR Command User (RR C/U)
- RR Response Network (RR R/N)
- RR Response User (RR R/U)

Werden nun zwei aufeinander folgende Rahmen in Beziehung gesetzt, so erhält man $6 \cdot 6 = 36$ Kombinationsmöglichkeiten von Nachrichtenpaaren. Misst man die Zeitdifferenz zwischen den Nachrichtenpaaren, so erhält man einen dreidimensionalen Datenraum, in dem die Beziehungen beobachtet werden können. Die ersten beiden Dimensionen beinhalten die Informationen über den Rahmentyp und die dritte Dimension stellt die Zeitdifferenz zwischen den beiden Rahmen dar. Bei korrektem Protokollablauf dürfen keine Zeiten auftreten, die größer als 10s sind, da nach 10s Inaktivität das „Keep-alive-Polling“ gestartet wird. Durch ungenaue Timer kann diese Zeitspanne leicht überschritten werden. Wesentliche Überschreitungen dieses Timers deuten jedoch auf einen Fehler im Protokollablauf hin. Bei dieser Repräsentation sind in den ersten beiden Dimensionen nur jeweils sechs diskrete Werte möglich, daher erhält man eigentlich 36 eindimensionale Räume.

Prinzipiell ist auch die Abbildung der Kennzeichen eines Rahmens im dreidimensionalen Raum möglich. Werden dann zwei Rahmen in Beziehung gesetzt und die Zeit dazwischen ermittelt, so erhält man einen sieben-dimensionalen Raum. Für den Lernalgorithmus stellt dies kein Problem dar, jedoch versagt das menschliche Darstellungsvermögen in dem Raum. Die Repräsentation der Daten im dreidimensionalen Raum kann dafür grafisch dargestellt werden und für einen Beobachter gut verglichen werden. Auch die dreidimensionale Ansicht von Clustern aus dem unüberwachten Lernverfahren ist dann möglich. Allerdings erschwert man sich die Analyse der Kennzeichen der Rahmen. Es wird schwieriger zu erkennen, ob eine Eigenschaft unabhängig ist. Da es wichtig ist, die Ergebnisse des Lernalgorithmus zu verstehen und zu hinterfragen, muss die Repräsentation der Daten nachvollziehbar sein. Daher wurde entschieden, die Kennzeichen der Rahmen im eindimensionalen Raum abzubilden.

Damit numerische Lösungsverfahren für die Analyse angewendet werden können, ist es sinnvoll diese Nachrichtenarten in einen numerischen Bereich zu transformieren. Dafür wird eine Zuordnungsvorschrift nach Tabelle 3.11 definiert, die Rahmen auf ganze Zahlen abbildet (vgl. [2]). Der Abstand be-

Rahmentyp	Command/Response	Sender	Wert
RR	Response	Network	-25
RR	Command	Network	-15
INFO	Command	Network	-5
INFO	Command	User	5
RR	Command	User	15
RR	Response	User	25

Tabelle 3.11: Abbildung des Rahmentyps auf numerischen Bereich

nachbarter Nachrichtenpaare wurde absichtlich auf zehn gesetzt, damit bei der Fuzzy-Clusteranalyse Punkte aus benachbarten Zeitachsen nicht zu einem Cluster zusammengefasst werden. Einem Cluster sollen ausschließlich Punkte angehören, die dieselben Rahmenpaare haben. Diese Punkte können höchstens den Abstand von zehn haben. Die ersten beiden Dimensionen wurden nun so skaliert, dass die Punkte aus benachbarten Zeitachsen weit genug auseinander liegen. Ist der Abstand zu Punkten anderer Zeitachsen mindestens zehn, so wird der FCM-Algorithmus keine Cluster finden, denen Punkte unterschiedlicher Nachrichtenpaare angehören.

Es werden nun die beiden Rahmentypen mit Angabe über die Richtung (User/Network) und des C/R-Flags sowie die Zeitdifferenz zwischen diesen Rahmen notiert. Daraus ergibt sich das Tripel

$$\vec{p} = \begin{pmatrix} x \\ y \\ t \end{pmatrix} \quad \text{mit } x, y \in \{-25, -15, -5, 5, 15, 25\}, \quad t \in \mathbb{R}_0^+. \tag{3.4}$$

x und y stellen dabei die oben beschriebenen Rahmen dar, wobei der Rahmen y direkt nach Rahmen x aufgezeichnet wurde. Die formale Definition der Punkte zeigt, dass die beiden Komponenten x, y des Vektors $\vec{p}$ nur sechs diskrete Werte annehmen dürfen. Daraus folgt, dass es insgesamt $6 \cdot 6 = 36$ Kombinationen aus x- und y-Werten gibt. Die Projektion beliebiger Punkte auf die x-y-Ebene ergibt demnach nur 36 verschiedene Positionen. Man kann sich den Raum nun so vorstellen, dass auf der x-y-Ebene 36 Zeitachsen stehen.

Die Menge $P = \{\vec{p_1}, \vec{p_2}, \ldots, \vec{p_n}\}$ enthält alle Punkte und stellt die Menge der Ausgangsdaten für die weitere Vorverarbeitung dar.

3.4 Vorverarbeitung der Daten

Die Menge der Ausgangsdaten muss nun für den Lernalgorithmus geeignet verarbeitet werden, so dass ein möglichst hoher Lernerfolg bei effizienter Laufzeit erzielt wird. Im folgenden werden mehrere Methoden vorgestellt, wie die Daten für den Lernalgorithmus vorverarbeitet werden können. Diese Vorverarbeitungsmethoden werden auf den Testdaten der emulierten Szenarien untersucht. Die Ergebnisse der Experimente werden in Kapitel 5 vorgestellt.

3.4.1 Ohne Vorverarbeitung

Als Ausgangspunkt soll ein Verfahren dienen, bei dem keine Vorverarbeitung der Daten erfolgt. Die danach vorgestellten Methoden werden dann mit diesem Verfahren verglichen und sollen eine Verbesserung zeigen.

Die Menge der Ausgangsdaten einer Aufzeichnung besteht nach dem Einleseprozess aus den dreidimensionalen Punkten

$$\vec{p}_x = \begin{pmatrix} p_{x,1} \\ p_{x,2} \\ p_{x,3} \end{pmatrix}, \quad x \in \{1, \ldots, n\}. \tag{3.5}$$

Diese Vektoren dienen ohne weitere Verarbeitung als Beispiele für die SVM. Ein Beispiel enthält die Koordinaten der Punkte, deren Werte in einem Vektor aneinander gehängt werden:

$$\vec{x}_i = (p_{1,1}, p_{1,2}, p_{1,3}, p_{2,1}, p_{2,2}, p_{2,3}, \ldots, p_{n,1}, p_{n,2}, p_{n,3})^T. \tag{3.6}$$

Der Beispiel-Vektor hat eine Dimension von $3n$, die proportional zur Anzahl der Eingabevektoren ist und daher unterschiedlich sein kann. Jedes dieser Beispiele wird mit einer Markierung versehen, die die Werte -1 oder 1 annehmen kann:

$$y_i \in \{-1, 1\}. \tag{3.7}$$

Die Trainingsdaten bestehen aus Tupel aus einem Muster $\vec{x}_i$ und einer Markierung $y_i \in \{-1, 1\}$:

$$(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n). \tag{3.8}$$

Bei dieser Methode scheint es fragwürdig, ob die Lernaufgabe überhaupt gelöst werden kann. Der Lernalgorithmus versucht eine Regel für die Erkennung unbekannter Beispiele zu finden. Die Vektoren werden in chronologischer Reihenfolge aneinander gehängt. Durch eine kleine Verschiebung der Zeitpunkte für den Anfang eines Beispiels können zwei identische Testdatenreihen unterschiedliche Beispiel-Vektoren ergeben.

3.4.2 Absolute Häufigkeit in Intervallen

Durch eine geeignete Vorverarbeitung der Daten wird eine Verbesserung der Ergebnisse erwartet. Dabei sollen nicht, wie in Abschnitt 3.4.1, alle Punkte nacheinander betrachtet werden. Die Grundidee ist, die Verteilung der Punkte in dem quaderförmigen Raum möglichst exakt abzubilden. Der dreidimensionale Raum wird in Teilräume unterteilt, in denen die Häufigkeit von aufgetretenen Punkten gemessen wird. Der Raum kann dabei nur auf den 36 Zeitachsen von Punkten besetzt sein. Die Zeitachsen des Raumes werden jeweils entlang eines festgelegten Weges in j gleichgroße Intervalle der Größe $t_g = 10,5/j$ unterteilt. Bei korrektem Verhalten der Kommunikationsgeräte dürfen keine Zeiten größer als 10 Sekunden auftreten. Durch ungenaue Timer oder Überlastsituationen kann jedoch diese Zeitdifferenz leicht überschritten werden, daher wurde der Raum in Richtung der Zeitachse auf 10,5 Sekunden beschränkt, um leichte Überschreitungen zu tolerieren.

Man erhält nach der Einteilung des Raumes $36j$ Intervalle gleicher Größe. In jedem Intervall wird dann die Anzahl der darin vorkommenden Punkte bestimmt. Die absoluten Häufigkeiten der Punkte in den Intervallen werden in der

Reihenfolge des Durchlaufs durch die Zeitachsen notiert und bilden ein Beispiel für die Lernmaschine[7].

Ein zunächst einfacher aber ineffizienter Algorithmus für das Problem sieht wie folgt aus: Für jedes Intervall wird jeweils die Menge der Punkte durchlaufen und dabei die Summe der in den Intervallen liegenden Punkte bestimmt. Mit jedem Punkt wird also ein Vergleich für jedes Intervall notwendig, mit dem überprüft wird, ob ein Punkt in dem jeweiligen Intervall liegt. Es sind somit $36jn$ Vergleiche erforderlich. Da jeder Punkt in genau einem Intervall liegt, wird zusätzlich eine Addition für jeden Punkt notwendig. Als gesamte Rechenzeit lässt sich daher abschätzen: $36jn + n$. Die Laufzeit ist daher in der Größenordnung von $O(nj)$. Die Ausführungsdauer des Algorithmus steigt also linear mit dem Produkt aus der Anzahl der Intervalle und der Anzahl der Punkte.

Ist beispielsweise ein Intervall 10 ms groß und enthält die Trainingsmenge 1000 Beispiele, so beträgt die Anzahl der Operationen (Vergleiche und Additionen) $36 \cdot 10500 \cdot 1000 + 1000 \approx 3,78 \cdot 10^8$. Der Aufwand für die Berechnung der Häufigkeiten dieses durchaus realistischen Beispiels ist enorm hoch und für noch größere Beispiele nicht mehr praktikabel. Daher ist eine effizientere Lösung wünschenswert.

Der Ansatz für die Entwicklung eines schnelleren Algorithmus beruht auf der Ermittlung eines eindeutigen Schlüssels für jedes Intervall in dem Raum. Aus den Koordinaten der Intervalle kann mit Hilfe einer entsprechenden Funktion ein Wert errechnet werden, der einem Intervall eindeutig zugeordnet werden kann. Dabei ist die Rundung des Zeitwertes auf die Auflösung der Intervallgröße zu beachten. Dementsprechend lässt sich auch zu jedem Punkt (ebenfalls durch entsprechende Rundung des Zeitwertes) ein Schlüssel errechnen. Somit kann anhand dieses Schlüssels jeder Punkt eindeutig einem Intervall zugeordnet werden. Ein effizienter Algorithmus kann dies ausnutzen, indem er für alle Punkte diesen Schlüssel berechnet und die Anzahl jedes aufgetretenen Wertes in einer entsprechenden Datenstruktur ablegt. Anschließend kann für jedes Intervall ebenfalls dieser Schlüssel berechnet werden, mit dem in der Datenstruktur die Anzahl der gefundenen Schlüssel gesucht werden kann. Benutzt man als Datenstruktur einen ausgeglichenen Binärbaum, so werden die häufig benötigten Einfüge- und Suchoperationen effizient in $O(\log n)$ unterstützt.

Der in dieser Arbeit benutzte Algorithmus läuft wie folgt ab: Für jeden Punkt wird der Schlüssel berechnet, mit dem im Binärbaum eine Suche durchgeführt wird. Wird der Schlüssel gefunden, so wird auf den Wert eine eins aufaddiert, anderenfalls wird der Schlüssel mit dem Wert eins in den Baum eingefügt. Dieser Vorgang hat die Komplexität $O(n \cdot \log n)$. Danach muss für jedes Intervall ebenfalls der Schlüssel berechnet und damit eine Suche im Baum gestartet werden. Die Laufzeit dieser Schleife ist $O(j \cdot \log n)$, so dass sich für die Komplexität des gesamten Algorithmus $O((j + n) \log n)$ ergibt. Die Anzahl der Operationen für das obige Beispiel reduziert sich damit auf die Größenordnung von weniger als 10^6. Praktische Versuche mit beiden Algorithmen haben gezeigt, dass der effiziente Algorithmus tatsächlich eine deutlich kürzere Ausführungszeit hat.

[7] Dieses Verfahren hat eine gewisse Analogie zu dem Verfahren der Textkategorisierung von Thorsten Joachims, bei dem die Häufigkeiten von Wortsilben in einem Text bestimmt werden (siehe [20]).

3.4.3 Relative Häufigkeit in Intervallen

Eine Variation der vorhergehenden Methode zur Vorverarbeitung der Daten wird in diesem Kapitel vorgestellt. Es wird auch hier wieder versucht, den Raum durch Intervalleinteilungen abzubilden. Jedoch wird nur die Häufigkeit in Bezug zur gesamten Anzahl von Punkten betrachtet. In den Intervallen wird somit die relative Häufigkeit von Punkten anstatt der absoluten Häufigkeit bestimmt. Dadurch wird nur die Verteilung der Punkte abgebildet, unabhängig von der Anzahl der Punkte eines Beispiels.

Der Raum wird wieder, wie oben beschrieben, in gleichgroße Intervalle unterteilt, in denen dann die relative Häufigkeit von Punkten bestimmt wird, indem die absolute Häufigkeit durch die Summe aller absoluten Häufigkeiten dividiert wird. Diese Zusatzberechnung macht sich in der Ausführungszeit für den Benutzer nicht bemerkbar und ist daher nicht weiter untersucht worden.

3.4.4 Koordinaten der Cluster

Bei diesem Verfahren werden die Ergebnisse einer vorangegangenen Fuzzy-Clusteranalyse für die Vorverarbeitung der Daten benutzt. Dazu wird zunächst eine vollständige Fuzzy-Clusteranalyse für jedes Zeitfenster durchgeführt nach dem Verfahren aus Abbildung 2.11 im Kapitel 2.6.3. Da die optimale Clusteranzahl bei jedem Zeitfenster unterschiedlich sein kann, wird in jedem Zeitfenster zuerst die optimale Anzahl von Clustern bestimmt. Die unteren Schranken der Clusteranzahlen in den Zeitfenstern werden direkt beim Einlesen der Tracedaten ermittelt, indem die Anzahl unterschiedlicher Nachrichtenpaare gezählt wird. Die maximale Clusteranzahl wurde willkürlich auf 30 beschränkt, um keine zu feinen Clustereinteilungen zu erhalten, die sehr viele Cluster enthalten.

In einem ersten Schritt wird zunächst die optimale Clusteranzahl bestimmt, mit der anschließend wieder eine Fuzzy-Clusteranalyse durchgeführt wird, wobei die Clusterzentren und die Partitionsmatrix berechnet wird. Als Ergebnis der Vorverarbeitung erhält man eine Menge von Clustereinteilungen, die vom Benutzer des System zu markieren sind.

Die berechneten Clusterzentren und Gewichte müssen nun geeignet als Beispiel aufbereitet werden. Eine einfache Möglichkeit besteht darin, die Clusterkoordinaten nach dem Gewicht sortiert aneinander zu hängen. Dies ist ein ähnlich triviales Verfahren wie in Abschnitt 3.4.1 ohne Vorverarbeitung der Daten.

3.4.5 Gewicht von Clustern in Intervallen

Das hier vorgestellte Verfahren benutzt wieder eine Fuzzy-Clusteranalyse zur Vorverarbeitung der Daten. Dabei werden aber nicht einfach die Clusterkoordinaten aneinander gehängt, sondern die Verteilung der Cluster im Raum soll möglichst gut abgebildet werden. Die Idee wurde aus den Verfahren mit der Bestimmung der Häufigkeiten in Intervallen übernommen. Hierbei wird der Raum in die gleichen Intervalle eingeteilt, wie in Kapitel 3.4.2 beschrieben. Diese Intervalle werden, wie in den vorhergehenden Verfahren, in der gleichen Reihenfolge durchlaufen. In den Intervallen wird die Summe der Gewichte der hierin aufgetretenen Cluster berechnet. Ist in einem Intervall kein Cluster aufgetreten, so wird als Summe der Gewichte null eingesetzt. Diese Werte dienen dann als ein

Beispiel für die Lernmaschine. Die Lage und die Gewichte der Cluster werden dadurch gut wiedergegeben.

Kapitel 4

Entwicklung der Analyse-Software

Um Experimente durchführen zu können, wurde eine Software entwickelt, die in die Protokollanalyse- und Emulationssoftware EasyTrace integriert wurde. Diese Software erweitert EasyTrace um ein Auswertungsfenster, das über einen Menüpunkt in der Hauptapplikation aufgerufen werden kann. Die Analysesoftware wurde in Form einer Dynamic Link Library (DLL) erstellt und zur Applikation gebunden. Dadurch entstand eine Software-Komponente, die dynamisch auch zu anderen Windows-Applikationen (z.B. Testanwendung oder dedizierte Analysesoftware) gebunden werden kann.

Die Software-Komponente übernimmt die Analyse der Daten und die Auswertung der Ergebnisse. Es werden dabei die in Abbildung 4.1 dargestellten Anwendungsfälle für den Benutzer ermöglicht. Die Software unterteilt sich demnach in die folgenden Programm-Module:

- Einleseprozess
- Fuzzy-Clusteranalyse mit grafischer Auswertung
- Support Vector Learning

Ein Benutzer kann dabei zwei Rollen einnehmen, aus denen er das System bedient. In der Rolle des Entwicklers ist es seine Aufgabe, das System zu lernen. Dafür müssen zunächst Beispiele aus den extrahierten Merkmalen generiert werden. Nach dem Öffnen einer Datei mit aufgezeichneten Tracedaten kann der Einleseprozess mit den angegebenen Optionen die Ausgangsdaten erzeugen. Die Datenmenge kann von der Fuzzy-Clusteranalyse zu Clustern zusammengefasst werden. Die mit Hilfe der Vorverarbeitung generierten Beispiele müssen anschließend vom Anwender als positiv oder negativ markiert werden. Die markierten Beispiele können dann zu den (evtl. bereits bestehenden) Trainingsdaten angehängt werden. Danach kann die Support Vector Machine die Klassifizierungsregel aus der Trainingsmenge neu lernen. Die Erkennungsleistung der SVM sollte anschließend überprüft werden.

Die trainierte Lernmaschine kann nun vom Fehlersuchenden eingesetzt werden, um Probleme zu diagnostizieren. Dafür muss er zunächst die Merkmale aus seinem Testbeispiel extrahieren. Dieses Beispiel kann dann von der SVM

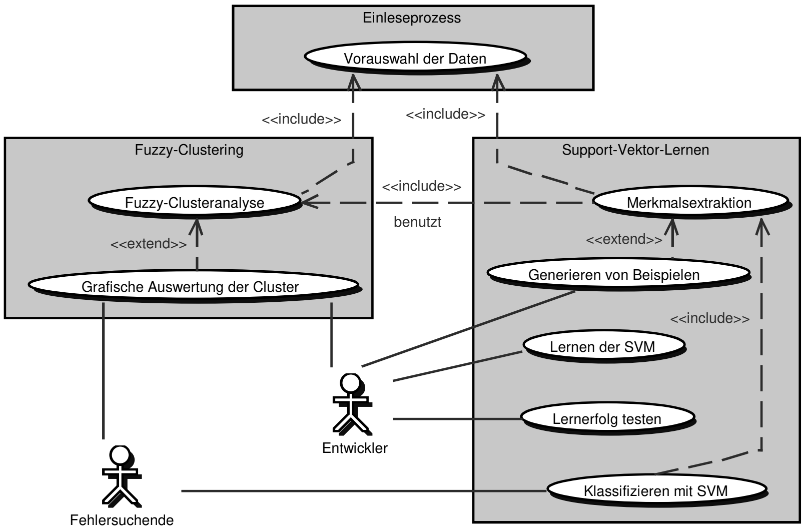


Abbildung 4.1: Anwendungsfalldiagramm

klassifiziert werden. In beiden Rollen steht außerdem die grafische Auswertung der Clustereinteilung als Hilfsmittel zur Verfügung. Damit können die Ergebnisse der Analyse veranschaulicht werden. Der Fehlersuchende erhält damit die Möglichkeit, die Ursache genauer zu lokalisieren.

4.1 Einlesevorgang

Der Einlesevorgang übernimmt die erste Stufe der Vorverarbeitung der Daten. Hier wird die Vorauswahl der Daten und die Speicherung in einer geeigneten Datenstruktur vorgenommen. Damit die Applikation durch diesen Vorgang nicht blockiert wird, ist dieser Prozess in einem eigenen Thread realisiert worden. In diesem Thread wird der gesamte Trace eingelesen, dekodiert, gefiltert und entsprechend aufbereitet. Für das Einlesen und Dekodieren werden Programmbibliotheken aus EasyTrace verwendet.

Der Einlesevorgang kann in einem Dialogfenster durch eine Fortschrittsanzeige kontrolliert werden. Der Dialog bietet jederzeit die Möglichkeit an, den Vorgang an der aktuellen Position abzubrechen. Dadurch bleiben die bis dahin eingelesen Daten erhalten und können weiter verarbeitet werden.

Filterung Die Filterung während des Einleseprozesses stellt sicher, dass nur die für die Analyse relevanten Daten berücksichtigt werden. Der Einlesevorgang bekommt den dekodierten ISDN-Datenstrom und filtert daraus die relevanten Frames. Da die Analyse ausschließlich auf Schicht 2 des ISO-OSI-Modells geschieht, werden die unwichtigen Statusmeldungen der Schicht 1 ignoriert. Man geht bei der Analyse davon aus, dass die Bitübertragungsschicht funktioniert.

Die Diagnose untersucht den Austausch von Signalisierungsinformationen zwischen dem Netzwerk und einem Endgerät. Da die Signalisierungsfunktion im HDLC durch den SAPI 0 angezeigt wird, werden Rahmen mit einem anderen

SAPI herausgefiltert. Da nur die Kommunikation zu *einem* Endgerät analysiert werden soll, werden während des Einleseprozesses nur die Rahmen berücksichtigt, die zu der Kommunikation mit dem Endgerät mit der gewünschten TEI gehören. Dafür kann der Benutzer im Optionen-Dialog die TEI von dem zu analysierenden Endgerät eintragen. Es werden ausschließlich die Rahmen berücksichtigt, die die eingestellte TEI oder die Broadcast-TEI (127) enthalten.

Aus dieser vorsortierten Menge werden nur noch die RR- und die INFO-Rahmen betrachtet. Es wird die Zeitdifferenz zwischen unmittelbar aufeinander folgender Rahmen berechnet. Daraus werden die in Kapitel 3.3 beschriebenen Punkte $(x, y, t)^T$ (siehe Gleichung 3.4) generiert und in einem Feld gespeichert. Die Größe des Feldes entspricht dabei der Anzahl der Punkte.

Während des Einleseprozesses wird bereits die Anzahl unterschiedlicher Rahmenpaare für jedes Zeitfenster effizient bestimmt. Jedes Nachrichtenpaar bekommt eine eindeutige Identifizierungsnummer, die einer Position in einem Bitvektor entspricht. Während des Einlesens der Nachrichtenpaare wird an der entsprechenden Position im Bitvektor eine eins gesetzt. Nach dem Einlesen entspricht die Anzahl der Einsen im Vektor der Anzahl der unterschiedlichen Nachrichtenpaare. Die Laufzeit des Algorithmus ist dementsprechend $O(n)$, wobei n die Anzahl der Nachrichtenpaare ist. Der Zusatzaufwand für die Berechnung macht sich allerdings in der Praxis nicht bemerkbar. Die Anzahl der unterschiedlichen Nachrichtenpaare dient als untere Schranke für die Clusteranzahl des Fuzzy-C-Means-Algorithmus. Dadurch kann die Bestimmung der optimalen Anzahl von Clustern beschleunigt werden, da unnötige Cluster-Berechnungen vermieden werden.

Skalierung Bereits beim Einlesevorgang wird die Skalierung der Zeitachse vorgenommen. Hier kann zwischen linearer und logarithmischer Skalierung gewählt werden. Die logarithmische Skalierung rechnet nach der folgenden Funktion um:

$$f(x) = 2(log_{10}x + 4). \tag{4.1}$$

Der kleinstmögliche Wert auf der Zeitachse ist 0,1 ms $= 10^{-4}$ s. Die Logarithmusfunktion 4.1 wurde so gewählt, dass der Wertebereich der Skalierung erhalten bleibt. Die Abbildung 4.2 zeigt den Graphen zu der Funktion. Durch die logarithmische Skalierung werden die Werte aus dem unteren Bereich besser aufgelöst. Die kleineren Werte werden stärker voneinander getrennt, dagegen liegen die größeren Werte näher zusammen.

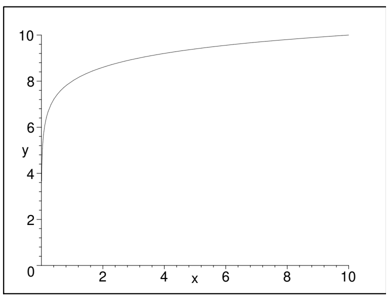


Abbildung 4.2: Graph der logarithmischen Skalierungsfunktion

Fenstereinstellung Beim Einlesevorgang werden die Tracedaten in Zeitfenster eingeteilt. Ein Fenster beinhaltet die in einem Zeitintervall aufgenommenen Daten und stellt ein Beispiel für die Analyse dar. Im Optionen-Dialog werden die Einstellungen zur Fenstergröße vorgenommen. Wird die Fenstereinteilung abgeschaltet, so wird die gesamte Aufnahme als *ein* Beispiel betrachtet. Die Fenstergröße kann in Sekunden oder in Anzahl von Frames angegeben werden. Zusätzlich kann die maximale Anzahl von Fenstern begrenzt werden.

4.2 Anwendung der Fuzzy-Clusteranalyse

Für die Fuzzy-Clusteranalyse wurde das Programm-Paket *fc* von Frank Höppner eingesetzt, das eine Reihe von Tools zur Verfügung stellt. Das Paket umfasst mehrere Algorithmen zum Fuzzy-Clustering, zur Initialisierung und zur Bewertung der Clustereinteilung. Die Tools stehen als Kommandozeilen-Programme zur Verfügung, die ihre Daten über Pipes weiterreichen. Die Daten sind dabei in dem speziellen Format „Data Description Language“ (DDL) abgelegt. Eine ausführliche Beschreibung des Pakets findet man in dem mitgelieferten Tutorial von Höppner [13].

Als Plattform für den Einsatz dieser Tools ist Unix oder Linux vorgesehen. Damit die Integration mit der Analysesoftware möglich wird, ist eine Version unter Windows erforderlich. Dafür wurde das Programmpaket mit dem GNU C/C++ Compiler aus Cygwin übersetzt. Cygwin bietet eine Unix-Umgebung für Windows an und besteht aus einer Emulationsschicht, die die Unix API Funktionalitäten unter Windows zur Verfügung stellt. In [25] wird das System ausführlich beschrieben.

Der Datenaustausch zwischen der Analysesoftware und dem fc-Tool geschieht über Dateien im DDL-Format. Daher wurden zunächst die zu analysierenden Ausgangsdaten in das DDL-Format konvertiert. Anschließend wurde das fc-Tool mit der erzeugten DDL-Datei als Parameter gestartet, das die Ergebnisse in einer DDL-Datei abspeichert. Diese Datei wurde dann wieder von der Analysesoftware eingelesen und in einer Datenstruktur abgelegt.

Bei der Benutzung der Tools fällt auf, dass das Fuzzy-Clustering schon bei kleinen Beispielen sehr rechenintensiv ist. Durch die Verwendung der Emulationsschicht von Cygwin ist die Laufzeit unter Windows noch erheblich länger als unter Linux. So dauert beispielsweise die Analyse einer einstündigen Aufzeichnung mit 1000 Frames ca. 20 Minuten. Damit ist zwar eine Echtzeitanalyse realisierbar, jedoch wäre eine Beschleunigung der Berechnung wünschenswert. Daher wurde versucht, die Ausführungszeit der Fuzzy-Clusteranalyse zu verkürzen. Es wurde analysiert, wo die meiste Rechenzeit beansprucht wird, damit genau da versucht werden kann, die Performanz zu steigern.

Trennung des Algorithmus Zum Starten des FCM-Algorithmus muss die Anzahl und die Lage der Cluster-Zentren vorgegeben werden. Bei unbekannten Daten ist leider vorher nicht bekannt, wie viele Cluster es gibt und wo diese liegen. Die Berechnung muss jeweils mit unterschiedlicher Anzahl von Clustern durchgeführt werden. Dabei wird die Clustereinteilung und ein Gütemaß berechnet. Die Partitionsmatrix ist hierbei nicht für jede Cluster-Anzahl interessant, sondern nur für die Anzahl, bei der das Gütemaß optimal ist.

Die Idee besteht darin, den Algorithmus in zwei Stufen zu trennen. Die erste Stufe berechnet zunächst nur die optimale Anzahl von Clustern, indem die Fuzzy-Clusteranalyse ohne Ausgabe der Partitionsmatrix wiederholt für alle Clusteranzahlen durchgeführt wird, wobei für jede Clusteranzahl das Gütemaß berechnet wird. Anhand dieses Gütemaßes wird dann die optimale Anzahl von Clustern bestimmt. Mit dieser optimalen Anzahl von Clustern wird erneut eine Fuzzy-Clusteranalyse mit Ausgabe der Partitionsmatrix gestartet. Man spart sich also die aufwändige Ausgabe von großen Partitionsmatrizen, die später nicht mehr benötigt werden. Die gesamte Ausführungszeit hat sich durch diese Optimierung ungefähr halbiert.

Außerdem wird es durch die Trennung des Algorithmus in zwei Stufen möglich, für die erste Stufe andere Parameter zu benutzen als für die zweite. Bei der Fuzzy-Clusteranalyse können Parameter eingestellt werden, die die maximale Anzahl von Iterationen beschränken bzw. die die Schwelle für den Abbruch der Iteration bestimmen. Dafür lassen sich im Programmpaket *fc* entsprechende Parameter einstellen. In der ersten Stufe könnte dies dazu genutzt werden, die Berechnung noch weiter zu beschleunigen auf Kosten von ungenaueren Ergebnissen. Die zweite Stufe wird dagegen mit hoher Genauigkeit durchgeführt, so dass insgesamt gute Ergebnisse bei hoher Effizienz erzielt werden.

Untere Schranke der Cluster-Anzahl Die Suche nach der optimalen Anzahl von Clustern macht trotz der Optimierung immer noch einen Großteil der Berechnnung aus. Daher scheint es sinnvoll, hier die Verbesserungen der Laufzeit anzusetzen. Eine Idee ist es, das Vorwissen beim Einlesen der Rohdaten anzuwenden. Da mindestens ein Cluster auf jeder Zeitachse sein muss, auf der sich Punkte befinden, ergibt sich als untere Schranke die Anzahl von Punkten belegten Zeitachsen. Bereits beim Einlesen der Daten kann dies effizient ermittelt werden. Die Suche nach dem Optimum kann dann bei dieser unteren Schranke begonnen werden. Die Bestimmung einer oberen Schranke bleibt leider schwierig und muss daher weiterhin grob abgeschätzt werden.

Der gesamte Ablauf des in der Software realisierten Fuzzy-Clustering-Verfahrens wird in Abbildung 4.3 veranschaulicht. Dieses Verfahren kann wahlweise für die gesamten Tracedaten, für einzelne Zeitfenster oder nacheinander für alle Zeitfenster durchgeführt werden. In einem Dialogfenster ist die Auswahl der Daten und die Eingabe der Parameter möglich. Nach dem Starten der Fuzzy-Clusteranalyse kann der Vorgang in einem Fortschrittsbalken beobachtet werden. Nach dem Beenden des Verfahrens werden die Koordinaten der Cluster eingelesen. Für jedes Cluster wird anschließend das Gewicht bestimmt, indem die Anzahl der zugehörigen Punkte ermittelt wird. Dabei gilt ein Punkt zu einem Cluster zugehörig, wenn er zu mehr als 50% einem Cluster angehört. Dieses Verfahren wird als Defuzzifizierung bezeichnet und wurde bereits von Barney in [2] eingesetzt. Dieses Gewicht wird danach normalisiert, indem die Werte durch die Summe der Gewichte dividiert werden.

Da die Streuung der zu einem Cluster zugehörigen Punkte interessant ist, wird die Standardabweichung des euklidischen Abstands der Punkte zum angehörigen Cluster berechnet. Sei $d_x, x = 1, \ldots, m$ der euklidische Abstand des Punktes p_x, dann berechnet sich die Standardabweichung σ wie folgt:

$$\sigma = \sqrt{d_1^2 + d_2^2 + \ldots + d_m^2}. \tag{4.2}$$

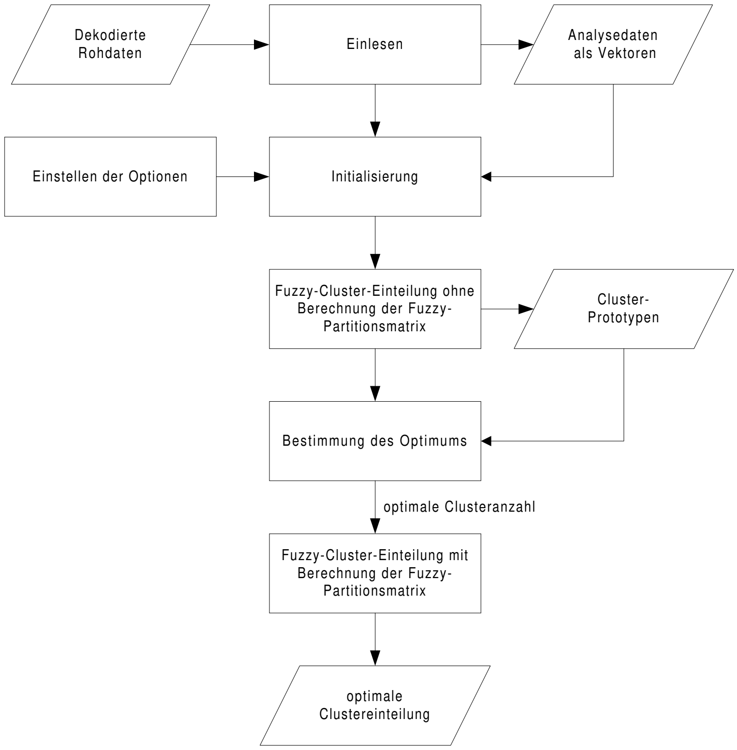


Abbildung 4.3: Fuzzy-Clusteranalyse-Methode

Die Ergebnisse der Analyse werden schließlich in textueller Form ausgegeben. Dabei werden in tabellarischer Form die Koordinaten, das normalisierte Gewicht und die Standardabweichung der Cluster angezeigt.

Grafische Auswertung Damit die Clustereinteilung besser beobachtet werden kann, ist eine grafische Ansicht der Ergebnisse einer Textdarstellung vorzuziehen. Um die Cluster sichtbar zu machen, werden sie als Kugeln im dreidimensionalen Raum dargestellt. Die relative Anzahl von zugehörigen Punkten stellt dabei das normalisierte Gewicht eines Clusters dar und wird durch die Farbe der Kugel wiedergegeben. Dabei durchlaufen die Farben mit wachsendem Gewicht die Regenbogenfarben von blau, grün, gelb bis rot. In der Grafik stellt die Farbe rot das maximale normalisierte Gewicht eines Clusters dar. Die anderen Farbwerte werden danach gleichmäßig verteilt.

In der Darstellung wird die Standardabweichung σ der zugehörigen Punkte eines Clusters durch den Radius der Kugeln dargestellt. Eine höhere Streuung wird als größere Kugel wiedergegeben. Daraus ergibt sich eine fünfdimensionale Darstellung der Cluster. Die ersten drei Dimensionen geben den Ort der Kugel wieder, die vierte Dimension (Gewicht) bestimmt die Farbe und die fünfte Dimension (Standardabweichung) drückt sich im Radius der Kugel aus.

Zweidimensionale Darstellung Für die grafische Auswertung wurde zunächst mit Hilfe der Borland-Entwicklungsumgebung ein zweidimensionales Diagramm zur Ansicht der Cluster entwickelt. Die ersten beiden Dimensionen spiegeln sich durch die Lage im Diagramm wider. Die Achsen sind dabei mit den entsprechenden Nachrichtenarten aus Tabelle 3.11 beschriftet, so dass eine Zuordnung der Cluster besser erkannt wird. Die dritte Dimension wird allerdings nur durch eine Beschriftung der Kugeln dargestellt. Die vierte und fünfte Dimension wird, wie oben beschrieben, durch die Farbe bzw. den Radius der Kugeln wiedergegeben. Man erhält somit eine Ansicht der Cluster aus der Vogelperspektive (siehe Abbildung 4.4).

Fährt man mit der Maus über einen Cluster, so werden die genauen Werte des Cluster angezeigt. Zusätzlich ist es möglich, entlang der Zeitachse ebenenweise tiefer in das Diagramm „hineinzugehen". Die Cluster, die über dem Schwellwert liegen, werden aus dem Diagramm ausgeblendet, und man bekommt eine Ansicht der restlichen Cluster. Damit können die entstandenen Cluster genauer beobachtet werden.

Häufig ist man an einer genauen Verteilung der zugehörigen Punkte zu den Clustern interessiert. Um dies visuell darzustellen, wurde eine Ansicht entwickelt, die die Verteilung der Punkte auf einer Zeitachse darstellt. Abbildung 4.5 zeigt einen beispielhaften Bildschirmausschnitt des Diagramms. Die Cluster werden als Projektion entlang einer ausgewählten Zeitachse dargestellt und als Balken sichtbar gemacht. Die Höhe des Balken gibt dabei das Gewicht des Clusters an. Die zugehörigen Punkte werden in einer zweiten Ebene in Form ihrer Verteilung über die Zeit aufgetragen. Eine dritte Ebene veranschaulicht zusätzlich die Zuordnung der Punkte zu ihren Clustern. Dadurch ist eine noch detailliertere Darstellung der Cluster und der dazugehörigen Punkte möglich.

Dreidimensionale Darstellung Die zweidimensionale Darstellung ist sehr hilfreich um die Zuordnung der Cluster zu den Nachrichtenpaaren zu beobach-

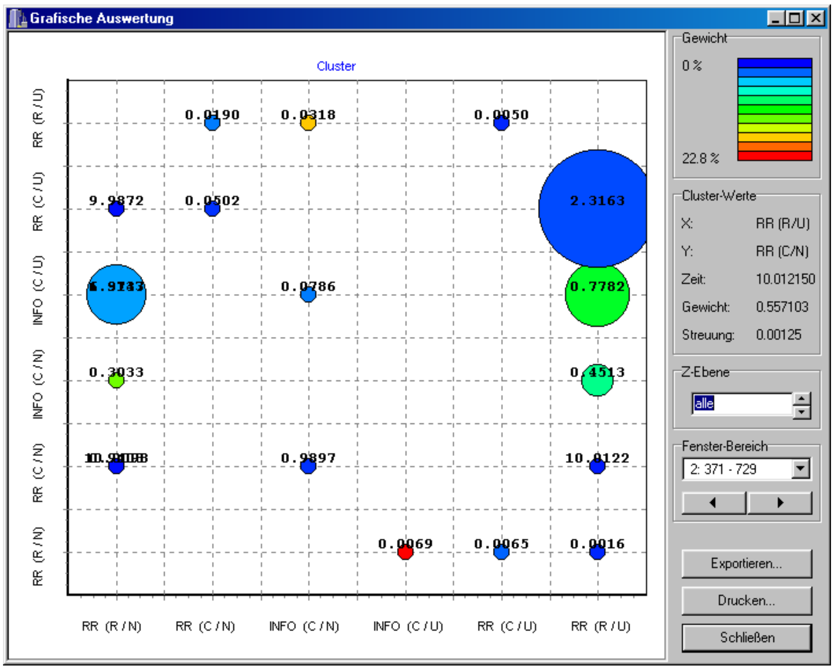


Abbildung 4.4: Zweidimensionale Ansicht der Cluster

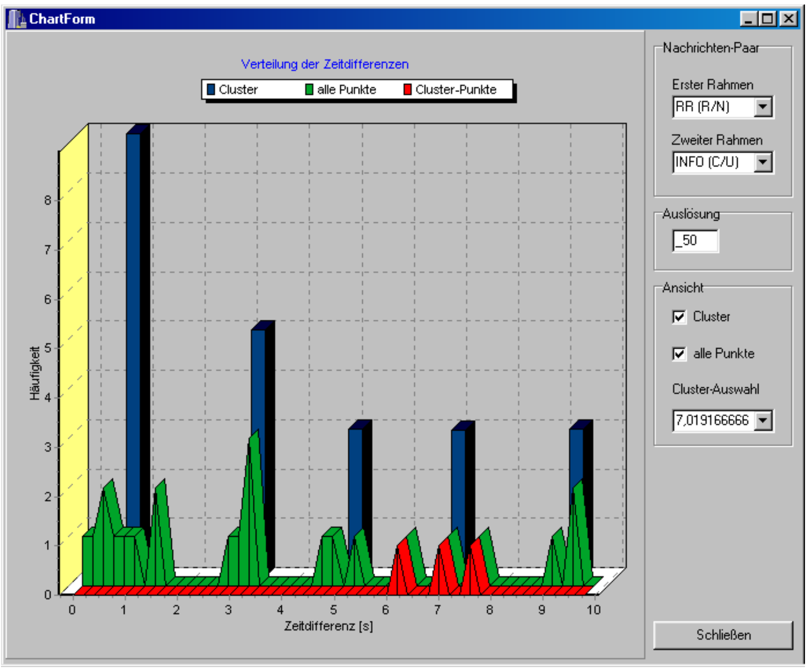


Abbildung 4.5: Ansicht der Verteilung entlang einer Zeitachse

ten. Darüber hinaus ist eine echte dreidimensionale Darstellung wünschenswert, um die Lage der Cluster im Raum zu erkennen. Dafür wurde eine dreidimensionale Visualisierung mit Hilfe der 3D-Grafikbibliothek *OpenGL* entwickelt (siehe Abbildung 4.6). OpenGL ist ein weitverbreiteter Standard zur Visualisierung

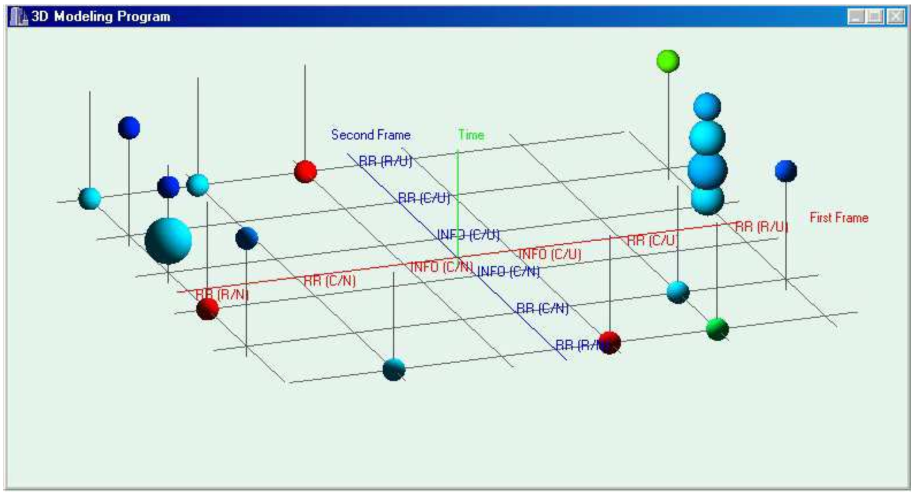


Abbildung 4.6: Dreidimensionale Ansicht der Cluster (orthogonal)

von 3D-Grafiken für mehrere Plattformen (siehe [28]). Die Positionen der Cluster werden dreidimensional dargestellt. Dabei kann die Perspektive orthogonal oder perspektivisch eingestellt werden. Bei der orthogonalen Ansicht schaut der Betrachter von außen auf den Raum. Die perspektivische Darstellung versetzt den Betrachter in den Raum, so dass die Kugeln ihn umgeben. Der Radius der Cluster spiegelt dabei die Streuung der Punkte wider, und die Farbe gibt das normalisierte Gewicht an.

Damit das Diagramm noch realistischer wirkt, lässt es sich mit Lichtquellen ausstatten, die die Kugeln anleuchten. Zudem kann man die Position des Betrachters ändern und somit das Diagramm von allen Seiten betrachten. Außerdem kann das ganze 3D-Szenario um die drei Achsen animiert werden. Lässt ein Betrachter z.B. in der orthogonalen Ansicht die Animation um die t-Achse drehen, so kann er von außen genau alle Kugeln beobachten. Hilfslinien von den Clustern zur x-y-Ebene erleichtern zudem die Zuordnung der Kugeln.

4.3 Anwendung der SVM

Als Support Vector Machine wurde die von Thorsten Joachims am Lehrstuhl für Künstliche Intelligenz der Universität Dortmund entwickelte Implementierung SVM^{light} V3.50 benutzt. Diese Implementierung bietet einen effizienten Optimierungs-Algorithmus, der auch Probleme mit mehreren Tausend Support Vektoren behandeln kann. Eine genaue Beschreibung findet man in [19].

Die in der Programmiersprache C erstellte SVM beinhaltet die beiden Konsolenprogramme *svm_learn* und *svm_classify*. Die Klassifizierungsregel wird im Modul svm_learn gelernt und in einer Model-Datei gespeichert. Das Modul

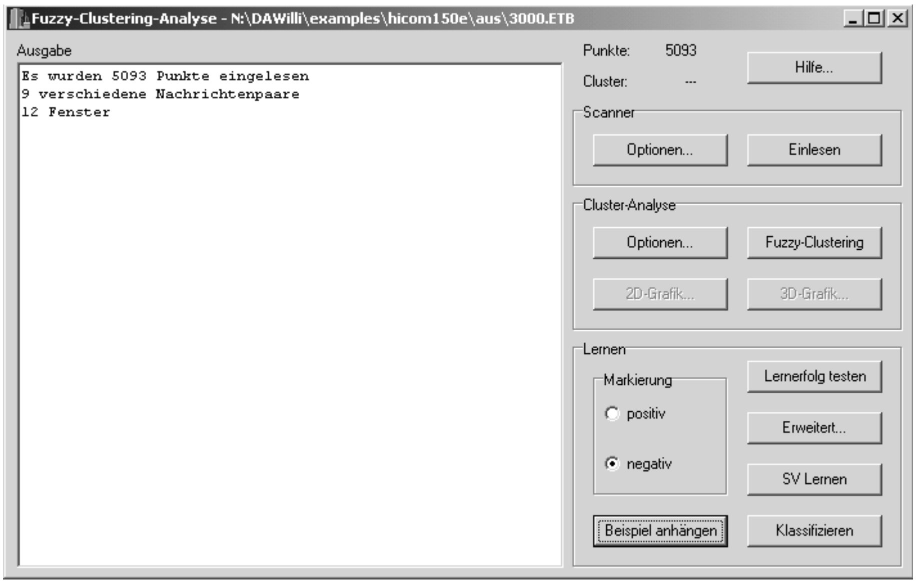


Abbildung 4.7: Screenshot der Benutzungsoberfläche der Software

svm_classify kann anschließend mit Hilfe der Model-Datei Testbeispiele klassifizieren.

Diese beiden Module wurden mit einem C++-Compiler unter Windows übersetzt und jeweils als Dynamic Link Library (DLL) zur Analyse-Software gebunden. Die Module werden über einen Funktionsaufruf gestartet, wobei die Kommandozeilen-Optionen als Parameter übergeben werden. Die Optionen lassen sich dabei komfortabel über einen Dialog einstellen.

4.4 Integration in bedienungsfreundliche Oberfläche

Zur Analyse der Daten werden viele Bearbeitungsschritte durchlaufen, die von einzelnen Werkzeugen erledigt werden. Bei der Benutzung der Tools fällt auf, dass noch sehr viel „Handarbeit“ notwendig ist, um die gewünschten Ergebnisse zu bekommen. Um die Handhabbarkeit zu verbessern, erscheint es sinnvoll, die Werkzeuge in eine benutzerfreundliche Oberfläche zu integrieren (siehe 4.7). Dadurch wird es möglich, viele Fälle komfortabel und effizient zu untersuchen. Dieses Software-Modul ist in die Applikation *EasyTrace für Windows* der Firma Herakom integriert worden und benutzt für das Fuzzy-Clustering das Programmpaket *fc* von Frank Höppner.

Kapitel 5

Untersuchungsdurchführung

5.1 Durchführung der Experimente

In diesem Kapitel werden die Untersuchungsergebnisse mit der SVM, die mit Hilfe der Analyse-Software auf den Testszenarien ermittelt worden sind, vorgestellt und verglichen. Die Ergebnisse sollen dazu dienen, die optimale Methode und Parametrierung für das Mustererkennungsproblem zu finden. Dafür muss zunächst ein geeignetes Leistungsmaß gefunden werden, dass in der Lage ist, die Erkennungsleistung der Lernmaschine gut zu bewerten. Die ersten Experimente dienen dazu, ein zweckmäßiges Leistungsmaß zu finden, mit dem die nachfolgenden Experimente bewertet werden. Dieses Leistungsmaß wird in Bezug auf die Genauigkeit der Erkennungsleistung und die dafür benötigte Rechenzeit ausgewählt.

Die Leistung der Lernmaschine wird mit der Kreuzvalidierung, dem Leave-One-Out-Verfahren und mit den $\xi\alpha$-Schätzern bewertet (vgl. Kapitel 2.5). Die Testmenge der Kreuzvalidierung enthält dabei 10% der gesamten Trainingsmenge. Auf der Restmenge wird die Lernmaschine trainiert. Zuvor werden die Trainingsdaten mit Hilfe eines Zufallsgenerators verwürfelt, da die Beispiele in geordneter Reihenfolge aneinander gehängt worden sind. Die $\xi\alpha$-Schätzer werden daraufhin untersucht, ob sie sinnvolle Abschätzungen in dieser Anwendung angeben.

Die Experimente werden mit unterschiedlichen Verfahren und Parametern durchgeführt. Wegen der Vielzahl von Parametern ist es sehr aufwändig, alle möglichen Kombinationen zu testen. Daher wird immer nur ein Parameter geändert, wobei alle anderen konstant bleiben. Dadurch wird nur der Einfluss *eines* Parameters auf die Leistung der SVM beobachtet. Mit der optimalen Einstellung dieses Parameters werden anschließend Versuche mit dem nächsten Parameter durchgeführt. Dieses Vorgehen führt somit zu einer guten, aber möglicherweise suboptimalen Lösung des Problems.

5.1.1 Vorverarbeitung der Ausgangsdaten

Zunächst erscheint es wichtig, die richtige Wahl für die Vorverarbeitung der Daten zu treffen, da die Art der Aufbereitung der Daten einen großen Einfluss auf die Leistung der Lernmaschine hat. Um dies zu untersuchen, werden die Verfahren der Vorverarbeitung (siehe 3.4) auf alle Daten der Testszenarien aus

Kapitel 3.2 angewendet. Dabei wird jeweils ein Ausschnitt von 5 Minuten aus den Tracedaten als ein Beispiel betrachtet, das entsprechend positiv oder negativ markiert wird. Die Beispiele stellen die Trainingsmenge dar, auf der die Lernmaschine trainiert wird.

Die Skalierung der Zeitachse ist linear, d.h. es wird keine Umrechnung der Zeitwerte vorgenommen. In einem weiteren Versuch wird das Lernergebnis bei logarithmischer Skalierung untersucht und mit dem linearen Fall verglichen.

Umfang der Beispiele:	5 Minuten
Skalierung der Zeitachse:	linear
Kernfunktion der SVM:	linear
Parameter C:	1

Tabelle 5.1: Parametrierung der Versuche

Die SVM^{light} benutzt eine lineare Entscheidungsfunktion und der Parameter C wird auf eins eingestellt. Auch diese Parameter werden in weiteren Tests genauer untersucht. Die Einstellungen der Ausgangsparameter werden in Tabelle 5.1 zusammengefasst und bleiben bei den folgenden Experimenten unverändert.

Keine Vorverarbeitung der Ausgangsdaten

Die Experimente mit dem Verfahren ohne Vorverarbeitung der Daten sollen dazu dienen, ein geeignetes Bewertungsmaß zu finden. Daher werden alle in Abschnitt 2.5 vorgestellten Leistungsmaße auf die vier Testszenarien angewendet und miteinander verglichen. Das Leistungsmaß, das sich am besten für die Bewertung des Lernerfolgs eignet, wird dann als einzige Methode für die weiteren Untersuchungen benutzt.

Die Ergebnisse des gewählten Leistungsmaßes bei der Methode ohne Vorverarbeitung der Daten stellen eine Referenz für die anderen Vorverarbeitungsmethoden dar. Es wird vermutet, dass die Ergebnisse für diese Methode keinen hohen Lernerfolg versprechen. Daher sollen die anderen Methoden eine Verbesserung gegenüber dieser Referenz zeigen.

Im ersten Szenario wurden jeweils die beiden Versuchsreihen und die Vereinigungsmenge dieser beiden Reihen untersucht. Die Ergebnisse der SVM, dargestellt in Tabelle 5.2, zeigen, dass die Fehlerrate gar nicht so schlecht ist, wie man vielleicht vermuten würde. Die Fehlerrate bei der Kreuzvalidierung und beim Leave-One-Out-Verfahren liegt jedoch über 10%, so dass der Lernmaschine nicht besonders viel Vertrauen geschenkt werden kann. Die Ergebnisse zwischen den Beispielen mit festen und zufällig verteilten Rufabständen liegen relativ nah beieinander. Den $\xi\alpha$-Schätzern nach erwartet man von dem Beispiel mit festen Rufabständen ein besseres Ergebnis. Die tatsächlichen Fehlerraten bei der Kreuzvalidierung und beim Leave-One-Out-Verfahren sind allerdings bei der Versuchsreihe mit negativ-exponentiell verteilten Rufen (Versuchsreihe 2) niedriger. Hier erzielt also die Versuchsreihe mit festen Rufabständen keinen Vorteil, obwohl man durch die gleichmäßigen Rufabstände ein besseres Ergebnis erwarten könnte. Vergleicht man die Resultate für Recall und Precision, so fällt auf, dass die $\xi\alpha$-Schätzer auch hier die tatsächlichen Werte weit unterschätzen. Bei der Vereinigungsmenge verbessert sich die Fehlerrate auf 10,76%. Der Wert

	Versuchsreihe		
	1	2	1 + 2
# Trainingsbeispiele	155	161	316
Kreuzvalidierung			
Fehlerrate	14,25%	12,98%	10,76%
Recall	90,40%	80,92%	89,78%
Precision	87,29%	95,18%	91,69%
Leave-One-Out			
Fehlerrate	15,48%	14,29%	
Recall	87,23%	80,90%	(siehe Text)
Precision	87,23%	92,31%	
$\xi\alpha$-Schätzer			
Fehler	$\leq$ 34,19%	$\leq$ 59,63%	$\leq$ 40,51%
Recall	$\geq$ 64,89%	$\geq$ 43,82%	$\geq$ 63,93%
Precision	$\geq$ 75,31%	$\geq$ 45,88%	$\geq$ 65,36%
Rechenzeit [s]	46,66	151,05	1713,61

Tabelle 5.2: Testergebnis im Szenario 1 ohne Vorverarbeitung

von 89,78% beim Recall zeigt dabei, dass die Fehler auf beide Klassen gleichmäßig verteilt sind, da die Anzahl der positiven Beispiele in etwa der Anzahl der negativen Beispiele entspricht.

Die benötigte Rechenzeit für einen Lerndurchgang ist allerdings relativ hoch und variiert sehr stark. So dauert der Lernvorgang beim Versuch mit festen Rufabständen noch unter einer Minute, die zweite Versuchsreihe benötigt jedoch gut zweieinhalb Minuten. Die dritte Testreihe mit 316 Beispielen beansprucht sogar mehr als 28 Minuten. Daher wird bei dem dritten Szenario auf eine Berechnung des Leave-One-Out-Errors verzichtet und nur eine Kreuzvalidierung durchgeführt.

Die Ergebnisse des zweiten Szenarios zeigen ebenfalls keine guten Ergebnisse der Lernmaschine (siehe Tabelle 5.3). Zwar wird bei der dritten Versuchs-

	Versuchsreihe			
	3	4	3 + 4	5
# Trainingsbeispiele	97	95	192	142
Kreuzvalidierung				
Fehlerrate	9,33%	48,22%	27,11%	52,43%
Recall	92,36%	52,12%	75,26%	48,79%
Precision	91,68%	52,33%	76,89%	48,23%
$\xi\alpha$-Schätzer				
Fehlerrate	$\leq$ 18,56%	$\leq$ 78,95%	$\leq$ 38,02%	$\leq$ 84,51%
Recall	$\geq$ 89,80%	$\geq$ 21,28%	$\geq$ 60,42%	$\geq$ 14,49%
Precision	$\geq$ 77,19%	$\geq$ 20,83%	$\geq$ 62,37%	$\geq$ 14,08%
Rechenzeit [s]	1,69	1,01	8,19	2,93

Tabelle 5.3: Testergebnis im Szenario 2 ohne Vorberarbeitung

reihe mit 9,33% eine noch bessere Fehlerrate erzielt als beim ersten Szenario, jedoch verschlechtern sich die Ergebnisse für die anderen Testreihen. Der Versuch mit festen Rufabständen ist hier diesmal besser als die Versuchsreihe mit negativ-exponentiell verteilten Rufen. Bei der fünften Versuchsreihe wird gar eine Fehlerrate von über 50% ermittelt. Die SVM ist damit nicht besser als eine Lernmaschine, die immer dieselben Aussagen macht. Die Fehlerrate streut zudem in einem weiten Bereich. Die $\xi\alpha$-Schätzer geben nur grobe Schätzungen für die Fehlerrate ab.

Die Rechenzeit für das Trainieren der Lernmaschine liegt zwar deutlich unter den Werten des ersten Szenarios, ist aber mit gut 8 Sekunden bei 192 Beispielen immer noch relativ hoch. Der Lernvorgang für das Trainieren der gesamten Menge dauert dagegen nur zwei Sekunden, jedoch beträgt die Laufzeit für die Kreuzvalidierung mehrere Minuten. Die Berechnung der Fehlerrate bei der Kreuzvalidierung dauert erheblich länger als die zehnfache Rechenzeit des einmaligen Lernvorgangs auf der gesamten Menge. Daher wird beim zweiten Szenario auf die Berechnung des Leave-One-Out-Fehlers verzichtet.

	Versuchsreihe		
	6	7	6 + 7
# Trainingsbeispiele	55	54	109
Kreuzvalidierung			
Fehlerrate	33,00%	33,00%	36,64%
Recall	25,93%	10,00%	26,67%
Precision	50,00%	37,50%	29,17%
Leave-One-Out			
Fehlerrate	34,55%	35,19%	38,53%
Recall	27,78%	11,76%	25,71%
Precision	45,45%	33,33%	36,00%
$\xi\alpha$-Schätzer			
Fehlerrate	$\leq$ 61,82%	$\leq$ 57,41%	$\leq$ 69,72%
Recall	$\geq$ 5,56%	$\geq$ 5,88%	$\geq$ 2,86%
Precision	$\geq$ 5,56%	$\geq$ 6,25%	$\geq$ 2,33%
Rechenzeit [s]	0,22	0,11	0,69

Tabelle 5.4: Testergebnis im Szenario 3 ohne Vorberarbeitung

Der Lernerfolg beim dritten Szenario kann ebenfalls nicht überzeugen (siehe Tabelle 5.4). Die von der Kreuzvalidierung und dem Leave-One-Out-Verfahren ermittelten Fehlerraten liegen bei den Testreihen 6 und 7 dicht beieinander. Die Ergebnisse verschlechtern sich, wenn man beide Testmengen vereinigt und darauf die Lernmaschine ansetzt. Besonders auffällig ist, dass die Werte für Recall sehr schlecht ausfallen. Es scheint also, dass vor allem die positiven Beispiele fehlerhaft klassifiziert werden. Die Abschätzungen der $\xi\alpha$-Schätzer weichen auch hier weit von den tatsächlichen Ergebnissen ab. Es fällt zudem auf, dass von einer Erhöhung der Abschätzung der $\xi\alpha$-Schätzer nicht auf eine tatsächlich höhere Fehlerrate geschlossen werden kann.

Die Ausführungszeiten für das Trainieren der Lernmaschine sind sehr kurz, allerdings enthalten die Versuchsreihen dieses Testszenarios weniger Beispiele als bei den zuvor untersuchten Szenarien. Die kurze Rechenzeit erlaubt diesmal die Berechnung des Leave-One-Out-Fehlers bei allen Testreihen. Hierbei fällt auf, dass die berechneten Gütemaße nah an den Ergebnissen der Kreuzvalidierung liegen.

	Versuchsreihe 8
# Trainingsbeispiele	111
Kreuzvalidierung	
Fehlerrate	0,91%
Recall	98,00%
Precision	100,00%
Leave-One-Out	
Fehlerrate	0,90 %
Recall	97,96 %
Precision	100,00 %
$\xi\alpha$-Schätzer	
Fehlerrate	$\leq$ 8,11%
Recall	$\geq$ 89,80%
Precision	$\geq$ 91,67%
Rechenzeit [s]	0,60

Tabelle 5.5: Testergebnis im Szenario 4 ohne Vorberarbeitung

Im vierten Szenario erreicht die SVM eine sehr hohe Erkennungsgenauigkeit (siehe Tabelle 5.5). Im Vergleich zu den bisher betrachteten Testszenarien werden hier eine sehr niedrige Fehlerrate und hohe Werte für Recall und Precision erreicht. Es fällt wieder auf, dass die Leistungsmaße der Kreuzvalidierung und des Leave-One-Out-Verfahrens fast identische Werte ermitteln. Die $\xi\alpha$-Schätzer überschätzen die tatsächliche Fehlerrate.

Die Experimente auf den Testszenarien zeigen, dass die Werte der Kreuzvalidierung und des Leave-One-Out-Verfahrens sehr dicht beieinander liegen. Somit kann man beide Verfahren als vertrauenswürdig einschätzen. Eine Kreuzvalidierung, bei der die Testmenge 10% der Gesamtmenge enthält, kommt sehr nah an die Werte des Leave-One-Out-Verfahrens heran. Somit reicht es aus, nur die Kreuzvalidierung durchzuführen. Man spart dadurch viel Rechenzeit, da bei der Kreuzvalidierung der Lernalgorithmus nur zehnmal ausgeführt wird, im Gegensatz zu n-mal beim Leave-One-Out.

Die $\xi\alpha$-Schätzer enthalten nur eine grobe Aussage über den erzielten Lernerfolg. Die Abschätzungen liegen zum Teil weit von den tatsächlichen Ergebnissen entfernt. Das Erhöhen des Wertes für die Abschätzung der Fehlerrate muss nicht zu einer Erhöhung der tatsächlichen Fehlerrate führen. Daher wird ein Vergleich zweier Lernmethoden schwierig. Den $\xi\alpha$-Schätzern kann somit nicht viel Vertrauen geschenkt werden.

Aus diesen Gründen wird die Kreuzvalidierung zur Bewertung der Leistung der SVM bevorzugt. Dabei wird nur noch die Fehlerrate betrachtet, da bei den Experimenten die Methoden und Parameter auf die Tauglichkeit zur Erkennung von positiven als auch von negativen Beispielen untersucht werden sollen. Die Abbildung 5.1 fasst die Fehlerraten, die bei den drei Bewertungsmethoden ermittelt wurden, in einem Diagramm zusammen. Leider konnten die Leave-Ont-Out-Fehlerraten im zweiten Szenario nicht berechnet werden (siehe Seite

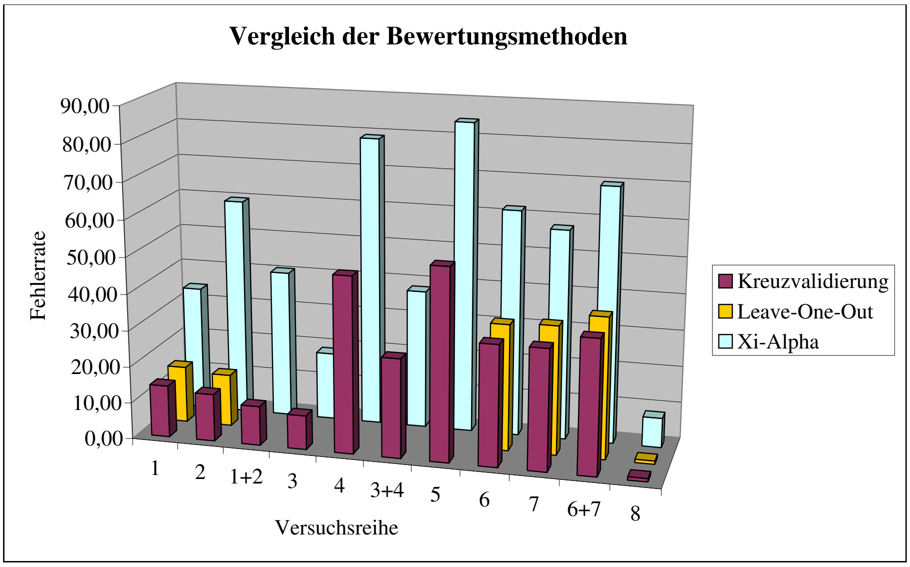


Abbildung 5.1: Vergleich der Leistungsmaße

76). Man sieht jedoch bei den anderen Versuchsreihen, dass die Kreuzvalidierung und die Leave-Ont-Out-Methode fast identische Ergebnisse liefern. Die $\xi\alpha$-Schätzer tendieren dazu, den tatsächlichen Fehler deutlich zu überschätzen.

Wird die Erkennung der positiven Beispiele wichtig, so können die Leistungsmaße Recall und Precision herangezogen werden. Bei den folgenden Experimenten mit den anderen Vorverarbeitungsmethoden werden diese Leistungsmaße jedoch nicht mehr berücksichtigt. Stattdessen wird nur noch die Fehlerrate bei der Kreuzvalidierung ermittelt.

Absolute Häufigkeit von Messpunkten in Intervallen

Bei diesem Verfahren wird der Raum in kleine Intervalle eingeteilt, in denen die absolute Häufigkeit von Punkten bestimmt wird (siehe Abschnitt 3.4.2). Dabei muss experimentell eine geeignete Intervallgröße gefunden werden. Bei sehr kleinen Intervallen wird die Verteilung der Punkte in dem Raum zwar exakter wiedergegeben, jedoch steigt die Laufzeit für die Berechnung an. Es muss also ein guter Kompromiss zwischen der Auflösung und der Ausführungszeit gefunden werden. Die Intervallgröße ist von 10 ms bis 1 s in logarithmischen Stufen erhöht worden, um eine passende Größenordnung zu finden.

Die Versuche mit den Testreihen aus Szenario 1 zeigen hervorragende Ergebnisse bei der Erkennungsleistung der SVM (siehe Tabelle 5.6). Hier erkennt man eine enorme Verbesserung der Leistung gegenüber dem Verfahren ohne Vorverarbeitung. Die besten Lernergebnisse werden bei der Intervallgröße von 10 ms erzielt. Hierbei werden alle Beispiele der Versuchsreihen mit der Kreuzvalidierung fehlerfrei klassifiziert.

Bei den Intervallgrößen 100ms und 1000ms wird die Erkennungsleistung bei der zweiten Testreihe etwas schlechter. Die Ergebnisse der Vereinigungsmenge (Reihe $1 + 2$) liegen zwischen den Werten der beiden ersten Testreihen. Die erste

Intervalllänge	Versuchsreihe		
	1	2	1 + 2
10 ms	0,00%	0,00%	0,00%
100 ms	0,00%	1,25%	0,31%
1000 ms	0,00%	1,88%	0,63%

Tabelle 5.6: Fehlerrate im Szenario 1 mit dem Verfahren „Absolute Häufigkeit in Intervallen“

Versuchsreihe zeigt dagegen bei allen Intervallgrößen eine fehlerfreie Klassifikation. Bei den Emulationen mit festen Rufabständen scheint also die Intervallgröße keine Rolle zu spielen. Die Lernmaschine erkennt auch bei schlechter Auflösung des Raumes eine exakte Klassifikationsregel. Feste Rufabstände kommen allerdings nicht im Echtbetrieb vor, sondern können nur in Testszenarien emuliert werden. Für die Analyse im Echtbetrieb ist die in der Praxis meist vorkommende negativ-exponentielle Verteilung der Rufabstände zu betrachten. Bei der Versuchsreihe mit dieser Verteilung ist eine Auflösung von 10 ms erforderlich, um genaue Ergebnisse zu erhalten.

Der Rechenaufwand für einen Lerndurchgang der SVM liegt dabei immer unter einer Sekunde. Daher wird bei diesem Verfahren auf eine exakte Messung und Notierung der Rechenzeiten verzichtet. Die gute Erkennungsleistung des Verfahrens bei diesem Szenario ist nicht verwunderlich, da mit steigender Auslastung des D-Kanals auch die absolute Häufigkeit von Punkten steigt. Damit nimmt auch die Anzahl von Punkten in einzelnen Intervallen zu. Hinzu kommt, dass die Punkte bei steigender D-Kanal-Auslastung sich anders verteilen. Der Lernalgorithmus ist hier also in der Lage die Auslastung des D-Kanals gut zu erkennen.

Intervalllänge	Versuchsreihe			
	3	4	3 + 4	5
10 ms	0,00%	0,00%	0,00%	2,86%
100 ms	0,00%	0,00%	0,00%	4,90%
1000 ms	0,00%	0,00%	0,53%	13,52%

Tabelle 5.7: Fehlerrate im Szenario 2 mit dem Verfahren „Absolute Häufigkeit in Intervallen“

Beim zweiten Szenario werden ähnlich gute Ergebnisse erzielt wie bei den ersten beiden Versuchsreihen, obwohl hier ein anderes Klassifikationskriterium gewählt wurde (siehe Tabelle 5.7). Die besten Ergebnisse erzielt man auch hier mit der Intervallgröße von 10 ms. Die ersten beiden Versuchsreihen erreichen bei allen Intervalllängen eine fehlerfreie Klassifikation. Die Vereinigungsmenge hat dagegen bei der Auflösung von 1000 ms eine Fehlerrate von 0,53%. Nur das erweiterte Beispiel der Versuchsreihe 5 hat bei 10 ms noch eine Fehlerrate von 2,86%. Auch hier wird die Erkennung mit größer werdenden Intervallen ungenauer. Bei dem Test mit 1000 ms lässt die Erkennungsgenauigkeit mit 13,52% Fehlerrate stark nach. Auffällig ist hierbei der Anstieg der Rechenzeit auf knapp 10 Sekunden. Bei allen anderen Tests liegt die Zeit dagegen unter einer Sekunde.

Um zu erklären, warum die Lernmaschine diese hohe Erkennungsgenauigkeit erreicht, wird eine Fuzzy-Clusteranalyse für beide Klassen durchgeführt. Die Clustereinteilung eines typischen Beispiels wird jeweils in der zweidimensionalen Ansicht der Analysesoftware in Abbildung 5.2 dargestellt. Schaut man sich darin die Cluster mit hohem Gewicht genauer an, so fällt der Cluster bei INFO (C/N) → RR (R/U) auf. Im unbelasteten Fall ist dieser Cluster bei 6,7 ms (siehe Abbildung 5.2a). Bei Belastung steigt dieser Cluster auf 196,7 ms an (siehe Abbildung 5.2b). Die Zeitdifferenz nimmt im belasteten Zustand also deutlich zu. Man erkennt daran, dass der User etwas mehr Zeit benötigt, um einen INFO-Rahmen vom Netz mit einem RR-Rahmen zu quittieren. Dies ist ein deutliches Anzeichen dafür, dass das Endsystem unter Last ist. Die Verteilung der Punkte auf dieser Zeitachse verschiebt sich dementsprechend auf die Intervalle mit größeren Zeitwerten. Durch die Anzahl von Punkten in den Intervallen wird die Verteilung der Punkte wiedergegeben. Dies erklärt auch, warum die SVM bei dieser Art von Vorverarbeitung die hohe Erkennungsleistung erzielt.

Allerdings verschieben sich nicht alle Cluster. So bleibt der Cluster bei INFO (C/U) → RR(R/N) konstant bei 7,1 ms. Dieser Cluster stellt die Reaktionszeit der Anlage auf ein INFO-Rahmen des Users da. Daran sieht man, dass die Anlage nicht belastet ist, da sie in beiden Fällen mit gleicher Verzögerung die INFO-Rahmen des Users quittiert.

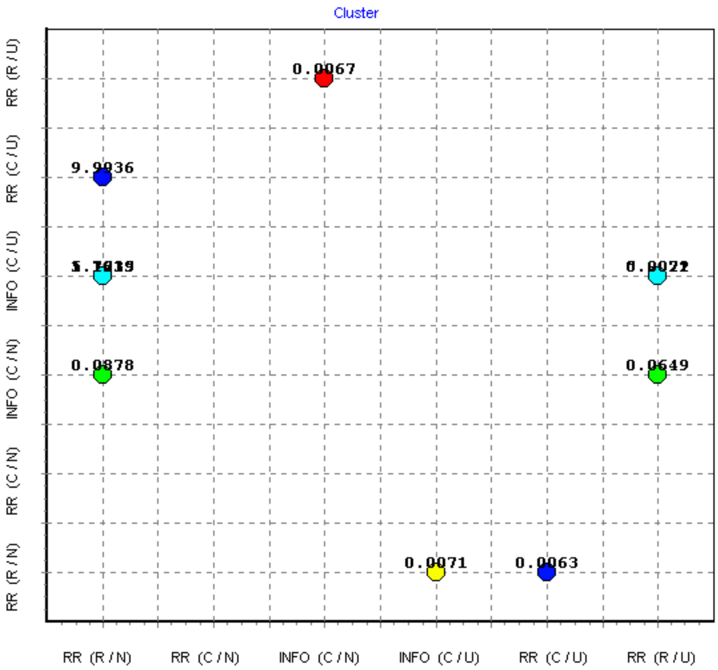
Intervalllänge	Versuchsreihe		
	6	7	6 + 7
10 ms	0,00%	1,67%	0,00%
100 ms	10,67%	9,00%	11,00%
1000 ms	18,67%	22,67%	24,82%

Tabelle 5.8: Fehlerrate im Szenario 3 mit dem Verfahren „Absolute Häufigkeit in Intervallen“

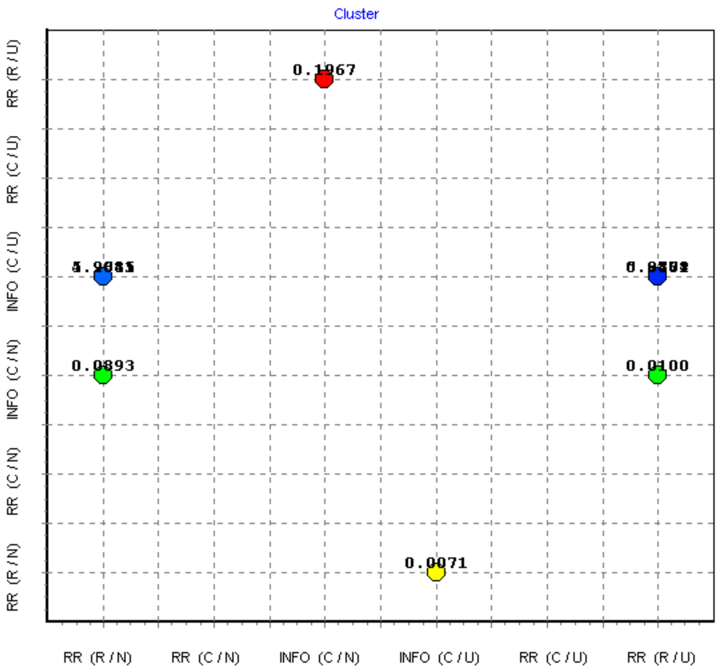
Beim dritten Szenario ist der Unterschied zwischen den Ergebnissen der Tests mit unterschiedlichen Intervalllängen noch deutlicher. Bei der Intervalllänge von 10 ms wird noch eine nahezu fehlerfreie Klassifikation erreicht, so steigt die Fehlerrate bei den beiden anderen Intervalllängen stark an. Die Resultate bei gleichen Intervallgrößen zwischen den Versuchsreihen liegen nah beieinander, wobei die Vereinigungsmenge (Reihe 6 und 7) eine höhere Fehlerrate aufweist. Bei diesem Szenario ist es also noch wichtiger, eine hohe Auflösung zu wählen. Erst dann erzielt man gute Lernergebnisse mit der SVM.

Die letzte Versuchsreihe zeigt eine fehlerlose Klassifikation aller Beispiele bei allen Intervalllängen. Der Lernalgorithmus erkennt bei diesem Szenario einen deutlichen Unterschied zwischen den beiden getesteten Anlagen. Die automatische Klassifikation eines unbekannten Beispiels zu dem Anlagenhersteller ist also möglich.

Aus den Ergebnissen der durchgeführten Tests lässt sich schlussfolgern, dass mit dem Verfahren der Bestimmung der absoluten Häufigkeit in Intervallen die Erkennung von unbekannten Beispielen nach unterschiedlichen Klassifikationskriterien möglich ist. Die Vorhersage wird dabei umso genauer, je größer die



a) Positiver Fall: unbelastet



b) Negativer Fall: belastet

Abbildung 5.2: Clustereinteilung im Szenario 2

Intervalllänge	Versuchsreihe 8
10 ms	0,00%
100 ms	0,00%
1000 ms	0,00%

Tabelle 5.9: Fehlerrate im Szenario 4 mit dem Verfahren „Absolute Häufigkeit in Intervallen"

Auflösung des Raumes ist. Vor allem das dritte Szenario zeigt deutlich, dass eine Auflösung in der Größenordnung von 10 ms notwendig ist, um eine nahezu fehlerfreie Klassifikation zu ermöglichen.

Relative Häufigkeit von Messpunkten in Intervallen

Dieses Verfahren der Vorverarbeitung der Daten unterscheidet sich zu dem vorhergehenden dadurch, dass hier die relative Häufigkeit anstatt der absoluten berechnet wird (siehe Abschnitt 3.4.3). Auch bei diesem Verfahren muss wieder eine geeignete Größe der Intervalle bestimmt werden. Es werden dafür dieselben Intervallgrößen benutzt wie bei dem zuvor untersuchten Verfahren. Dadurch ist ein direkter Vergleich der Methoden bei gleichen Intervallgrößen möglich.

Intervalllänge	Versuchsreihe		
	1	2	1 + 2
10 ms	0,00%	0,00%	0,00%
100 ms	0,00%	0,63%	0,31%
1000 ms	0,00%	1,25%	0,63%

Tabelle 5.10: Fehlerrate im Szenario 1 mit dem Verfahren „Relative Häufigkeit in Intervallen"

Im Szenario 1 werden ähnlich gute Ergebnisse erzielt, wie mit dem Verfahren der Bestimmung der absoluten Häufigkeit in Intervallen. Es wird ab einer Auflösung von 10 ms eine fehlerfreie Klassifikation der Beispiele erreicht (siehe Tabelle 5.10). Die zweite Versuchsreihe erzielt sogar eine geringere Fehlerrate. Werden die Intervalle vergrößert, so verringert sich auch hier die Genauigkeit der Klassifizierung.

Die Ergebnisse des ersten Szenarios zeigen, dass auch dieses Verfahren für eine fehlerfreie Klassifikation eingesetzt werden kann. Intuitiv könnte man vermuten, dass das Verfahren mit der absoluten Häufigkeit bei diesem Szenario bessere Ergebnisse erzielt als dieses Verfahren. Denn bei dem ersten Szenario wird bei einer höheren Auslastung des D-Kanals auch mehr Nachrichten verschickt. Dadurch steigt natürlich die absolute Anzahl von Punkten. Dass die relative Häufigkeit genauso gute Werte für den Lernerfolg liefert wie das vorgehende Verfahren, zeigt, dass sich offensichtlich die *Verteilung* der Punkte auf die Intervalle geändert hat. In einigen Intervallen entstehen neue Punkthäufungen und in anderen verschwinden sie. Denn bei einem gleichmäßigen relativen Anstieg der Häufungen von Punkten, bleibt in den Intervallen die relative Häufigkeit konstant. Daraus lässt sich schlussfolgern, dass durch die Erhöhung der

Auslastung auf dem D-Kanal die Anzahl von Punkten in Intervallen nicht gleichmäßig zunimmt, d.h. die Punkthäufungen verlagern sich auf andere Intervalle. Dementsprechend verlagern sich auch die Cluster. Daher kann die Methode mit der Bestimmung der relativen Häufigkeit genauso gut angewendet werden wie die Methode mit der absoluten Häufigkeit.

Intervalllänge	Versuchsreihe			
	3	4	3 + 4	5
10 ms	0,00%	0,00%	0,00%	0,71%
100 ms	0,00%	0,00%	0,00%	4,19%
1000 ms	0,00%	0,00%	0,00%	12,00%

Tabelle 5.11: Fehlerrate im Szenario 2 mit dem Verfahren „Relative Häufigkeit in Intervallen"

Die in Tabelle 5.11 dargestellten Ergebnisse zeigen auch beim zweiten Szenario hervorragende Ergebnisse. Auffällig ist hierbei, dass die Beispiele aus den Versuchsreihen 3 und 4 bei allen Intervalllängen fehlerfrei erkannt werden. Bei der fünften Testreihe steigt die Fehlerrate zwar mit abnehmender Auflösung an, bleibt aber unter den Fehlerraten des Verfahrens mit absoluter Häufigkeit. Bei der hohen Auflösung von 10 ms wird eine sehr gute Leistung erreicht.

Intervalllänge	Versuchsreihe		
	6	7	6 + 7
10 ms	1,67%	0,00%	0,00%
100 ms	5,33%	5,33%	8,27%
1000 ms	12,33%	14,00%	26,36%

Tabelle 5.12: Fehlerrate im Szenario 3 mit dem Verfahren „Relative Häufigkeit in Intervallen"

Die Ergebnisse, die für das dritte Szenario ermittelt wurden, sind ebenfalls mit dem Ergebnis des vorhergehenden Verfahrens vergleichbar. Bei der Auflösung von 10 ms wird auch hier eine beinahe fehlerfreie Klassifikation erreicht. Hier werden lediglich mit der sechsten Reihe Fehlklassifikationen beobachtet. Der steile Anstieg der Fehlerrate bei zunehmender Intervallgröße ist auch hier zu beobachten.

Intervalllänge	Versuchsreihe 8
10 ms	0,00%
100 ms	0,00%
1000 ms	0,00%

Tabelle 5.13: Fehlerrate im Szenario 4 mit dem Verfahren „Relative Häufigkeit in Intervallen"

Die Ergebnisse im vierten Szenario sind identisch mit denen aus Kapitel 5.1.1. Alle Testbeispiele werden wieder fehlerlos klassifiziert. Die Daten sind auch mit diesem Verfahren gut trennbar.

Das Verfahren mit der Bestimmung der relativen Häufigkeit erweist sich als sehr zuverlässige und universelle Methode, um eine gute Erkennungsleistung bei allen Szenarien zu erzielen. Die ermittelten Ergebnisse zeigen, dass diese Vorverarbeitung für unterschiedliche Klassifikationskriterien angewendet werden kann. Bei allen Szenarien wird eine nahezu fehlerfreie Klassifikation möglich, wenn die Auflösung hoch genug ist. Eine Intervalleinteilung in der Größenordnung von 10 ms ist dafür allerdings notwendig.

Koordinaten der Cluster

Bei diesem Verfahren wird eine vorangegangene Fuzzy-Clusteranalyse zur Vorverarbeitung der Daten verwendet. Zunächst wird das einfache Verfahren untersucht, bei dem die Clusterkoordinaten in der Reihenfolge der Clustergewichte aneinander gehängt werden (siehe Abschnitt 3.4.4).

	Versuchsreihe		
	1	2	1 + 2
Fehlerrate	17,88%	12,50%	15,70%

Tabelle 5.14: Fehlerrate im Szenario 1 mit dem Verfahren „Koordinaten der Cluster“

Die Leistung der Lernmaschine für die Versuchsreihen im ersten Szenario ist in Tabelle 5.14 dargestellt. Hier erkennt man, dass keine guten Lernerfolge erzielt werden. Die Fehlerrate bleibt immer über 10% und ist mit 17,88% in der ersten Versuchsreihe sogar über dem Wert des Verfahrens ohne Vorverarbeitung. Das Ergebnis der Vereinigungsmenge liegt zwischen den Werten für die beiden einzeln betrachteten Reihen. Der Rechenaufwand für die Vorverarbeitung mit dem Fuzzy-Clustering-Verfahren dauert dabei mehr als eine Stunde.

	Versuchsreihe			
	3	4	3 + 4	5
Fehlerrate	19,89%	33,00%	17,63%	14,86%

Tabelle 5.15: Fehlerrate im Szenario 2 mit dem Verfahren „Koordinaten der Cluster“

Die Erkennungsleistung der SVM fällt im zweiten Szenario noch schlechter aus als im ersten Szenario. Bei der dritten Versuchsreihe wird eine höhere Fehlerrate ermittelt als bei dem Verfahren ohne Vorverarbeitung. Allerdings ist dies bei den anderen Testreihen dieses Szenarios umgekehrt. Eine Regelmäßigkeit lässt sich nicht feststellen.

Auch im dritten Szenario können die Ergebnisse ähnlich ausgewertet werden. Der Loss liegt erheblich über den Werten der beiden vorhergehenden Verfahren, die die Häufigkeit von Punkten in Intervallen bestimmen. Es werden vergleichbar schlechte Resultate wie mit dem ersten Verfahren ohne Vorverarbeitung ermittelt.

Bei dem vierten Szenario verschlechtert sich die Fehlerrate gegenüber dem ersten Verfahren sogar erheblich. Beim Verfahren ohne Vorverarbeitung wurde

	Versuchsreihe		
	6	7	6 + 7
Fehlerrate	37,00%	25,67%	40,27%

Tabelle 5.16: Fehlerrate im Szenario 3 mit dem Verfahren „Koordinaten der Cluster“

	Versuchsreihe
	8
Fehlerrate	18,94 %

Tabelle 5.17: Fehlerrate im Szenario 4 mit dem Verfahren „Koordinaten der Cluster“

noch eine Fehlerrate von 0,91% bei der Kreuzvalidierung erzielt, wogegen hier die Rate auf 18,94% gestiegen ist. Der Rechenaufwand ist dabei ein Vielfaches größer als beim ersten Verfahren.

Das Lernproblem kann mit dieser Art von Vorverarbeitung der Daten nicht gelöst werden. Die Fehlerrate ist dafür zu hoch und liegt teilweise erheblich über den Werten, die ohne Vorverarbeitung erzielt worden sind. Dieses Verfahren kann daher nicht eingesetzt werden.

Hinzu kommt die hohe Rechenzeit für die Fuzzy-Clusteranalyse. Je nach Auslastung des D-Kanals dauert die Clusteranalyse für ein Beispiel ca. zwei Minuten. Ein Beispiel war in diesen Fällen ein Ausschnitt von fünf Minuten. Daher ist noch eine Analyse in Echtzeit möglich. Allerdings nimmt eine nachträgliche Analyse einer aufgezeichneten Aufnahme viel Zeit in Anspruch. Die Ergebnisse, die man dafür erhält, sind aber nicht akzeptabel.

Gewicht von Clustern in Intervallen

Bei diesem Verfahren wird zumindest eine Verbesserung der Erkennungsleistung gegenüber dem trivialen Verfahren mit der sortierten Aneinanderreihung der Clusterkoordinaten erwartet. Der Nachteil des hohen Zeitaufwands bleibt aber bestehen, da der gleiche FCM-Algorithmus verwendet wird.

Intervalllänge	Versuchsreihe		
	1	2	1 + 2
10 ms	0,62%	1,25%	0,97%
100 ms	0,00%	3,12%	1,57%
1000 ms	0,00%	2,50%	0,96%

Tabelle 5.18: Fehlerrate im Szenario 1 mit dem Verfahren „Gewicht von Clustern in Intervallen“

Die Lernergebnisse im ersten Szenario sind deutlich besser als mit dem trivialen Verfahren aus dem vorhergehenden Kapitel. Vor allem die erste Versuchsreihe erreicht eine gute Erkennungsgenauigkeit. Erstaunlicherweise werden die Ergebnisse besser, je größer die Intervalllänge wird. Dieses Phänomen wird allerdings nicht bei den anderen Versuchsreihen beobachtet. Die zweite Versuchsreihe erzielt dagegen die besten Ergebnisse mit der höchsten Auflösung von 10 ms. Die Ergebnisse der Vereinigungsmenge der ersten beiden Versuchsreihen liegen dabei zwischen den Leistungen der beiden Reihen.

Intervalllänge	Versuchsreihe			
	3	4	3 + 4	5
10 ms	0,00%	0,00%	0,00%	7,86%
100 ms	0,00%	0,00%	0,00%	12,10%
1000 ms	2,10%	1,10%	3,10%	15,62%

Tabelle 5.19: Fehlerrate im Szenario 2 mit dem Verfahren „Gewicht von Clustern in Intervallen“

Die guten Ergebnisse werden auch in den Testreihen 3 und 4 erzielt (siehe Tabelle 5.19). Allerdings ist die Fehlerrate bei der fünften Versuchsreihe bei hohen 7,86% und verschlechtert sich bei steigender Intervalllänge noch mehr.

Intervalllänge	Versuchsreihe		
	6	7	6 + 7
10 ms	11,33%	16,33%	12,82%
100 ms	14,33%	10,00%	14,73%
1000 ms	29,67%	20,67%	22,00%

Tabelle 5.20: Fehlerrate im Szenario 3 mit dem Verfahren „Gewicht von Clustern in Intervallen“

Im dritten Szenario verschlechtern sich die Ergebnisse noch mehr, sind aber noch besser als bei dem vorhergehenden Verfahren (siehe Tabelle 5.20). In der achten Versuchsreihe wird dagegen wieder bei allen Intervallgrößen eine fehlerfreie Klassifikation erreicht.

Das Verfahren erzielt also je nach Testszenario teilweise sehr gute, aber nicht durchweg zufriedenstellende Resultate. Der hohe Zeitaufwand, der für das Verfahren in Kauf genommen werden muss, steht nicht im Verhältnis zu den erzielten Ergebnissen.

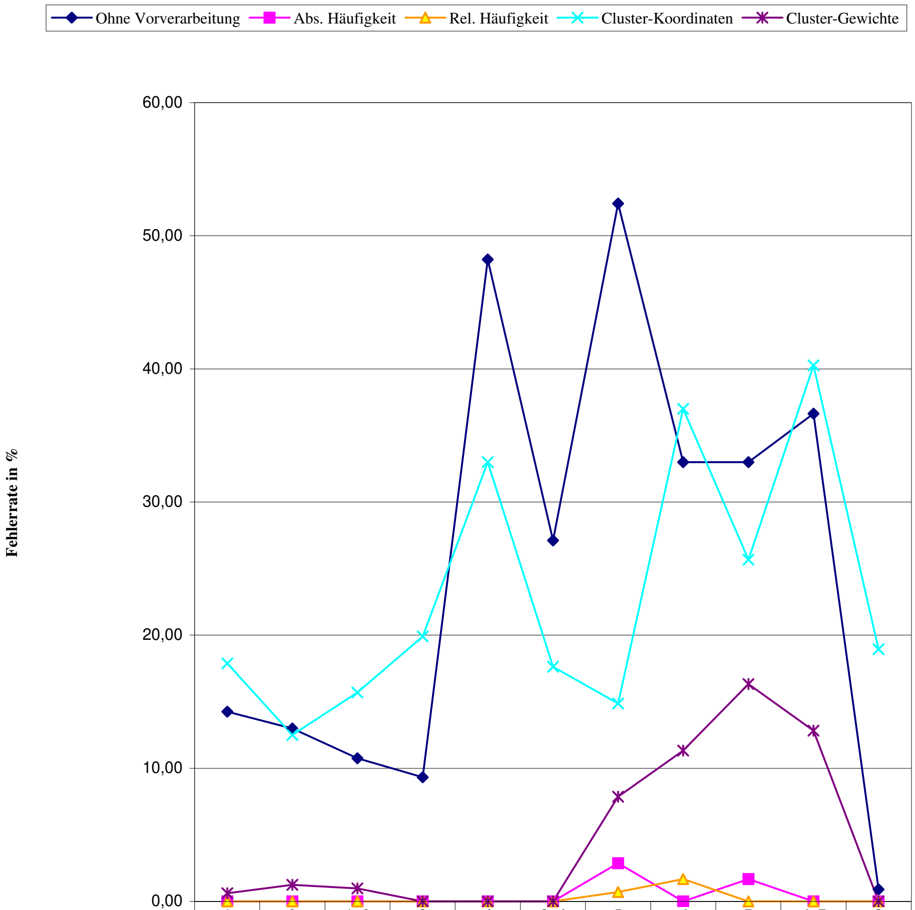
Intervalllänge	Versuchsreihe 8
10 ms	0,00%
100 ms	0,00%
1000 ms	0,00%

Tabelle 5.21: Fehlerrate im Szenario 4 mit dem Verfahren „Gewicht von Clustern in Intervallen“

In Abbildung 5.3 werden die Ergebnisse der Vorverarbeitungsverfahren auf den unterschiedlichen Testreihen zusammengefasst und grafisch dargestellt. Vergleicht man dabei die Fehlerraten der Kreuzvalidierung, so kann man nur die

Verfahren, die den Raum durch die Häufigkeit von Punkten in Intervallabschnitten abbilden, als Methode zur Vorverarbeitung der Daten für das Lernproblem verwenden. Diese Methoden erreichen in beiden vorgestellten Varianten eine so niedrige Fehlerrate, dass die Beispiele fast aller Testdaten nahezu fehlerfrei erkannt werden. Die Variante mit der Bestimmung der relativen Häufigkeit hat gegenüber der anderen Variante eine geringfügig bessere Performanz. Das Verfahren hat zudem den Vorteil, dass es mit allen Klassifikationskriterien eine hohe Vorhersagegenauigkeit erzielt. Um diese Leistung zu erreichen, ist eine hohe Auflösung in der Größenordnung von 10 ms erforderlich. Dadurch wird die Verteilung der Punkte in dem Raum gut wiedergegeben.

Fehlerrate bei der Kreuzvalidierung



	1	2	1+2	3	4	3+4	5	6	7	6+7	8
Ohne Vorverarbeitung	14,25	12,98	10,76	9,33	48,22	27,11	52,43	33,00	33,00	36,64	0,91
Abs. Häufigkeit	0,00	0,00	0,00	0,00	0,00	0,00	2,86	0,00	1,67	0,00	0,00
Rel. Häufigkeit	0,00	0,00	0,00	0,00	0,00	0,00	0,71	1,67	0,00	0,00	0,00
Cluster-Koordinaten	17,88	12,50	15,70	19,89	33,00	17,63	14,86	37,00	25,67	40,27	18,94
Cluster-Gewichte	0,62	1,25	0,97	0,00	0,00	0,00	7,86	11,33	16,33	12,82	0,00

Versuchsreihen

Abbildung 5.3: Vergleich der Verfahren

Die Laufzeit für die Berechnung der relativen Häufigkeiten für die vielen Intervalle bei der kleinen Intervalleinteilung ist sehr niedrig und bei den hier betrachteten Testdaten immer im Bereich unter einer Sekunde. Diese Verzögerung ist im Vergleich zur Laufzeit für das Einlesen der Tracedaten sehr gering. Bei fester Auflösung steigt die Laufzeit mit $O(n \log n)$ an und ist somit auch für große Beispiele anwendbar.

Die Lernmaschine ist dabei mit linearem Kernel in der Lage, die Beispiele klar voneinander zu trennen. Die naheliegende Vermutung, dass hier ein linearer Zusammenhang zugrunde liegt, wird in weiteren Experimenten mit anderen Kernfunktionen genauer untersucht. Bei den folgenden Untersuchungen wird ausschließlich das Verfahren der relativen Häufigkeit in Intervallen der Größe von 10 ms verwendet, um den Lernerfolg zu bewerten.

5.1.2 Umfang der Beispiele

In diesem Kapitel wird untersucht, ob und in wie weit die Größe eines Zeitfensters einen Einfluss auf die Lernergebnisse hat. Bei den zuvor durchgeführten Untersuchungen werden fünf Minuten als ein Beispiel betrachtet, das von der SVM klassifiziert werden kann. Wünschenswert wären allerdings kürzere Zeitfenster, so dass bei einer Echtzeitanalyse schneller auf zurückliegende Ereignisse reagiert werden kann, da man das Ablaufen eines Zeitfensters abwarten muss. Außerdem wird durch ein kleineres Zeitfenster eine bessere Lokalisierung der Ereignisse möglich.

Um das zu untersuchen, wird bei den Versuchsreihen das Zeitfenster halbiert und die Bestimmung der Fehlerrate mit dem Verfahren der relativen Häufigkeit in Intervallen erneut durchgeführt. Bei allen Testreihen müssen dafür die Daten mit einem Zeitfenster von 150 Sekunden erneut eingelesen und vorverarbeitet werden. Durch die Halbierung des Zeitfensters entstehen doppelt so viele Beispiele wie bei den zuvor durchgeführten Experimenten, die dafür halb so viele Daten enthalten.

Versuchsreihe	Zeitfenster von	
	150 Sekunden	300 Sekunden
1	0,00%	0,00%
2	0,31%	0,00%
1+2	0,16%	0,00%
3	0,00%	0,00%
4	0,00%	0,00%
3+4	0,00%	0,00%
5	2,07%	0,71%
6	4,47%	1,67%
7	2,73%	0,00%
6+7	3,14%	0,00%
8	0,00%	0,00%

Tabelle 5.22: Vergleich der Fehlerraten bei unterschiedlichen Zeitfenstern

Die Ergebnisse der gemessenen Fehlerraten werden in Tabelle 5.22 dargestellt und mit den Ergebnissen bei einem Zeitfenster von 300 Sekunden verglichen. Dabei fällt auf, dass die Methode mit großem Zeitfenster bei allen Versuchsreihen eine niedrigere Fehlerrate erzielt als die Methode mit kleinem Zeitfenster. Mit dem kürzeren Zeitfenster werden die Beispiele mit einer etwas höheren Ungenauigkeit vorhergesagt. Ein zu kleines Fenster ist also ungeeignet für das Lösen des Lernproblems. Aufgrund der höheren Erkennungsleistung wird empfohlen, das Zeitfenster auf fünf Minuten beizubehalten.

5.1.3 Skalierung der Zeitachse

Die Skalierung der Zeitachse war bisher immer linear, d.h. es wurde keine Umrechnung der Zeitwerte vorgenommen. Da die Zeitdifferenzen oft nahe bei null sind, ist es wünschenswert, den kleinen Zeitbereich besser aufzulösen, so dass diese Punkte besser voneinander getrennt werden. Dazu werden die Werte auf der Zeitachse, wie in Abschnitt 4.1 beschrieben, logarithmisch skaliert. Die Zeitkomponente der Punkte wird mit der Funktion $f(x) = 2(log_{10}x + 4)$ multipliziert. Der Wertebereich bleibt dabei erhalten.

Es wird nun mit dem bisher besten Verfahren, die Bestimmung der relativen Häufigkeit von Punkten in Intervallen, untersucht, welchen Einfluss eine logarithmische Skalierung auf die Erkennungsleistung hat. Die Umrechnung geschieht bereits während des Einlesens einer Aufzeichnung. Die Tracedaten der emulierten Szenarien werden nun mit logarithmischer Einteilung vorverarbeitet. Die Lernmaschine wird mit diesen Daten neu trainiert und mit dem linearen Fall verglichen.

Versuchsreihe	Skalierung	
	log.	linear
1	0,00%	0,00%
2	0,00%	0,00%
1+2	0,00%	0,00%
3	0,00%	0,00%
4	0,00%	0,00%
3+4	0,00%	0,00%
5	1,40%	0,71%
6	1,67%	1,67%
7	9,67%	0,00%
6+7	4,64%	0,00%
8	0,00%	0,00%

Tabelle 5.23: Vergleich der Fehlerraten bei unterschiedlicher Skalierung

Die Fehlerrate liegt bei allen Versuchsreihen im ersten Szenario bei 0%. Damit wird in diesem Szenario die gleiche Erkennungsleistung erreicht wie in dem linearen Fall. Im zweiten Szenario wird mit der logarithmischen Skalierung, bis auf die fünfte Versuchsreihe, eine ebenfalls fehlerfreie Klassifizierung erreicht. Allerdings fällt die Fehlerrate beim fünften Versuch höher aus als bei der linearen Vorverarbeitung. Die logarithmische Einteilung der Zeitachse bringt auch beim dritten und vierten Szenario keine Verbesserung der Fehlerrate.

Die Ergebnisse der Experimente mit logarithmischer Skalierung der Zeitachse zeigen, dass die Leistung der SVM im linearen Fall nicht übertroffen werden kann. Im Gegenteil, die Erkennungsleistung verschlechtert sich sogar bei einigen Versuchsreihen ganz erheblich. Daher kann dieses Verfahren nicht empfohlen werden. Die Werte im unteren Bereich werden zwar besser aufgelöst, allerdings liegen dadurch die Werte im oberen Bereich näher zusammen. Dies kann der Grund dafür sein, dass die SVM die Beispiele nicht mehr so exakt erkennt wie im linearen Fall. Die beste Methode zur Vorverarbeitung der Daten bleibt also das Verfahren ohne Umrechnung der Zeitachse.

5.1.4 Einstellungen der SVM

Die Experimente mit unterschiedlichen Vorverarbeitungsmethoden zeigen, dass die Methode, die die Verteilung der Punkte durch die Häufigkeit in Intervallen abbildet, am besten für die Vorverarbeitung der Rohdaten geeignet ist. Mit dieser Methode werden auch mit linearem Kernel sehr zufriedenstellende Ergebnisse hinsichtlich Erkennungsgenauigkeit und Rechenaufwand erzielt. In diesem Abschnitt soll der Einfluss der Kernfunktion auf die Erkennungsleistung untersucht werden. Es wird experimentell eruiert, ob und in wie weit die Leistung von der Kernfunktion und der Kapazität abhängt. Die SVM$^{\text{light}}$ V3.50 von Thorsten Joachims bietet die Möglichkeit an, eine andere Entscheidungsfunktion zu benutzen und die Kapazität C zu parametrieren.

Wahl der Kernfunktion

In diesem Abschnitt wird der Einfluss der Kernfunktion auf die Erkennungsleistung der SVM untersucht. Dabei wird ein polynomieller Kernel mit unterschiedlichem Grad d eingesetzt. Die SVM wird mit den bekannten Testreihen der Szenarien neu trainiert. Die Fehlerrate bei der Kreuzvalidierung wird anschließend ermittelt und mit dem linearen Kernel verglichen.

Versuchsreihe	Polynom vom Grad			Lin.
	2	3	4	
1	0,00%	0,00%	0,00%	0,00%
2	0,00%	0,00%	0,00%	0,00%
1+2	0,00%	0,00%	0,00%	0,00%
3	0,00%	0,00%	0,00%	0,00%
4	0,00%	0,00%	0,00%	0,00%
3+4	0,00%	0,00%	0,00%	0,00%
5	2,14%	2,14%	2,81%	0,71%
6	1,67%	1,67%	2,00%	1,67%
7	0,00%	0,00%	0,00%	0,00%
6+7	0,00%	0,00%	0,00%	0,00%
8	0,00%	0,00%	0,00%	0,00%

Tabelle 5.24: Vergleich der Fehlerraten bei unterschiedlichen Kernfunktionen

Bei den ersten vier Versuchsreihen erreicht die SVM mit polynomiellen Kernel, genauso wie die lineare SVM, eine fehlerfreie Klassifikation aller Testbeispiele. Erst bei der fünften Testreihe zeigen sich Unterschiede zum linearen Kernel. Hierbei zeigt sich, dass bei der SVM mit polynomieller Entscheidungsfunktion höhere Werte für die Fehlerrate zu verzeichnen sind. Ebenso sieht es bei der sechsten Testreihe aus. Die Tests mit Polynomen vom Grad 2 und 3 haben dabei identische Werte, wogegen beim Polynom vierten Grades noch schlechtere Werte erzielt werden. Bei den letzten drei Versuchsreihen ist wieder eine fehlerfreie Klassifikation zu beobachten.

Aus den Ergebnissen geht hervor, dass die Leistung einer linearen SVM von einer SVM mit polynomieller Kernfunktion nicht übertroffen werden kann. Es werden bestenfalls gleiche Ergebnisse erzielt. Bei einigen Testdaten werden etwas höhere Fehlerraten gemessen, je größer der Grad des Polynoms wird. Die Kernfunktion hat aber insgesamt keinen hohen Einfluss auf das Lernergebnis. Der polynomielle Kernel erzielt in den meisten Versuchsreihen dieselbe Fehlerrate wie die lineare Entscheidungsfunktion. Die SVM mit polynomieller Kernfunktion hat zudem den Nachteil, dass eine längere Laufzeit für das Lernen notwendig ist. Daher ist es empfehlenswert, für die Untersuchungen eine lineare SVM zu benutzen.

Wahl der Kapazität C

Bei den folgenden Versuchen wird der Einfluss der Kapazität C auf das Lernergebnis der SVM untersucht. Der Parameter wägt eine geringe Komplexität der Entscheidungsfunktion (kleines C) gegen eine hohe Genauigkeit auf der Trainingsmenge (hohes C) ab, d.h. kleine Werte für C vergrößern den Trainingsfehler bei geringer Komplexität. Es wird experimentell versucht gute Einstellungen für diesen Parameter zu finden. Dafür werden die Fehlerraten bei den bekannten Versuchsreihen mit unterschiedlichen Werten für den Parameter C ermittelt.

Versuchsreihe	10^{-3}	10^{-2}	10^{-1}	1	10^1	10^2	10^3
1	0,00	0,00	0,00	0,00	0,00	0,00	0,00
2	1,21	0,00	0,00	0,00	0,00	0,00	0,00
1+2	0,96	0,00	0,00	0,00	0,00	0,00	0,00
3	0,00	0,00	0,00	0,00	0,00	0,00	0,00
4	0,00	0,00	0,00	0,00	0,00	0,00	0,00
3+4	0,00	0,00	0,00	0,00	0,00	0,00	0,00
5	9,95	1,43	0,71	0,71	0,71	0,71	0,71
6	30,33	2,00	2,00	1,67	1,67	1,67	1,67
7	32,33	0,00	0,00	0,00	0,00	0,00	0,00
6+7	24,73	0,00	0,00	0,00	0,00	0,00	0,00
8	0,00	0,00	0,00	0,00	0,00	0,00	0,00

Tabelle 5.25: Vergleich der Fehlerrate bei unterschiedlichen Kapazitäten

Die Ergebnisse der Experimente werden in Tabelle 5.25 dargestellt. Der ursprüngliche Wert des Parameters bei den vorhergehenden Versuchen war eins. Das Erhöhen des Parameters zeigt bei allen Versuchsreihen keine Änderung der Fehlerrate. Bei kleineren Werten wird die Fehlerrate bei einigen Testreihen größer. Auffällig ist, dass der Parameter C bei den Versuchsreihen 3 und 4 des zweiten Szenarios überhaupt keinen Einfluss hat auf die Fehlerrate. Es wird hier bei allen Messungen eine fehlerfreie Klassifikation erzielt. Bei den meisten anderen Testreihen wird eine deutliche Erhöhung der Fehlerrate bei 10^{-3} beobachtet. Bleibt der Wert jedoch über 10^{-2}, so werden identische Fehlerraten gemessen.

Der Parameter C hat keinen großen Einfluss auf das Lernergebnis, wenn er nicht zu klein ist. Werte unterhalb von 10^{-2} sind nicht zu empfehlen, da die Fehlerrate enorm steigt. Eine Verbesserung der Ergebnisse durch Änderung des Parameters C gegenüber dem Ausgangswert von eins ist nicht zu erreichen. Daher wird empfohlen, den Wert für den Parameter auf eins zu setzen.

5.2 Auswertung

Die zahlreichen Experimente mit den Versuchsreihen der vier Szenarien haben gezeigt, dass Lernen aus Beispielen mit der SVM möglich ist. Die Lernmaschine kann nach unterschiedlichen Klassifikationskriterien trainiert werden. Daraus wird eine Klassifikationsregel gelernt, die es ermöglicht unbekannte Beispiele zu klassifizieren. Anhand der von den Kommunikationsgeräten gesendeten Daten können charakterischtische Merkmale im Protokollverhalten erkannt werden. Aus dem gesendeten Datenstrom können Merkmale extrahiert werden, die den Zustand des Kommunikationsverhaltens wiedergeben.

Bei allen Klassifikationskriterien ist es wichtig, die richtige Methode zur Vorverarbeitung der extrahierten Merkmale zu wählen. Die Wahl des Verfahrens hatte den größten Einfluss auf die Leistung der SVM. Ohne Vorverarbeitung der Daten ist Lernen kaum möglich, da die Fehlerrate zu hoch und der Aufwand für das Trainieren nicht praktikabel ist. Bei den Verfahren, die die Verteilung der Punkte im Raum abbilden, werden bezüglich Erkennungsgenauigkeit und Zeitaufwand die besten Ergebnisse erzielt. Dabei erreicht das Verfahren, das die relativen Häufigkeiten von Punkten in Intervallen des Raumes berechnet, die beste Erkennungsleistung. Es muss aber darauf geachtet werden, dass die Auflösung hoch eingestellt wird. Die erzielten Fehlerraten bei einer Intervalleinteilung von 10 ms sind dann nahe 0%, und die Laufzeiten für das Vorverarbeitungsverfahren und für den Lernalgorithmus sind sehr kurz.

Die Fuzzy-Clusteranalyse ist dagegen nicht gut für die Vorverarbeitung zum Lernen geeignet, da die erzielten Fehlerraten nicht den hohen Aufwand rechtfertigen. Das Clustering-Verfahren ist jedoch anwendbar, um einzelne Beispiele näher zu beobachten und zu analysieren. Die SVM ist dagegen nur in der Lage unbekannte Beispiele zu klassifizieren, wobei die Clusteranalyse bei der Suche nach der Ursache für diese Klassifikation behilflich ist. Beide Verfahren zusammen dienen der Analyse der aufgezeichneten Daten.

Die Einstellungen der anderen Parameter beeinflussen das Lernergebnis nur gering. Bei der SVM hat die Kernfunktion und die Kapazität C nur geringen Einfluss auf das Ergebnis der Lernmaschine. Ein linearer Kernel bei $C = 1$ reicht für eine hohe Erkennungsleistung aus. Die empfohlene Konfiguration für

Umfang der Beispiele:	5 Minuten
Skalierung der Zeitachse:	linear
Vorverarbeitung:	Relative Häufigkeit in Intervallabschnitten
Intervalleinteilung:	10 ms
Kernfunktion der SVM:	linear
Parameter C:	1

Tabelle 5.26: Optimale Konfiguration des Analyseverfahrens

das Diagnoseverfahren wird in Tabelle 5.26 zusammengefasst.

Kapitel 6

Ausblick

Die Untersuchungen der vorliegenden Arbeit haben gezeigt, dass es möglich ist, das Kommunikationsverhalten von Telekommunikationsanlagen mit wenigen Informationen zu repräsentieren und damit Muster im Verhalten von Kommunikationssystemen zu erkennen. Die Repräsentation eines Ausschnitts von fünf Minuten des gesendeten Datenstroms und die Vorverarbeitung mit Hilfe der Häufigkeit spiegelt das charakteristische Kommunikationsverhalten der Anlagen wider. Die durchgeführten Versuche zeigen, dass die Support Vector Machine mit der vorgestellten Datenrepräsentation und -vorverarbeitung in der Lage ist, unbekannte Beispiele mit einer sehr niedrigen Fehlerrate zu erkennen. Dabei kann die SVM das Protokollverhalten nach verschiedenen Kriterien klassifizieren. Die Lernmaschine erkennt signifikante Änderungen im Verhalten von Kommunikationssystemen. Für die automatische Klassifikation von aufgezeichneten Protokolldaten kann die SVM herangezogen werden. Bei den Experimenten wurde ausschließlich die binäre Klassifikation untersucht. Um genauere Aussagen bei der Diagnose zu erhalten, könnte dieses Verfahren auf eine Multi-Klassen-Klassifikation erweitert werden, indem mehrere Lernmaschinen eingesetzt werden, die binär klassifizieren können.

Die SVM ist zwar in der Lage Muster zu erkennen und unbekannte Beispiele zu klassifizieren, man erhält jedoch keine direkten Informationen über die Ursachen. Eine genaue Analyse der von der SVM ermittelten trennenden Hyperebene könnte bei der Suche nach der Ursache behilflich sein. Es könnte untersucht werden, welche Merkmale die Hyperebene festlegen und wo die Schwelle zwischen den beiden Klassen ist. Bei der Analyse der Ursache kann auch eine Fuzzy-Clusteranalyse eingesetzt werden, mit der man einzelne Beispiele genauer beobachten und untersuchen kann. Beide Lernverfahren zusammen dienen als intelligente Verfahren, um Probleme im Protokollverhalten zu finden und zu diagnostizieren: Mit Hilfe der SVM lässt sich ein fehlerhaftes bzw. seltsames Verhalten lokalisieren und die Fuzzy-Clusteranalyse dient zur Diagnose für einen Experten. Bei der Bewertung einer Clustereinteilung ist noch eine computerunterstützte Hilfe denkbar, die jeden einzelnen Cluster bewertet und auf merkwürdige Cluster aufmerksam macht.

Die Daten der durchgeführten Experimente wurden jeweils nach der Aufzeichnung mit den Verfahren analysiert. Dies ist zum Trainieren der SVM und zur Ermittlung der Leistung durchführbar, allerdings ist eine Diagnose auch während der Aufzeichnung wünschenswert. So könnte unmittelbar nach dem

Ablauf eines Zeitfensters, dieses von der SVM klassifiziert werden. Wird dabei ein Zeitfenster negativ klassifiziert, so könnte eine Warnung ausgegeben werden. Die Analyse könnte damit als Frühwarnsystem im laufenden Betrieb eingesetzt werden.

Auch während der Entwicklung kann die Diagnostik benutzt werden, um signifikante Änderungen von neuen Softwarereleases zu erkennen. Der Aufwand für den Test des stabilen Betriebs einer Anlage kann somit durch Automatisierung der Testabläufe verringert werden. Dies reduziert die Kosten der Qualitätssicherung und verkürzt die Testphasen. Softwarereleases können damit schneller auf den Markt gebracht werden.

Allerdings hat diese Diagnostik ihre Grenzen. Hier werden nur die Rahmen auf Schicht 2 analysiert und das Antwortverhalten beobachtet. Bei dieser Diagnostik können jedoch einige Fehler nicht erkannt werden. So kann es vorkommen, dass in Schicht 2 alles korrekt abläuft, aber in Schicht 3 Fehler auftreten.

Diese Art der Diagnostik von Telekommunikationsanlagen könnte auch auf andere Protokolle und Netze übertragen werden. So könnte das Verfahren auch auf den Informationsaustausch auf Schicht 3 im ISDN angewendet werden, wobei allerdings mehr Informationselemente auftreten als im LAP-D-Protokoll. Daher wird es in dem dreidimensionalen Raum mehr Zeitachsen geben, auf denen sich Punkte befinden können. Das Prinzip der Analyse bleibt jedoch gleich.

Auch in Computernetzen gibt es einen Informationsaustausch, den man mit dieser Methode analysieren könnte. Hierfür ist allerdings noch Forschungsarbeit notwendig, um zu untersuchen, ob auch hier dieses Verfahren eingesetzt werden kann. Die Repräsentation und Vorverarbeitung ist in dieser Arbeit speziell auf die Diagnose im LAP-D-Protokoll im ISDN ausgerichtet. Eine Übertragung des Verfahrens auf andere Protokolle wird sicherlich eine Anpassung der Repräsentation und der Vorverarbeitung der Daten zu Folge haben.

Literaturverzeichnis

[1] Aizerman, M.A. ; Braverman, E.M. ; Rozoner, L.I.: Theoretical foundations of the potential function method in pattern recognition learning. In: *Automation and Remote Control*. 1964, Kapitel 25, S. 821–837

[2] Barnutz, Matthias: *Klassifikation und Bewertung des Verhaltens von Telekommunikationsanlagen mit Methoden der künstlichen Intelligenz*. Dortmund, Universität Dortmund, Fachbereich Elektrotechnik, Diplomarbeit, 2001

[3] Boser, B.E. ; Guyon, I.M. ; Vapnik, Vladimir N.: A training algorithm for optimal margin classifiers. In: Haussler, D. (Hrsg.): *Fifth Annual Workshop on Computational Learning theory*. Pittsburgh, ACM, 1992, S. 144–152

[4] Burges, Christopher J. C.: A Tutorial on Support Vector Machines for Pattern Recognition. In: *Data Mining and Knowledge Discovery 2*. 1998, S. 121–167

[5] Christianini, Nello ; Shawe-Taylo, John: *An Introduction to Support Vector Machines*. Cambridge : Cambridge University Press, 2000

[6] Cortes, C. ; Vapnik, Vladimir N.: Support vector networks. In: *Machine Learning*. 1995, Kapitel 20, S. 273–297

[7] Courant, Richard ; Hilbert, David: *Methods of Mathematical Physics*. New York : Wiley, 1953

[8] ETSI: *ETS 300 125: User-network interface data link layer specification. Application of ITU-T Recommendations Q.920/I.440 and Q.921/I.441*. ETSI, September 1991

[9] ETSI: *ETS 300 102: ISDN - User-network interface layer 3 - Specifications for basic call control*. Sophia Antipolis: ETSI, 1993

[10] ETSI: *ETS 300 403-1: ISDN - Digital Subscriber Signalling System No. one (DSS1) protocol - Signalling network layer for circuit-mode basic call control - Part 1 Protocol specification*. Sophia Antipolis: ETSI, 1995

[11] Flecther, R.: *Practical Methods of Optimization*. 2nd ed. Chichester : Wiley, 1987

[12] Hajer, Hans ; Kolbeck, Rainer: *ISDN: Einführung und Überblick*. Haar bei München : Markt und Technik, 1994

[13] HÖPPNER, Frank: *Fuzzy Clustering Algorithms - A Tool Library*, 2000. – http://www.fuzzy-clustering.de

[14] HÖPPNER, Frank ; KLAWONN, Frank ; KRUSE, Rudolf: *Fuzzy-Clusteranalyse*. Braunschweig, Wiesbaden : Vieweg, 1997

[15] ISO: *ISO 7498: Information processing systems - Open Systems Interconnection - Basic Reference Model*. ISO, 1984

[16] ITU-T: *Q.920: ISDN user-network interface data link layer - General aspects*. Genf: ITU, 1988

[17] ITU-T: *ISDN user-network interface layer 3 - Specification for basic call control (Rec. Q.931)*. Genf: ITU, 1993

[18] ITU-T: *Q.921: ISDN user-network interface data link layer specification*. Genf: ITU, 1993

[19] JOACHIMS, Thorsten: Making Large-Scale SVM Learning Practical. In: SCHÖLKOPF, Bernhard (Hrsg.) ; BURGES, Christopher J. C. (Hrsg.) ; SMOLA, Alexander J. (Hrsg.): *Advances in Kernel Methods - Support Vector Learning*. Cambridge, USA : MIT Press, 1998, Kapitel 11

[20] JOACHIMS, Thorsten: Estimating the Generalization Performance of a SVM Efficiently. In: *Proceedings of the Seventeenth International Conference on Machine Learning*. San Francisco : Morgan Kaufman, 2000

[21] KANBACH, Andreas ; KÖRBER, Andreas: *ISDN - Die Technik*. 3.Aufl. Heidelberg : Hüthig, 1999

[22] LUNTS, A. ; BRAILOVSKIY, V.: Evaluation of attributes obtained in statistical decision rules. In: *Engineering Cybernetics* 3 (1967), S. 98–109

[23] MICHALSKI, Ryszard: Understanding the Nature of Learning. In: MICHALSKI, R.S. (Hrsg.) ; CARBONELL, J.G. (Hrsg.) ; MITCHELL, T.M. (Hrsg.): *Machine Learning - An Artificial Intelligence Approach*. Los Altos, California : Morgan Kaufmann, 1986

[24] MORIK, Katharina: Maschinelles Lernen. In: GÖRZ, Günther (Hrsg.): *Einführung in die künstliche Intelligenz*. Bonn, Paris : Addison-Wesley, 1995, S. 243–297

[25] NOER, Geoffrey J.: *Cygwin: A Free Win32 Porting Layer for UNIX® Applications*. Red Hat, 1998. – http://sources.redhat.com/cygwin/

[26] SCHÖLKOPF, Bernhard: *Support Vector Learning*. München : Oldenbourg, 1997

[27] SCOTT, P.D.: Learning: The Construction of a Posteriori Knowledge Structures. In: *AAAI-83*. Washington, 1983

[28] SEGAL, Mark ; AKELEY, Kurt: *The OpenGL® Graphics System: A Specification, Version 1.3*. Silicon Graphics, Inc., 2001. – http://www.opengl.org

[29] Simon, H.A.: Why Should Machines Learn? In: Michalski, R.S. (Hrsg.) ; Carbonell, J.G. (Hrsg.) ; Mitchell, T.M. (Hrsg.): *Machine Learning: An Artificial Intelligence Approach.* Tioga, Palo Alto, CA, 1983, S. 25–38

[30] Vapnik, Vladimir N.: *The nature of statistical learning theory.* 2. ed. New York : Springer, 2000

[31] Xie, X.L. ; Beni, G.: A Validity Measure for Fuzzy Clustering. In: *IEEE Trans. Pattern Analysis and Machine Intelligence.* 13. 1991, S. 841–847

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