Lehrstuhl für Mensch-Maschine-Kommunikation
Technische Universität München

Multimodale Modellierung von Gruppenaktionen zur Segmentierung von Besprechungen

Stephan A. Reiter

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktor-Ingenieurs (Dr.-Ing.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr.-Ing. Eckehard Steinbach
Prüfer der Dissertation:	1.	Univ.-Prof. Dr.-Ing. habil. Gerhard Rigoll
	2.	Univ.-Prof. Dr.-Ing. Fernando Puente León

Die Dissertation wurde am 24.10.2007 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik am 24.06.2008 angenommen.

Vorwort

Die vorliegende Arbeit ist im Laufe meiner Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Mensch-Maschine-Kommunikation der Fakultät für Elektro- und Informationstechnik an der Technischen Universität München entstanden.

Mein ganz besonderer Dank gilt meinem Doktorvater Univ.-Prof. Dr.-Ing. habil. Gerhard Rigoll für die Möglichkeit der Bearbeitung der vorgestellten Themen, für die wissenschaftliche Betreuung und die fruchtbaren Anregungen, die erheblich zum Gelingen dieser Arbeit beigetragen haben.

Ebenso gilt mein Dank an dieser Stelle Univ.-Prof. Dr.-Ing. Fernando Puente León vom Fachgebiet Verteilte Messsysteme an der Technischen Universität München für die Übernahme des Zweitgutachtens und dem damit verbundenen Aufwand.

Ferner danke ich den Mitarbeitern des Lehrstuhls für Mensch-Maschine-Kommunikation für das gute Arbeitsklima und die stets vorhandene Unterstützung und Diskussionsbereitschaft. Besonderer Dank gebührt dabei Dr.-Ing. Björn Schuller, Dipl.-Ing. Marc Al-Hames, Dipl.-Ing. Benedikt Hörnler und Dipl.-Ing. Joachim Schenk für das Korrekturlesen der Arbeit. An Peter Brand, Heiner Hundhammer und Ernst Ertl ein Dankeschön für die stets einsatzbereite Infrastruktur.

Für die Motivation und Unterstützung möchte ich ganz besonders meiner Frau Claudia und meiner Familie danken.

München, im September 2007

Stephan A. Reiter

Kurzfassung

Geschäftliche Besprechungen nehmen einen immer größeren Zeitraum im Berufsleben ein. Vor diesem Hintergrund werden in dieser Arbeit innovative Verfahren vorgestellt, eine aufgezeichnete Besprechung automatisch zu analysieren und in sinnvolle Abschnitte zu untergliedern. Diese Abschnitte beschreiben Gruppenaktionen, die das Verhalten aller Teilnehmer einer Besprechung als Gruppe modellieren. Dies ist ein wichtiger Teilaspekt für die Analyse der Interaktion von Teilnehmern eines Meetings und stellt somit einen wesentlichen Beitrag zum Forschungsgebiet der „Human-to-Human Communication“ dar, bei dem massiv Methoden der Mensch-Maschine-Kommunikation zum Einsatz kommen.
Für die Modellierung werden zunächst statische Klassifikationsverfahren eingesetzt, darunter Neuronale Netze, Bayessche Netze und Support-Vektor-Maschinen. Die Segmentierung erfolgt hierbei mit Hilfe zweier in dieser Arbeit neu entwickelter Verfahren.
Rekurrente Neuronale Netze, erweitert durch Long Short-Term Memory Zellen, zeigen eindrucksvoll ihre Mächtigkeit bei der Klassenzuordnung definierter Segmente einer Besprechung. Eine aus der Theorie der Neuronalen Felder neu abgeleitete Architektur eines Parallelen Rekurrennten Neuronalen Netzes wird eingesetzt, um eine parallele Verarbeitung aller Daten einer Besprechung zu ermöglichen.
Um die dynamischen Eigenschaften einer Besprechung zu verarbeiten, werden sowohl Hidden-Markov-Modelle als auch erstmals Hidden Conditional Random Fields für diese Aufgabe verwendet. Ein diskriminatives Training hilft dabei, die Modelle besser an die Trainingsdaten anzupassen.
Dem Problem der Spärlichkeit der Daten wird mit hybriden und zweistufigen Modellen begegnet. Dabei werden mehrere unterschiedliche Klassifikationsansätze vereint. Durch den Einsatz des in dieser Arbeit vorgeschlagenen gleitenden Mittelwertfilters kann das Ergebnis weiter verbessert werden. Die zweistufigen Ansätze modellieren zunächst Aktionen einzelner Besprechungsteilnehmer; in einem zweiten Schritt erfolgt die eigentliche Segmentierung der Besprechung mit Hilfe von HMM.
Als Ergebnis dieser Forschungstätigkeit können Besprechungen unter Vorliegen bestimmter Rahmenbedingungen zuverlässig in sinnvolle Einheiten für eine weitergehende Verarbeitung unterteilt werden.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die automatische Verarbeitung und Analyse von geschäftlichen Besprechungen rückt immer mehr in den Vordergrund wissenschaftlicher Forschungsarbeiten [Wai01], [Mor01], [Cut02], [McC03], [Hil03]. Grund dafür ist die schier überwältigende Anzahl von Meetings, die in Firmen oder Forschungseinrichtungen abgehalten werden. Bei der Firma Intel beispielsweise sind die Besprechungsräume pro Jahr für drei Millionen Stunden gebucht, vier Millionen Stunden werden für Schulungsmaßnahmen für Mitarbeiter verwendet und 5,7 Millionen Telefonkonferenzen werden pro Jahr abgehalten [Hou05]. Ein Ansatz, diese Flut von Besprechungen in den Griff zu bekommen, sind Schulungen, wie Meetings effektiver abgehalten werden können. Dafür werden bei der Firma Intel etwa 56.000 Stunden pro Jahr aufgewendet [Hou05].
Umfassende Untersuchungen von Romano und Nunamaker [Rom01] zeigen, dass Meetings einen wesentlichen Bestandteil des Wirtschaftslebens darstellen. Ihren Analysen zufolge verbringen leitende Angestellte etwa 25 % ihrer Arbeitszeit in Besprechungen, Topmanager sogar bis zu 80 %. Die Zeit, die für Meetings aufgewendet wird ist in den letzten Jahren angestiegen und wird Umfragen zufolge weiter zunehmen [Rom01]. Die daraus entstehenen Kosten sind nicht zu unterschätzen. In einem Hightech-Unternehmen werden die Kosten der Besprechungen auf 100 Millionen Dollar geschätzt. Aufgrund der Ineffizienz der Meetings geht man von einem Verlust von etwa 54 Millionen Dollar pro Jahr aus [Rom01]. Die Größenordnungen dieser Schätzungen liefern die Grundlage, darüber nachzudenken, auf welche Art und Weise Meetings effektiver gestaltet werden können.
Es wird erwartet, dass technische Systeme in Zukunft verstärkt in Meetings eingesetzt werden [Bou06], [Per06]. Besprechungen können meist nicht effektiver abgehalten werden, da dies sehr stark von den Teilnehmern abhängt. Es werden jedoch effektive Systeme entwickelt, die zumindest eine automatische Übersicht über ein abgehaltenes Meeting geben können. Im Idealfall wird eine kurze prägnante schriftliche Zusammenfassung über den Inhalt oder ein kurzer Videofilm erstellt, der die wichtigsten Ereignisse treffend zusammenfasst. Dadurch wird es einem Mitarbeiter einer Firma, der ein Meeting versäumt hat, leichter gemacht, möglichst schnell auf den neuesten Stand der Entwicklungen zu kommen.
In den letzten Jahren wurden mehrere Forschungsprojekte ins Leben gerufen, die sich mit dieser automatischen Aufbereitung und Verarbeitung von Besprechungen beschäftigen:

- Das europäische Forschungsprojekt M4 (MultiModal Meeting Manager)[1] [Ren02],

[1] EU FP5-IST Programm IST-2001-34485, http://www.dcs.shef.ac.uk/spandh/projects/m4/index.html

in dessen Kontext auch diese Arbeit entstanden ist, hat das Ziel, ein System zu entwickeln, mit dem archivierte Besprechungen strukturiert, durchsucht und angezeigt werden können, nachdem sie automatisch analysiert wurden. Voraussetzung hierfür ist natürlich eine geeignete Aufzeichnung der Bild- und Toninformationen während eines Meetings. Aufgrund der Tatsache, das die Auswertealgorithmen nicht in Echtzeit ausführbar sind und die Analyse erst nach Beendigung eines Meetings initiiert wird, spricht man von einem sogenannten *off-line*-System. Ein *on-line*-System hätte den Vorteil, dass schon während der Besprechung automatische Analysen durchgeführt werden, auf deren Ergebnisse auch sofort zugegriffen werden könnte.

- Das auf M4 aufbauende europäische Forschungsprojekt AMI (Augmented Multi-Party Interaction)[2] [Ren05] wählt einen umfassenderen Ansatz und beschäftigt sich mit der menschlichen Interaktion und der Modellierung derselben in Besprechungen. Das Ziel ist unter anderem die Generierung von automatischen Zusammenfassungen durch Abstraktion von Besprechungen.

- Das Folgeprojekt AMIDA (Augmented Multiparty Interaction with Distance Access)[3] erweitert die in AMI eingeschlagenen Richtungen um Echtzeitfähigkeit und verteilte Teilnehmer.

- Das NIST Meeting Room Project[4] stellt das Pendant zu den europäischen Projekten auf dem amerikanischen Kontinent dar und veranstaltet regelmäßig Evaluationen von für Meetings relevanten Technologien. Neben der Gewinnung von geeignetem Datenmaterial wird versucht, formelle von informellen Besprechungen zu unterscheiden, das Besprechungsziel zu benennen und die Art des Informationsflusses zu bestimmen. Aufbauend auf dieses Projekt sollen zwei ergänzende Vorhaben die Abstraktion und die Einbindung räumlich entfernter Personen untersuchen.

- Ein schon seit mehreren Jahren existierendes Projekt zur Erforschung unterschiedlicher Aspekte in Meetings wird an der Carnegie Mellon University, Pittsburgh in Zusammenarbeit mit der Universität Karlsruhe verfolgt: Das ISL Meeting Room System[5] [Sch01a]. Hier beschäftigt man sich unter anderem damit, automatische Transkriptionen zu erstellen, die Aufmerksamkeit der Teilnehmer zu beobachten und viele andere Aspekte von Meetings zu analysieren. Bei diesem Projekt existiert bereits ein Meeting-Browser, um archivierte Besprechungen durchsuchen zu können.

- Beim International Computer Science Institute (ICSI) in Berkely[6] beschränkt man sich bei der Analyse von Besprechungen auf die Audiodaten. Die gesprochene Sprache wird automatisch online transkribiert, wobei jeder Teilnehmer auf den erkannten Text Einfluss nehmen kann.

[2] EU FP6-IST Programm IST-2002-506811, http://www.amiproject.org
[3] EU FP6-IST Programm IST-2005-2.5.7, http://www.amidaproject.org
[4] http://www.nist.gov/speech/test_beds/mr_proj
[5] http://penance.is.cs.cmu.edu/meeting_room
[6] http://www.icsi.berkeley.edu/Speech/mr/mtgrcdr.html

1.1 Motivation

Während einer Besprechung ist die Sprache das Kommunikationsmittel, das überwiegend zum Einsatz kommt. Aus diesem Grund sind sprachbasierte Technologien, wie Spracherkennung, Sprechererkennung, Themenerkennung und Dialogmodellierung, im Kernfokus der Forschung im Meetingumfeld [Kub99], [Mor01], [Wai01]. Aber auch die Verarbeitung visueller Informationen, wie das Verfolgen von Personen oder des Aufmerksamkeitsfokus, gewinnt an Bedeutung [Wal04c], [Wai03], [Cut02]. Auch weitergehende Analysen von geschriebenem Text, individuellen Gesten und emotionalen Zuständen können im Rahmen der Verarbeitung von Besprechungen wichtig werden [Zob03], [AH05a]. Das Meeting-Umfeld bietet darüber hinaus weitere Aufgaben, welche die Verarbeitung von Ton-, Bild- oder multimodalen Informationen erfordern.

Viele Arbeiten beschäftigen sich damit, das Verhalten und die Informationen einzelner Personen zu analysieren. Dabei wird jedoch häufig vernachlässigt, dass gerade bei Besprechungen das Verhalten in einer Gruppe und der Gruppe der Teilnehmer als ganzes von besonderer Bedeutung sein kann. Das übergeordnete Ziel einer automatischen Besprechungsanalyse ist nicht unbedingt die Erstellung einer akkuraten Sprachtranskription, die Identifizierung der Teilnehmer oder die Erkennung visueller Gesten, sondern vor allem die Zusammenfassung eines Meetings in eine Abfolge von übergeordneten Agendapunkten [McC05]. Diese Zusammenfassung auf höherer semantischer Ebene beschreibt die Aktivität der Gruppe als ganzes, anstatt nur einzelne Personen zu beleuchten. Der Grund dafür ist die Annahme, dass die wichtigen Informationen im Zusammenspiel und der Interaktion der Teilnehmer ausgetauscht werden.

Die automatische Analyse der Interaktion von Menschen ist eine weites Forschungsgebiet. Abgesehen von Besprechungen besteht ein wachsendes Interesse im automatischen Verstehen von Gruppenverhalten [Joh98], [Jeb99], [Oli00]. Die meisten Arbeiten konzentrieren sich überwiegend auf die visuelle Information, zum Beispiel bei der Überwachung von öffentlichen Plätzen [Oli00], oder am Arbeitsplatz [Joh98], [Jeb99]. Die Haupthypothese in diesen Arbeiten besteht darin, dass das Verhalten eines Einzelnen bei Aktivitäten in einer Gruppe durch das Verhalten der Anderen stark beeinflusst wird. Modelliert man also die Beeinflussungen der Personen untereinander, so beschreibt man zugleich auch die Interaktionen.

Während die automatische Analyse multimodaler Gruppenaktivitäten in Besprechungen erst in jüngerer Zeit[7] in den Fokus der Forschung gerückt ist, wird das Gruppenverhalten seit mehr als 50 Jahren von Psychologen studiert [Bal51], [McG84], [McG82]. Für die Entwicklung von Technologien zur automatischen Analyse von Besprechungen können aus diesen Arbeiten wertvolle Hinweise entnommen werden, zum Beispiel bezüglich der Mechanismen des Sprecherwechels [Fay00].

Auch im Bereich der Strukturierung von Besprechungen existieren zahlreiche Ansätze in der Sozialpsychologie. Generell wird eine Kategorisierung des Gruppenverhaltens vorgenommen. Die definierten Kategorien schließen sich gegenseitig aus, sind aber vollständig.

[7] Projekte zur automatischen Analyse von Besprechungen existieren erst etwa seit dem Jahr 2000

Das heißt, für jeden Zeitabschnitt einer Besprechung kann genau eine Kategorie definiert werden, und die gesamte Besprechung wird abgedeckt. Man kann zum Beispiel eine prozessorientierte Einteilung vornehmen, wie sie von Bales [Bal51] vorgeschlagen wird, oder eine aufgabenorientierte Einteilung, wie von McGrath [McG84] angeregt.

Es ist offensichtlich, dass das Wechseln der Sprecher hauptsächlich durch den auditiven Kanal charakterisiert ist. Jedoch können auch visuelle Einflüsse, wie Gesichtsausdrücke, Gesten, usw., zu einem besseren Verständnis der Gruppenaktionen und -interaktionen beitragen. Deshalb erscheint es sinnvoll, einen multimodalen Ansatz zur automatischen Analyse von Besprechungen zu wählen.

In jüngster Zeit sind, wie erwähnt, mehrere Systeme entstanden, die sich mit der automatischen Analyse von Meetings beschäftigen. Da jedoch die Techniken zur automatischen Generierung von schriftlichen Zusammenfassungen oder Überblicksvideos, d.h. Kurzzusammenschnitte, gerade aktueller Gegenstand der Forschung sind, beschränkt man sich meist darauf, Anwendungen zu entwickeln, die ein möglichst effektives Betrachten von Besprechungen ermöglichen. Mit Hilfe dieser sogenannten Meeting-Browser kann man sich sehr schnell einen guten Überblick über ein archiviertes Meeting verschaffen. Tucker [Tuc05] gibt einen guten Überblick über bereits vorhandene Meeting-Browser. Darin werden auch die Probleme und Möglichkeiten aufgezeigt, die in einem solchen System liegen. Der Meeting-Browser, in dem Teile dieser Arbeit eingebunden wurden und werden, ist der Ferret-Browser [Wel05], der in Zusammenarbeit mit internationalen Forschungseinrichtungen im Zuge des EU-Forschungsprojekts M4 entstand. Dieser ist in Abbildung 1.1 zu sehen.

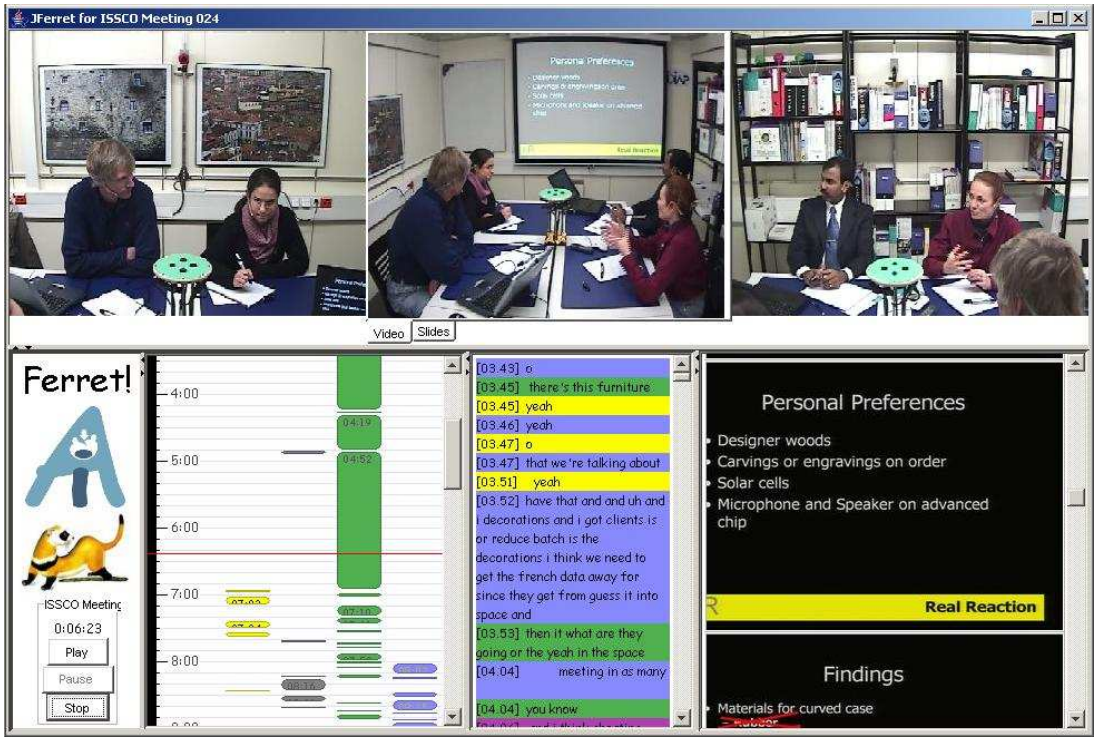


Abbildung 1.1: *Bildschirmfoto des Ferret-Meeting-Browsers zum effizienten Betrachten von aufgezeichneten Besprechungen*

Mit diesem Meeting-Browser ist es möglich, wichtige Ausschnitte einer aufgezeichneten Besprechung anzusehen, bzw. nur die Abschnitte eines Meetings anzusehen, die für einen Betrachter von Interesse sind. Dazu können beispielsweise einzelne Sprecher, Präsentationsfolien oder Gruppenaktionen direkt angewählt werden.

1.2 Beiträge der Arbeit

Die vorliegende Arbeit beschäftigt sich mit der Entwicklung von neuartigen Verfahren zur automatischen Unterteilung von Besprechungen in bedeutungstragende Einheiten von mehreren Sekunden Länge. Der Hauptaugenmerk liegt dabei im Vergleich unterschiedlicher Möglichkeiten der Segmentierung und Klassifizierung von Meetingdaten. Werden im Allgemeinen für die Analyse von Zeitreihen dynamische Klassifikatoren verwendet, so werden in dieser Arbeit auch Methoden zur Klassifizierung statischer Muster analysiert, die neue Arten der Segmentierung notwendig machen. Zudem werden auch neue dynamische Ansätze untersucht, die in dieser Form erstmals im Bereich der Meetinganalyse eingesetzt werden. Hybride Ansätze und zweistufige Verfahren, die die Vorteile verschiedener Klassifikatoren vereinen, werden als mächtiges Werkzeug für die gestellte Aufgabe vorgestellt.
Ein beispielhaftes Ergebnis einer Segmentierung einer Besprechung mit einer Erkennung der zugehörigen Gruppenaktionen ist in Abbildung 1.2 gezeigt.

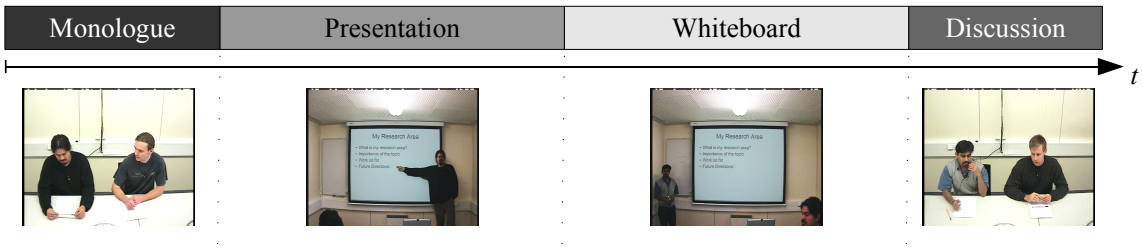


Abbildung 1.2: *Beispiel einer automatisch erfolgten Segmentierung einer Besprechung mit Erkennung der zugehörigen Gruppenaktionen. Für jede erkannte Gruppenaktion ist ein Bildschirmfoto einer typischen Kamerasicht gezeigt.*

1.3 Aufbau der Arbeit

In **Kapitel 2** werden zunächst die verwendeten Datenbanken vorgestellt. Danach wird auf die Generierung der Merkmale, die für die automatische Analyse eingesetzt werden, eingegangen. Abschließend werden die verwendeten Maße zur Evaluierung der Güte der vorgestellten Algorithmen beschrieben.
Kapitel 3 stellt zunächst Klassifikatoren für statische Muster vor, die als Grundlage für zwei neuartige Segmentierungsalgorithmen dienen, die anschließend präsentiert werden. Den Abschluss dieses Kapitels bilden die Ergebnisse, die mit diesen Ansätzen erreicht werden konnten.

Im darauffolgenden **Kapitel 4** werden Modelle vorgestellt, die auf biologischen Systemen beruhen. Insbesondere die neurologische Signalverarbeitung bildet die Grundlage für neue, in diesem Abschnitt vorgestellte, Klassifikatoren. Im Speziellen sind das Long Short-Term Memory Neuronale Netze und ein aus der Theorie der Neuronalen Felder abgeleitetes paralleles rekurrentes neuronales Netz. Auch hier werden wieder Ergebnisse der Anwendung auf die Gruppenaktionserkennung gezeigt.
In **Kapitel 5** wird die Analyse von Besprechungen mit Hilfe von dynamischen Klassifikatoren angegangen. Hier werden Hidden-Markov-Modelle und Hidden Conditional Random Fields vorgestellt. Als Abschluss dieses Kapitels werden Ergebnisse der Gruppenaktionserkennung, die mit dem Viterbi-Algorithmus erhalten werden, vorgestellt.
Kapitel 6 beschreibt hybride Verfahren zur Erkennung von Gruppenaktionen, die in dieser Arbeit entwickelt wurden. Ausgehend von einem MLP-HMM Klassifikator werden zweistufige Ansätze dargestellt, die verschiedene statische und dynamische Klassifikatoren kombinieren. Die Ergebnisse dieser neuartigen Methoden werden am Ende dieses Kapitels gezeigt.
Den Abschluss dieser Arbeit bildet das **Kapitel 7**, in dem die vorgestellten Ansätze zusammengefasst werden und ein Ausblick auf weiterführende Forschungsarbeiten aufgezeigt wird.

Kapitel 2

Datenbanken und Merkmalsextraktion

Für die in dieser Arbeit beschriebenen Experimente zur Segmentierung und Erkennung von Gruppenaktionen steht der im Rahmen des europäischen Forschungsprojekte M4 (MultiModal Meeting Manager) speziell aufgezeichnete Meeting-Corpus zur Verfügung, der im Folgenden beschrieben wird. Anschließend wird auf die aus den Daten extrahierten Merkmale eingegangen, und die zur Evaluierung verwendeten Bewertungsmaße werden vorgestellt.

2.1 Die M4-Datenbank

Die M4-Datenbank besteht aus vollständig aufgezeichneten Besprechungen, die in einem speziell dafür ausgestatteten Besprechungsraum erfasst wurden. Diese Datenbank ist öffentlich frei zugänglich und kann im Internet unter der Adresse: `mmm.idiap.ch` heruntergeladen werden. Eine Beschreibung ist bei McCowan [McC03] zu finden.

2.1.1 Konferenzraum

Im Forschungsinstitut IDIAP[1] in Martigny in der Schweiz wurde ein Konferenzraum eingerichtet, der speziell für die automatische Erfassung von Besprechungen ausgestattet ist [Moo02]. Er ist in Abbildung 2.1 gezeigt.
Dieser rechteckige Raum hat die Maße 8,2 m × 3,6 m × 2,4 m. In der Mitte des Raumes befindet sich ein rechteckiger Konferenztisch der Größe 4,8 m × 1,2 m, der Platz für bis zu zwölf Personen bietet. An einem Ende des Raumes befindet sich ein Whiteboard und eine ausfahrbare Leinwand. An dem anderen Ende des Raumes wurde die Technik, die für die Aufzeichnung der Besprechungen notwendig ist, installiert.
Einen Überblick über den Aufbau und die Platzierung der Sensoren und Besprechungsteilnehmer gibt Abbildung 2.2. Dieser Besprechungsraum ist mit Sensoren ausgestattet, die es ermöglichen, eine Besprechung vollständig aufzuzeichnen. Dazu gehören drei fest angebrachte Kameras und 24 Mikrophone [Moo02], deren Signale gleichzeitig aufgezeichnet werden können. Für das Videosignal wurden mobile digitale Videorekorder verwendet. Die aufgezeichneten Videos stehen in voller PAL Auflösung (720×576) bei einer Bildrate

[1] Institut Dalle Molle d'Intelligence Artificielle Perceptive, Rue Marconi 19, Case Postale 592, CH-1920 Martigny, Switzerland

Abbildung 2.1: *Der IDIAP Smart Meeting Room*

von 25 Hz zur Verfügung. Zur Videoaquirierung wurden identische CCTV-Kameras verwendet, die eine einstellbare Weitwinkellinse besitzen, womit das Gesichtsfeld in einem Bereich von $38° - 80°$ eingestellt werden kann. Zwei Kameras sind seitlich an den Wänden des Besprechungsraumes befestigt und haben jeweils die zwei auf der gegenüberliegenden Seite befindlichen Teilnehmer im Bild. Im Blickwinkel einer jeden Kamera befinden sich die Oberkörper der Personen sowie ein Teil des Konferenztisches. Die dritte Kamera erfasst alle Geschehnisse am Kopf des Konferenztisches, wie Aktionen am Whiteboard und Präsentationen auf der Projektionsleinwand. Bildschirmfotos aller drei Kameras sind in Abbildung 2.3 gezeigt.
Das Audiosignal wurde über Ansteckmikrophone erfasst, die jedem Teilnehmer zur Verfügung standen. Zusätzlich wurden alle Geräusche über ein achtfaches Mikrophon-Array, das auf dem Konferenztisch stand, aufgezeichnet.

2.1.2 Meeting Daten

Die M4-Datenbank besteht aus insgesamt 59 aufgezeichneten Besprechungen[2]. Jedes Meeting hat eine Dauer von etwa fünf Minuten. Es liegt also Videomaterial mit einer Gesamtdauer von circa fünf Stunden vor. An jeder Besprechung nahmen vier Personen teil. Jedes

[2]Ursprünglich wurden 60 Meetings aufgezeichnet. Aus inhaltlichen Gründen musste jedoch ein Meeting gestrichen werden, sodass der Corpus nun aus 59 Meetings besteht.

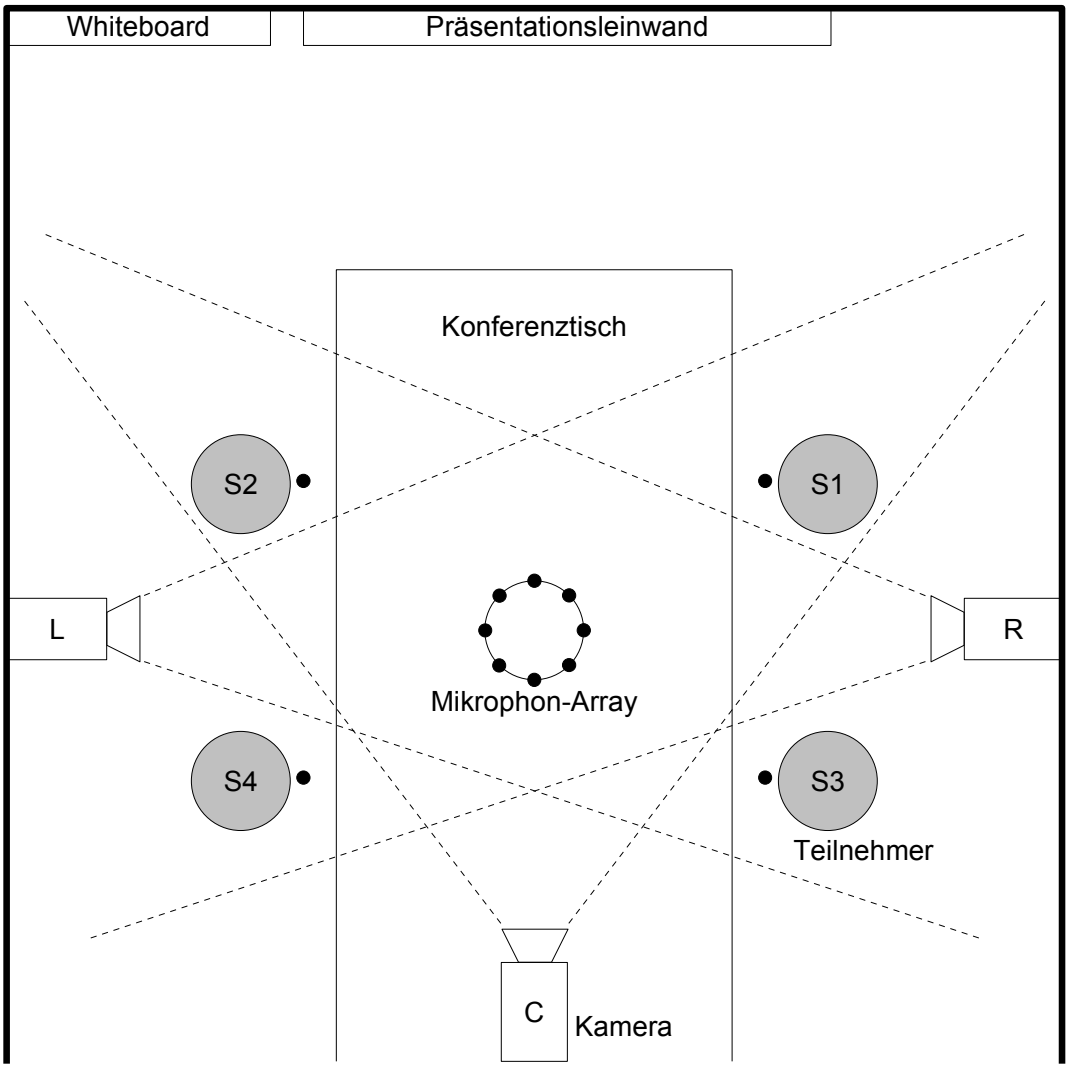


Abbildung 2.2: *Skizze des Besprechungsraumes im Forschungsinstitut IDIAP*

Meeting besitzt eine vorgegebene Agenda von Gruppenaktionen, an der sich die Teilnehmer orientieren. Die Abfolge der Gruppenaktionen wurde mit Hilfe eines ergodischen Hidden Markov Modells[3] vor der Aufzeichnung festgelegt [McC03]. Jede Gruppenaktion entspricht einem Zustand des Markov-Modells. Die Wahrscheinlichkeiten der Selbstübergänge steuern die Dauer jeder Gruppenaktion. Um sinnvolle Ergebnisse zu erhalten, wurden diese Wahrscheinlichkeiten manuell angepasst. Jedes Meeting besitzt durchschnittlich fünf Gruppenaktionen, wobei mit einem Monolog oder einer Diskussion begonnen und geschlossen wird. Zwei disjunkte Gruppen von jeweils acht Personen wurden aus der internationalen Belegschaft von IDIAP ausgewählt. Für jede Gruppe wurden dreißig Agenden erzeugt. Die vier Teilnehmer wurden nach dem Zufallsprinzip aus den acht Kandidaten ausgewählt. Jede Gruppenaktion wurde dann einem Teilnehmer zugeordnet. Gruppenaktionen, bei denen alle Teilnehmer betroffen sind, wurden von allen Teilneh-

[3]Zur Erklärung des Modells siehe Kapitel 5.1

Abbildung 2.3: *Bildschirmfotos der drei im IDIAP Meeting Raum verwendeten Kameraperspektiven (links: Kamera L, mitte: Kamera C, rechts: Kamera R*

mern ausgeführt. Jedem Meeting wurde zufällig ein Thema gegeben (z. B. „Mein letzter Urlaub“). Eine sich im Hintergrund aufhaltende Person war dafür zuständig, die Dauern der Aktionen zu überwachen und lautlose Gesten auszuführen, wenn ein Übergang von einer Aktion zu einer anderen erfolgen sollte.
Die Datenbank ist in ein festes Trainingsset und ein Testset mit jeweils vier Teilnehmern unterteilt. Die Trainingsdaten bestehen aus 30 Meetings, das Testmaterial setzt sich aus 29 Besprechungen zusammen. Die Teilnehmer aus dem Testset sind nicht im Trainigsmaterial enthalten.

2.1.3 Lexika

Aufbauend auf der M4-Datenbank werden zwei verschiedene Lexika definiert, die für die Unterteilung von Besprechungen von Relevanz sind. Diese Gruppenaktionen sind multimodal und werden häufig in einer Besprechung beobachtet. Im ersten Lexikon (Lexikon L8) werden acht Gruppenaktionen unterschieden, wie zum Beispiel Diskussion, Monolog und Präsentationen (siehe auch [McC05]). Die Monologe werden weiterhin nach der Person, die den jeweiligen Monolog hält, unterschieden. Dadurch ergeben sich die acht in Tabelle 2.1 definierten Klassen. Für die Modellierung wird angenommen, dass die definierten Aktionen sich nicht überlappen und ein vollständiges Set darstellen, das heißt,

Aktion	**Beschreibung**
Discussion	Zwei oder mehr Teilnehmer nehmen am Gespräch teil
Monologue1	Teilnehmer auf Position 1 spricht ohne Unterbrechung
Monologue2	Teilnehmer auf Position 2 spricht ohne Unterbrechung
Monologue3	Teilnehmer auf Position 3 spricht ohne Unterbrechung
Monologue4	Teilnehmer auf Position 4 spricht ohne Unterbrechung
Note-taking	Alle Teilnehmer schreiben sich Notizen auf
Presentation	Ein Teilnehmer hält eine Präsentation vor der Leinwand
Whiteboard	Ein Teilnehmer agiert vor dem Whiteboard

Tabelle 2.1: *Gruppenaktionen in Lexikon L8*

dass jeder Zeitpunkt einer Besprechung von einer Aktion beschrieben werden kann (es gibt keine Lücken). Für durchzuführende Experimente wurde die Datenbank in ein festes Trainigs- und ein festes Testset unterteilt (vgl. Abschnitt 2.1.2). Die Verteilung der einzelnen Klassen in der Trainings- bzw. Testmenge ist in Tabelle 2.2 gezeigt.

Aktion	**Trainingsset**	**Testset**
Discussion	48	49
Monologue1	14	12
Monologue2	10	13
Monologue3	10	14
Monologue4	9	10
Note-taking	6	3
Presentation	11	18
Whiteboard	16	20
Gesamt	124	139

Tabelle 2.2: *Anzahl der Klasseninstanzen des Lexikon L8 der Trainings- bzw. Testmenge*

Zusätzlich wurde ein erweitertes Set von Meeting Events definiert (vgl. [Zha04a]), das den realen Charakter eines Meetings besser beschreiben kann. Das zweite Lexikon (Lexikon L14) geht aus dem ersten hervor und wird durch Aktivitäten, die zusätzlich gleichzeitig stattfinden können, ergänzt, wie sie zum Beispiel bei Diktaten oder beim Protokollschreiben auftreten können. Insgesamt wurden 14 Klassen definiert, die in Tabelle 2.3 dargestellt sind. Wie beim ersten Set wird auch bei diesem zweiten Set davon ausgegangen, dass jeweils zwei Aktionen sich nicht überlappen. Außerdem wird angenommen, dass die Aktionen vollständig sind, das heißt, jedem Bereich eines Meetings kann eines der vorgestellten Label zugeordnet werden. Die Anzahl der Instanzen der Klassen des Lexikon L14 ist in Tabelle 2.4 gezeigt.
Generell besteht ein Meeting aus einer Abfolge der oben genannten Gruppenaktionen. Im Allgemeinen bewegt sich die Anzahl der Aktionen in einem Meeting im Bereich von drei bis sechs.

2.2 Grundlagen der Mustererkennung

Die Aufgabe der automatischen Mustererkennung besteht im Allgemeinen darin, einem gemessenen Signal eine bestimmte Bedeutung zuzuordnen [Rus02].
In Abbildung 2.4 ist der wesentliche Ablauf der Mustererkennung ausgehend von einem Signal bis zur Klassifikation gezeigt. Das gemessene Signal wird üblicherweise einer Vorverarbeitung unterzogen, da nahezu keine Aufgabe im Bereich der Mustererkennung existiert, bei der das ursprünglich beobachtete Signal direkt einem Klassifikator zugeführt werden kann [Rig05]. Ziel der Vorverarbeitung und Merkmalsextraktion ist oft die Verringerung

Aktion	Beschreibung
Discussion	Zwei oder mehr Teilnehmer nehmen am Gespräch teil
Monologue1	Teilnehmer auf Position 1 spricht ohne Unterbrechung
Monologue1 + Note-taking	Teilnehmer auf Position 1 spricht ohne Unterbrechung, während andere Teilnehmer Notizen machen
Monologue2	Teilnehmer auf Position 2 spricht ohne Unterbrechung
Monologue2 + Note-taking	Teilnehmer auf Position 2 spricht ohne Unterbrechung, während andere Teilnehmer Notizen machen
Monologue3	Teilnehmer auf Position 3 spricht ohne Unterbrechung
Monologue3 + Note-taking	Teilnehmer auf Position 3 spricht ohne Unterbrechung, während andere Teilnehmer Notizen machen
Monologue4	Teilnehmer auf Position 4 spricht ohne Unterbrechung
Monologue4 + Note-taking	Teilnehmer auf Position 4 spricht ohne Unterbrechung, während andere Teilnehmer Notizen machen
Note-taking	Alle Teilnehmer schreiben sich Notizen auf
Presentation	Ein Teilnehmer hält eine Präsentation vor der Leinwand
Presentation + Note-taking	Ein Teilnehmer hält eine Präsentation vor der Leinwand, während andere Teilnehmer Notizen machen
Whiteboard	Ein Teilnehmer agiert vor dem Whiteboard
Whiteboard + Note-taking	Ein Teilnehmer agiert vor dem Whiteboard, während andere Teilnehmer Notizen machen

Tabelle 2.3: *Gruppenaktionen des erweiterten Lexikon L14*

Aktion	Trainingsset	Testset
Discussion	48	49
Monologue1	10	6
Monologue1 + Note-taking	4	6
Monologue2	6	8
Monologue2 + Note-taking	4	5
Monologue3	5	7
Monologue3 + Note-taking	5	7
Monologue4	5	5
Monologue4 + Note-taking	4	5
Note-taking	6	3
Presentation	6	9
Presentation + Note-taking	5	9
Whiteboard	5	1
Whiteboard + Note-taking	11	19
Gesamt	124	139

Tabelle 2.4: *Anzahl der Klasseninstanzen des Lexikon L14 der Trainings- bzw. Testmenge*

von Störungen wie Rauschen, die Reduktion der Datenmenge oder die Transformation der Daten in einen anderen Merkmalsraum [Rig05].

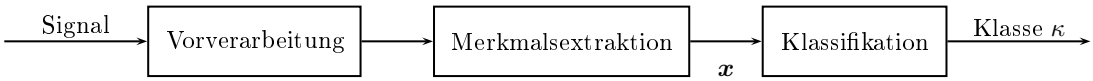


Abbildung 2.4: *Wesentlicher Ablauf bei der Klassifikation von Mustern*

Die Klassifikation hat die Aufgabe, die nach der Merkmalsextraktion vorliegenden Mustervektoren $\boldsymbol{x}$ einer jeweiligen Klasse κ von K Klassen zuzuordnen:

$$\boldsymbol{x} \mapsto \Omega_\kappa, \tag{2.1}$$

wobei Ω_κ aus der Menge $\Omega = \{\Omega_1, \Omega_2, \ldots, \Omega_K\}$ genommen wird. Dabei werden folgende Bedingungen eingehalten [Nie03]:

$$\Omega_\kappa \neq \emptyset \quad \text{für } \kappa = 1, \ldots, K \tag{2.2a}$$

$$\Omega_\kappa \cap \Omega_\lambda = \emptyset \quad \text{für } \kappa \neq \lambda \tag{2.2b}$$

$$\bigcup_{\kappa=1}^{K} \Omega_\kappa = \Omega \tag{2.2c}$$

Die Klassen Ω_κ sind hier als disjunkt vorausgesetzt. Für Muster, die nicht eindeutig einer Klasse zugeordnet werden können, kann eine zusätzliche Rückweisungsklasse Ω_0 eingeführt werden.

Aus der Menge der verfügbaren Klassen wählt der Klassifikator die in einem gewissen Sinne am besten passende Klasse aus. Die so erkannte Klasse wird im Weiteren mit κ_e bezeichnet.
Welche Art von Klassifikator zum Einsatz kommt, hängt von verschiedenen Faktoren ab, die sowohl die Güte der Erkennungsleistung, als auch die Effizienz oder ökonomische Faktoren betreffen[4]. Daraus begründet sich auch der in dieser Arbeit angestrebte Vergleich unterschiedlicher Verfahren zur Untergliederung von Besprechungen.

2.3 Multimodale Merkmalsextraktion

Dieses Kapitel beschreibt die Erstellung geeigneter Merkmale zur Erkennung und Segmentierung von Gruppenaktionen, wie sie in Abschnitt 2.1.3 beschrieben wurden. In Abschnitt 2.3.1 werden zunächst Merkmale erläutert, die direkt aus den Signalströmen induziert werden können. In Abschnitt 2.3.2 werden dann daraus abgeleitete, für den Menschen leicht interpretierbare Merkmale vorgestellt.

2.3.1 Low-Level-Merkmale

In diesem Abschnitt werden die Merkmale beschrieben, die direkt aus dem Audiosignal oder aus dem Videosignal abgeleitet werden können. Dazu gehören unter anderem Sprecher-Wechsel (engl. Speaker-Turn), Energie, relative Tonhöhe, Sprechrate, MFCC, Hautfarben-Bereiche (engl. Skin-color Blobs) und globale Bewegungsmerkmale (engl. Global Motion Features).

Audiobasierte Merkmale

Speaker-Turn Detection:[5] Von den Signalen des Mikrophon-Arrays wird die „Sprach-Aktivität“ an sechs verschiedenen Orten bestimmt: an jedem der vier Sitzplätze, sowie an den Orten, die mit den Aktionen *Whiteboard* und *Presentation* korrespondieren. Diese Orte werden als dreidimensionale Vektoren festgelegt. Die „Sprach-Aktivität“ wird als Steered Response Power (SRP) unter Zuhilfenahme des SRP-PHAT[6] Maßes [DiB00, DiB01] berechnet. Das ist ein kontinuierlicher Wert, der die Aktivität einer speziellen Position angibt. Dadurch ist es möglich zu entscheiden, wann welche Position aktiv ist. Anschließend wird mit Hilfe des Verhältnisses der Wahrscheinlichkeiten von Sprache und Stille bzw. mit Hilfe eines Steered Response Power Ansatzes [Lat03] eine Segmentierung in Sprache bzw. Ruhe vorgenommen.

Aus den Sprachsegmenten, die von der Speaker-Turn Detection gewonnen wurden, werden die folgenden drei Merkmale berechnet:

[4] Für eine ausführlichere Behandlung diese Themas sei auf die einschlägige Literatur verwiesen (siehe z.B. [Sch06])

[5] Die folgenden vier Merkmale (Speaker-Turn Detection, Energie, Relative Tonhöhe und Sprechrate) wurden uns freundlicherweise vom IDIAP zur Verfügung gestellt (vgl. [McC05])

[6] Steered Response Power with Phase Transform

Energie: Die Signalenergie wird in quasi-stationären Kurzzeitfenstern berechnet. Zur Fensterung wird eine Fensterfunktion $w(n)$, hier ein Hamming-Fenster $w_{\text{Ham}}(n)$, verwendet:

$$w_{\text{Ham}}(n) = 0.54 - 0.46 \cdot \cos\left(\frac{2\pi n}{N}\right), \quad n = -\frac{N}{2}, \ldots, \frac{N}{2} \tag{2.3}$$

N entspricht hierbei der Fensterbreite, n ist die aktuelle Position innerhalb des Fensters. Die Fensterbreite beträgt in diesem Fall 32ms. Die Berechnung der Energie E erfolgt dann in jedem Fenster mit Gleichung (2.4).

$$E = \sum_{n=1}^{N} |s(n)|^2 \tag{2.4}$$

Relative Tonhöhe: Die Tonhöhe wird mit Hilfe des SIFT-Algorithmus[7] [Mar72] berechnet. Dieser Algorithmus kombiniert gewünschte Eigenschaften der Autokorrelation und der Cepstralanalyse. Gleichzeitig führt dieser Algorithmus eine Unterscheidung in stimmhafte und stimmlose Abschnitte durch. Der erste Schritt besteht in einer inversen Filteranalyse. Die Ausgangswerte des inversen Filters werden danach einer Autokorrelation unterzogen. Der Reziprokwert der größten Erhebung der so erhaltenen Sequenz bestimmt die Grundfrequenz F_0, sofern dieser Abschnitt als stimmhaft eingestuft wird.

Sprechrate: Die Sprechrate wird durch eine Kombination von Schätzern [Lat03] gewonnen. Für jeden Frame werden die Wahrscheinlichkeiten für Sprache bzw. Stille bestimmt, die aus Zeitverzögerungsschätungen [Bra97] mit verschiedenen Wahrscheinlichkeitsdichtefunktionen gewonnen werden. Daraus kann dann das Verhältnis von Sprache zu Stille als Quotient der Wahrscheinlichkeiten berechnet werden.

Mel-Frequenz-Cepstral-Koeffizienten: Unter anderem in der automatischen Spracherkennung eingesetzte Merkmale sind Mel-Frequenz-Cepstral-Koeffizienten (MFCC). Sie sollen eine Trennung des Anregungssignals von der Impulsantwort des Vokaltrakts ermöglichen. Damit ist es möglich, die informationstragenden Anteile zu extrahieren. MFCC werden im Wesentlichen durch folgende Schritte berechnet:

1. Unterteilung des Eingangssignals in quasi-stationäre Segmente durch Fensterung (hier mit einem Hamming-Fenster, vgl. Glg. (2.3)).
2. Berechnung des Kurzzeitleistungsspektrums (Diskrete Fourier-Transformation) eines Fensters
$$X_{\text{DFT}}[n] = \sum_{k=0}^{N-1} x[k]\, e^{-j\frac{2\pi n}{N}k} \tag{2.5}$$
3. Filterung des Spektrums mit einer Mel-Filterbank (ähnlich dem menschlichen Gehör)
$$S_{\text{Mel}}(f) = 2595 \log_{10}\left(1 + \frac{f}{700}\right) \tag{2.6}$$

[7] SIFT: Simple Inverse Filter Tracking

4. Logarithmieren des Mel-skalierten Leistungsspektrums
5. Dekorrelation durch abschließende Diskrete Cosinus Transformation (DCT)

$$c_i = \sqrt{\frac{2}{N}} \sum_{j=1}^{N} m_j \cos\left(\frac{\pi i}{N}(j - 0.5)\right) \tag{2.7}$$

Hierbei ist m_j die Amplitude in der j-ten Filterbank, N ist die Anzahl der Filterbänke. In dieser Arbeit wurde mit $N = 12$ Koeffizienten gearbeitet.

Zusätzlich zu den Koeffizienten wird auch die erste und zweite zeitliche (diskrete) Ableitung berechnet: $\Delta c_i = c_i - c_{i-1}$, bzw. $\Delta\Delta c_i = \Delta c_i - \Delta c_{i-1}$.

Videobasierte Merkmale

Merkmale aus dem Videostream werden mit Standardmethoden extrahiert. Dabei werden die Bereiche der Personen (Kopf, Schulter, Hände), sowie die Umgebung des Whiteboards und der Präsentationsleinwand berücksichtigt.

Skin-color Blob Merkmale:[8] Auf den Bildern der beiden Kameras, die auf jeweils zwei Teilnehmer zeigen, werden Gaussian Mixture Models (GMM) (vgl. Kapitel 3.1.2) der Hautfarbe im RGB-Farbraum verwendet, um Kopf- und Hand-/Unterarm-Blobs[9] zu extrahieren [Jon02]. Ein GMM mit zwanzig Mixturen wird aus den Trainingsdaten trainiert, die Bilder von Köpfen und Händen von kaukasischen, indischen und lateinamerikanischen Personen enthalten. Hautfarbene Pixel werden klassifiziert, wenn die Wahrscheinlichkeit für Hautfarbe über einer Schwelle liegt. Eine morphologische Nachverarbeitung der typischen Regionen für den Kopf hilft, Skin-Blobs zu extrahieren.

Für jede Person werden die gefundenen Kopf-Blobs durch die vertikale Position des Schwerpunkts (normiert auf den gemittelten Schwerpunkt des gesamten Meetings) repräsentiert. Die Hand-Blobs werden zusätzlich durch drei Merkmale charakterisiert: die normierte horizontale Position des Schwerpunkts, die Exzentrizität und den Winkel zur Horizontalen [Sta95]. Aufgrund der Schwierigkeit, die Hände immer richtig zuordnen zu können, beschränkt man sich hier auf die rechte Hand, indem angenommen wird, dass diese immer weiter links auf dem Videobild zu finden ist, als die linke Hand.

Zuletzt wird noch ein grobes Maß für die Bewegung der Person berechnet. Man betrachtet dazu den Durchschnitt der Bewegungen des Kopfes und der Hand. Diese Bewegungen berechnen sich als die Differenz der Schwerpunkte von Kopf bzw. Hand in aufeinanderfolgenden Frames.

[8] Die Merkmale, die aus den Skin-Blobs berechnet wurden, wurden uns freundlicherweise vom IDIAP zur Verfügung gestellt (vgl. [McC05])

[9] Ein Blob beschreibt ein zusammenhängendes Gebiet in einem Bild

Aus den Bildern der dritten Kamera werden sich bewegende Blobs extrahiert, indem der Hintergrund subtrahiert wird. Repräsentiert werden die Blobs durch ihre horizontale Position. Da ein festes Hintergrundbild zur Subtraktion verwendet wird, treten in seltenen Fällen Fehler in der Merkmalsextraktion auf. McCowan et al. [McC05] schlagen ein adaptives Verfahren zur Hintergrund-Subtraktion vor, mit dem sich solche Artefakte vermeiden lassen sollen.

Global Motion Features: Global Motion Features (GMF) werden bereits erfolgreich bei der Erkennung von Gesten in Meetings eingesetzt [Eic98], [Wal04b], [Zob03]. Sie werden aus Differenzbildern berechnet. Ein Differenzbild $I_d^{'}$ ist die Differenz aus einem Vorgängerbild I_b und dem aktuellen Frame I_a.

$$I_d^{'}(x,y,t) = I_b(x,y,t) - I_a(x,y,t) \tag{2.8}$$

Hierbei bezeichnen x und y Koordinaten im Bild in x- bzw. y-Richtung. Die Zeit wird mit t bezeichnet.

Um Rauschen und sonstige Artefakte zu eliminieren, wird ein Schwellwertoperator nachgeschaltet. Alle Pixelwerte, die unter einer Schwelle T_I liegen, werden zu Null gesetzt:

$$I_d(x,y,t) = \begin{cases} 0 & |I_{db}^{'}(x,y,t)| < T_I \\ I_d^{'}(x,y,t) & |I_d^{'}(x,y,t)| \geq T_I \end{cases} \tag{2.9}$$

Weitere Interferenzen werden durch morphologische Operationen entfernt. Wenn an einer der sechs interessierenden Positionen (Sitzplätze, Whiteboard, Präsentationsleinwand) die Summe der verbleibenden Pixel

$$\sum_{(x,y)\in R_i} |I_d(x,y,t)| > T_R \tag{2.10}$$

größer ist als ein Schwellwert T_R, wird um den Massenschwerpunkt ($[p_x(t), p_y(t)]$) eine Region R_i festgelegt, in der GMF wie folgt berechnet werden [Rig98a]: Ausgehend vom Differenzbild $I_d(x,y) = I_b(x,y,t-2) - I_a(x,y,t)$, das aus jedem zweiten Frame und dem Vorgänger $I_b(x,y,t-2)$ berechnet wird, können Merkmale bestimmt werden, die die Bewegung in der interessierenden Region beschreiben. Das *Zentrum der Bewegung* wird durch die Berechnung des Massenschwerpunkts $\boldsymbol{m'}(t) = [m'_x(t), m'_y(t)]^T$ gewonnen.

$$m'_x(t) = \frac{\sum\limits_{(x,y)\in R_i} x \cdot |I_d(x,y,t)|}{\sum\limits_{(x,y)\in R_i} |I_d(x,y,t)|}, \quad m'_y(t) = \frac{\sum\limits_{(x,y)\in R_i} y \cdot |I_d(x,y,t)|}{\sum\limits_{(x,y)\in R_i} |I_d(x,y,t)|} \tag{2.11}$$

Damit die Merkmale unabhängig von der absoluten Position der Person sind, werden sie mit dem Massenschwerpunkt normiert:

$$\boldsymbol{m}(t) = [m_x(t), m_y(t)] = \left[m'_x(t) - p_x(t), m'_y(t) - p_y(t)\right]^T \tag{2.12}$$

Um die *Richtungsänderungen der Bewegung* zu berücksichtigen, werden zusätzlich die zeitliche Änderung der Schwerpunkte $\Delta m_x(t)$ in x– und $\Delta m_y(t)$ in y–Richtung bestimmt:

$$\Delta m_x(t) = m_x(t) - m_x(t-1), \quad \Delta m_y(t) = m_y(t) - m_y(t-1) \tag{2.13}$$

Zusätzlich wird noch die *Standardabweichung eines Pixels* (x, y) relativ zum Bewegungszentrum benutzt:

$$\boldsymbol{\sigma}(t) = [\sigma_x(t), \sigma_y(t)]^T \tag{2.14a}$$

$$\sigma_x^2(t) = \frac{\sum\limits_{(x,y)\in R_i} |I_d(x,y,t)| \cdot (x - m_x(t))^2}{\sum\limits_{(x,y)\in R_i} |I_d(x,y,t)|} \tag{2.14b}$$

$$\sigma_y^2(t) = \frac{\sum\limits_{(x,y)\in R_i} |I_d(x,y,t)| \cdot (y - m_y(t))^2}{\sum\limits_{(x,y)\in R_i} |I_d(x,y,t)|} \tag{2.14c}$$

Ein weiteres wichtiges Merkmal zur Beschreibung der Bewegung ist die *Intensität der Bewegung* $i(t)$. Sie ist definiert als der Durchschnitt der absoluten Werte der Bewegungsverteilung:

$$i(t) = \frac{\sum\limits_{(x,y)\in R_i} |I_d(x,y,t)|}{\sum\limits_{(x,y)\in R_i} 1} \tag{2.15}$$

Große Werte von $i(t)$ repräsentieren intensive Bewegungen der Teilnehmer, während kleine Werte ein fast stehendes Bild charakterisieren.

Alle sieben Global Motion Merkmale können dann zu einem sieben-dimensionalen Merkmalsvektor zusammengefasst werden:

$$\boldsymbol{x}_{\mathrm{GMF}}(t) = [m_x(t), m_y(t), \Delta m_x(t), \Delta m_y(t), \sigma_x(t), \sigma_y(t), i(t)]^T \tag{2.16}$$

2.3.2 Semantische Merkmale

Unter semantischen Merkmalen werden hier Merkmale verstanden, die im Unterschied zu den Low-Level Merkmalen die Bedeutung des Inhalts beschreiben. Um diese semantischen Merkmale zu erhalten, werden die Merkmalsströme in zeitliche Abschnitte unterteilt und mit einem Label versehen. Semantische Merkmale können aus den Beschreibungen der Abschnitte abgeleitet werden, wie zum Beispiel die Dauer, oder der Beginn eines Abschnitts. Da die Merkmale bzw. Abschnitte nicht von vornherein zur Verfügung stehen und keine zuverlässigen automatischen Methoden zur Generierung derselben vorhanden sind, müssen die Daten zuerst manuell annotiert werden. Anschließend können auf dieser Basis Algorithmen angewendet werden, die zur Erkennung und Segmentierung von Gruppenaktionen dienen.

Annotierung

Die Annotierung des Ausgangsmaterials ist grundlegend für die Verwendung von Algorithmen und Tools, die auf den Ergebnissen von vorgeschalteten Erkennern oder Algorithmen aufbauen sollen. So werden die Annotierungen zum einen zum Training für spezielle Klassifikatoren benötigt, wie zum Beispiel Gestenerkenner oder ähnliche Verfahren, zum anderen sind sie als Testumgebung unentbehrlich, um einen Eindruck zu bekommen, wie gut darauf aufbauende Verfahren für die gestellte Aufgabe geeignet sind. Dies stellt quasi ein „upper benchmarking“ dar, ein bestes theoretisch zu erreichendes Ergebnis, wenn die zugrunde liegenden Klassifikatoren keine Klassifizierungsfehler machen würden. Um nur die darauf aufbauenden Verfahren zu testen, werden manuell erzeugte Annotationen verwendet; dadurch wird eine gewisse Unabhängigkeit von Klassifikationsfehlern in einem vorhergenden Schritt erreicht.
Die Annotierung wurde mit dem graphischen Annotierungsprogramm Anvil[10] [Kip01] vorgenommen. Dazu wurde ein Annotationsschema entwickelt, das sowohl aktuelle, als auch zukünftige Anforderungen abdeckt. Das Schema orientiert sich dabei an einer Studie von Marchand-Maillet [MM03], in der „Meeting-activities“ (Meeting Actions, Meeting Events, Gruppenaktionen) definiert werden und ein Benutzermodell vorgestellt wird. Die Meeting-Aktivitäten gliedern sich wie folgt:

`Opening, Presentation, Vote, Break, Global Discussion, Multidiscussion, Silence, Closing.`

Um Konformität mit anderen Forschungsgruppen zu erreichen, wurden letztendlich folgende Gruppenaktionen annotiert (vgl 2.1.3):

- Discussion
- Monologue
- Note-taking
- Whiteboard
- Presentation
- Consensus
- Disagreement

Das Benutzermodell teilt jeden Teilnehmer einer Besprechung in vier Aktivitätsbereiche ein:

`Ort, Interaktivität, Aktivität und Physical-State.`

[10] Das Programm Anvil kann kostenlos unter `http://www.dfki.de/~kipp/anvil/` heruntergeladen werden.

Ort kann einer der Sitzplätze sein, der Platz vor dem Whiteboard oder der Präsentationsleinwand. Andernfalls ist der Ort undefiniert. Als *Interaktivität* definieren sich alle Aktionen, die in einer erkennbaren Weise einen anderen Teilnehmer involvieren. Darunter fallen Aktionen wie „nodding“, „talking“, „silent“, „chatting“, „laughing“ und „shake-head“. *Aktivität* beschreibt Aktionen, bei denen keine direkte Interaktion mit einem anderen Teilnehmer stattfindet. Als Beispiele sind hier „writing“, „voting“ und „pointing“ zu nennen. Der Ruhezustand wird als „idle“ beschrieben. Der *Physical-State* beschreibt, in welcher körperlichen Position ein Teilnehmer eines Meetings sich befindet. Darunter fallen die Zustände „sitting“, „standing“, „walking“, „absent“ und „still“.
Dieses Annotationsschema wurde auf alle verfügbaren Videoaufzeichnungen der M4 Datenbank (siehe 2.1.2) angewendet. Es liegen damit die Annotierungen von 59 Meetings vor.

Merkmalsextraktion Aus der oben beschriebenen Annotierung können Merkmale mit semantischem Bedeutungsinhalt gewonnen werden. Dazu werden alle in einem bestimmten vorgegebenen Zeitintervall auftretenden Aktionen getrennt voneinander ausgewertet. Die zeitlichen Dauern der einzelnen Aktionen werden für jede Person separat aufsummiert. Zusätzlich werden die so entstandenen Größen auf die Länge des vorgegebenen Zeitfensters normiert, so dass eine Unabhängigkeit von der Wahl der Länge des Zeitfensters gegeben ist. Es ergeben sich also relative Dauern einer Aktion innerhalb eines vorgebeben Zeitfensters (zum Beispiel eine Gruppenaktion). Während eines Monologs ist zum Beispiel die relative Dauer von „talking“ des Redners nahe bei 1, während sie für die restlichen Teilnehmer nahezu 0 sein sollte. Äquivalent dazu ist der Ort „Whiteboard“ für den Präsentierenden nahezu 1 und für die übrigen Teilnehmer nahe bei 0. Auf diese Weise werden alle Aktionen, die annotiert werden, ausgewertet. Dadurch wird auch jede Aktion gleich stark gewichtet. Aus folgenden Aktionen werden Größen abgeleitet, aus denen der Merkmalsvektor zusammengestellt werden kann:

- **Interaktivität:** nodding, talking, shakehead
- **Aktivität:** idle, writing, voting, pointing
- **Position:** seat, whiteboard, presentation
- **Physical-State:** sitting, standing, walking, absent, still

Insgesamt ergeben sich dadurch $15 \cdot 4 = 60$ verschiedene Größen, da immer vier Teilnehmer vorhanden sind.
Zusätzlich wird noch der „Hauptredner“ n_{HR} extrahiert, indem über die Größe „talking“ der teilnehmenden Personen das Maximum gesucht wird:

$$n_{\mathrm{HR}} = \arg\max_{n} \quad \mathrm{talking}(n), \quad n = 1, \ldots, 4 \tag{2.17}$$

Aus den direkt verfügbaren Größen werden weitere Größen mit Hilfe von Funktionalen abgeleitet, die für die Aufgabe der Erkennung von Gruppenaktionen als diskriminativ eingeschätzt werden:

- der Abstand des „talking“ des Hauptredners, zu der Summe der anderen Teilnehmer:

$$d^{(1)}_{t_{\mathrm{HR}}} = \mathrm{talking}(n_{\mathrm{HR}}) - \left[\sum_{\substack{n=1 \\ n \neq n_{\mathrm{HR}}}}^{4} \mathrm{talking}(n)\right] \tag{2.18}$$

- der Abstand des „talking“ des Hauptredners zum zweitgrößten „talking“:

$$d^{(2)}_{t_{\mathrm{HR}}} = \mathrm{talking}(n_{\mathrm{HR}}) - \mathrm{talking}(\text{zweithäufigster Redner}) \tag{2.19}$$

- der Abstand dessen, der am meisten geschrieben hat n_{HW}, zu der Summe der anderen:

$$d_{w_{\mathrm{HW}}} = \mathrm{writing}(n_{\mathrm{HW}}) - \left[\sum_{\substack{n=1 \\ n \neq n_{\mathrm{HW}}}}^{4} \mathrm{writing}(n)\right] \tag{2.20}$$

$$\text{mit} \quad n_{\mathrm{HW}} = \arg\max_{n} \quad \mathrm{writing}(n)$$

- die maximale relative Länge von „standing“, d.h. die längste relative Dauer innerhalb eines Zeitfensters, die eine Person steht.

- die Anzahl der Sprecherwechsel

Für die Klassifikation können verschiedene Merkmalsvektoren verwendet werden. Beispielhaft ist der folgende Merkmalsvektor $\boldsymbol{f}$ angeführt:

$$\boldsymbol{f} = \Bigg[\mathrm{talking}(1),\ \mathrm{talking}(2),\ \mathrm{talking}(3),\ \mathrm{talking}(4),\ \sum_n \mathrm{whiteboard}(n), \\ \sum_n \mathrm{presentation}(n),\ \sum_n \mathrm{writing}(n),\ \sum_n \mathrm{standing}(n),\ d^{(2)}_{t_{\mathrm{HR}}}\Bigg] \tag{2.21}$$

Automatisch generierte Daten

Selbstverständlich müssen für eine reale Anwendung automatisierte Verfahren ohne manuellen Eingriff die entsprechenden notwendigen Merkmale zur Verfügung stellen. Dies wird in dieser Arbeit durch die Ergebnisse einer automatischen Sprecheraktivitätserkennung [Lat04] und einer Gestenerkennung [Wal04b] erreicht. Das Vokabular der Gestenerkennung beinhaltet sechs verschiedene Gesten (*sit down*, *stand up*, *nodding*, *shaking*, *writing*, *pointing*). Von der Güte der Gesten- bzw. Sprecheraktivitätserkennung hängt auch die weitere Leistung der Segmentierung ab. Bei der Gestenerkennung konnte eine Erkennungsrate von 86 % erreicht werden [Wal04b]. Für die Sprecheraktivitätserkennung erfolgte eine bis zu 94,7 % richtige Zuordnung [Lat04].

Merkmalsvektor Aus den Resultaten der Sprecheraktivitätserkennung und der Gestenerkennung wird in ähnlicher Weise wie bei den manuell erstellten Annotationen ein Merkmalsvektor erstellt. Dieser enthält nun statistische Werte der Einzelaktionen der Meeting-Teilnehmer. Ebenso wie im Abschnitt zuvor werden die Summen der Aktionen auf die Dauer des betrachteten Zeitfensters normiert. Auf diese Weise werden wieder Werte zwischen 0 und 1 gewonnen. Dadurch können nun auch die gleichen Segmentierungsalgorithmen angewendet werden wie bei den manuell erzeugten Annotationen. Ein Merkmalsvektor $\boldsymbol{f}$ besteht hier aus den sprachlichen Aktivitäten sowie einer Auswahl der erkannten Gesten (in diesem Fall nur *writing*):

$$\boldsymbol{f} = \Big[\text{talking}(1),\ \text{talking}(2),\ \text{talking}(3),\ \text{talking}(4),\ \text{talking}(5),\ \text{talking}(6), \sum_n \text{writing}(n)\Big] \quad (2.22)$$

2.3.3 Gruppen von Merkmalen

		Beschreibung	IDIAP	TUM	UEDIN
Personen-spezifische Merkmale	Visuell	Kopf vertikaler Schwerpunkt	X		
		Kopf Exzentrizität	X		
		rechte Hand horizontaler Schwerpunkt	X		
		rechte Hand Winkel	X		
		rechte Hand Exzentrizität	X		
		Kopf- und Hand-Bewegung	X		
		Global Motion Features von jedem Platz		X	
	Auditiv	SRP-PHAT von jedem Platz	X		
		Sprache relative Tonhöhe	X		X
		Sprache Energie	X	X	X
		Sprechrate	X		X
		MFCC Koeffizienten		X	
		Binäre Sprache und Pause Segmentierung		X	
	Semantisch	Individuelle Gesten		X	
		Sprech-Aktivität		X	
Guppen-Merkmale	Visuell	Mittlere Abweichung im Bereich Whiteboard	X		
		Mittlere Abweichung im Bereich Leinwand	X		
		Global Motion Features vom Whiteboard		X	
		Global Motion Features von der Leinwand		X	
	Auditiv	SRP-PHAT vom Whiteboard	X		
		SRP-PHAT von der Leinwand	X		
		Sprecher Aktivität Merkmale			X
		Binäre Sprache im Bereich Whiteboard		X	
		Binäre Sprache im Bereich Leinwand		X	

Tabelle 2.5: *Übersicht über auditive, visuelle and semantische Merkmale, und die resultierenden Merkmalssätze*

Die vorgestellten Merkmale werden zur leichteren Referenzierung zu Gruppen zusammengefasst. Die Merkmale und die Gruppen sind in Tabelle 2.5 aufgelistet. Die drei rechten Spalten geben jeweils an, von welcher Forschungsgruppe welche Merkmale erstellt worden sind. Zhang [Zha04b] verwendet ausschließlich Low-Level Merkmale (aus der Spalte IDIAP), Dielmann [Die04] von der Universität von Edinburgh (Spalte UEDIN) verwendet

nur ein Subset davon und ein zusätzliches Gruppenmerkmal. Semantische Merkmale werden erstmals in dieser Arbeit eingesetzt. Zur leichteren Referenzierung werden Gruppen von Merkmalen mit einer Abkürzung bezeichnet. Diese sind in Tabelle 2.6 gezeigt.

Abkürzung	Beschreibung
SEM-M	manuell erzeugte semantische Merkmale (Annotationen)
SEM-M+S	manuell erzeugte semantische Merkmale mit automatischer Sprechersegmentierung
SEM-A	automatisch erzeugte semantische Merkmale
LOW	automatisch erzeugte Low-Level Merkmale

Tabelle 2.6: *Gruppen von Merkmalen*

2.4 Bewertungsmaße

Die Evaluierung der Ergebnisse soll objektiv und reproduzierbar sein. Aus diesem Grund werden standardisierte Verfahren verwendet, die teilweise aus der Spracherkennung bekannt sind, um die Resultate der verwendeten Algorithmen vergleichen zu können. Im Folgenden werden die in dieser Arbeit verwendeten Bewertungsmaße vorgestellt.

2.4.1 Frame Error Rate

Die Frame Error Rate (Frame-Fehlerrate, FER) ist ein Maß, wie gut zwei Sequenzen elementweise übereinstimmen. Dabei werden keine Segmentgrenzen berücksichtigt. Die FER gibt den Prozentsatz von falsch klassifizierten Frames in Bezug auf alle Frames an und ist wie folgt definiert:

$$\mathrm{FER} = \frac{\text{Anzahl der falsch erkannten Frames}}{\text{Gesamtanzahl der Frames}} \cdot 100\,\% \tag{2.23}$$

Mit dieser Art der Fehlermessung kann ermittelt werden, ob der zeitliche Abgleich der erkannten Segmente innerhalb sinnvoller Grenzen liegt.
Diese Art der Fehlermessung wird in leicht abgewandelter Form auch bei Aufgaben in der Video-Indexierung verwendet (vgl. [Eic00]) und im Folgenden auch mit *M1* bezeichnet.

2.4.2 Action Error Rate

Die Action Error Rate (Event-Fehlerrate, AER) ist analog zur Wortfehlerrate in der automatischen Spracherkennung definiert. Dieses Maß legt einen größeren Schwerpunkt auf korrekt erkannte Sequenzen von Gruppenaktionen und vernachlässigt präzise Segmentgrenzen [AH06]. Die AER ist definiert aus der Summe der Zahl der Einfügungen (*insertions, Ins*, zusätzlich erkannte Events ohne Entsprechung in der Ground Truth) N_{Ins}, der Zahl der Summe der Löschungen (*deletions, Del*, ausgelassene Events) N_{Del} und der Zahl

der Ersetzungen (*substitutions, Sub*, falsch benannte Segmente, bei richtigen Segmentgrenzen) N_{Sub}, dividiert durch die Gesamtzahl der Events in der Ground Truth N_{Gesamt} multipliziert mit 100 [McC05]:

$$\text{AER} = \frac{N_{\text{Sub}} + N_{\text{Ins}} + N_{\text{Del}}}{N_{\text{Gesamt}}} \cdot 100\,\% \tag{2.24}$$

2.4.3 Akkuratheit

Alternativ zur Action Error Rate kann auch das komplementäre Maß der Akkuratheit verwendet werden. Es wird bestimmt indem die AER von 100 % subtrahiert wird:

$$\text{Akkuratheit} = 100\,\% - \text{AER} = \left(1 - \frac{N_{\text{Sub}} + N_{\text{Ins}} + N_{\text{Del}}}{N_{\text{Gesamt}}}\right) \cdot 100\,\% \tag{2.25}$$

2.4.4 Fehlermaß auf Basis der Segment-Übereinstimmung

Ein speziell auf die Bedürfnisse des Video-Indexing zugeschnittenes Fehlermaß wird von Hampapur vorgeschlagen [Ham95]: Dieses Fehlermaß basiert auf der maximalen globalen Intervallüberlappung zweier Segmente im Referenz- und Testset. Dieses Fehlermaß kann wie folgt berechnet werden:
Zuerst wird die zeitliche Überlappung $O(T_i, T_j)$ aller Intervalle $T_i = [t_{ib}; t_{ie}]$, $T_j = [t_{jb}; t_{je}]$ der Referenz- und des Testsegmentierung berechnet:

$$O(T_i, T_j) = |t_{ib} - t_{je}| - (|t_{ib} - t_{jb}| + |t_{ie} - t_{je}|) \tag{2.26}$$

Dabei ist T_i ein Intervall, das zum Zeitpunkt t_{ib} beginnt und zum Zeitpunkt t_{ie} endet. Anders als Hampapur schlägt Eickeler [Eic00] folgende Gleichung vor:

$$O(T_i, T_j) = \frac{1}{2}\left(|t_{ib} - t_{je}| + |t_{jb} - t_{ie}| - |t_{ib} - t_{jb}| - |t_{ie} - t_{je}|\right) \tag{2.27}$$

Diese Gleichung liefert für alle $t_{ib} \leq t_{ie}$ und $t_{jb} \leq t_{je}$ ein korrektes Ergebnis.
Die Intervalle der Testsegmentierung werden dann so mit der Referenz in Übereinstimmung gebracht, dass die Summe der zeitlichen Überlappung ein Maximum erreicht. Das kann zum Beispiel mit Hilfe der Dynamischen Programmierung [Bel59] erreicht werden.
Im nächsten Schritt werden die Intervalle des Test- bzw. Referenzsets, die keine Übereinstimmung haben, gezählt. Dadurch erhält man die Größen n' bzw. k'.
Das Fehlermaß E_{sb} für den Segmentierungsschritt wird dann wie folgt definiert:

$$E_{sb} = \frac{\sum\limits_{k} e_{sb}(i,j)}{f} + \frac{n' + k'}{\lambda} \tag{2.28}$$

Dabei ist

$$e_{sb}(i,j) = |t_{ib} - t_{jb}| + |t_{ie} - t_{je}| \tag{2.29}$$

und

$$\lambda = \begin{cases} n & \text{falls } k' \leq n' \text{ (Unter- oder Gleichsegmentierung)} \\ f & \text{falls } k' > n' \text{ (Übersegmentierung)} \end{cases} \tag{2.30}$$

f ist die Anzahl der Frames der Videosequenz, n die Anzahl der Segmente der Referenz und k die Anzahl der automatisch bestimmten Segmente. $e_{sb}(i,j)$ ist der Fehler zwischen zwei korrespondierenden Segmenten des Referenz- und Testsets. Anstelle des zweiten Summanden in Gleichung (2.28) wird

$$\frac{n'}{n} + \frac{k'}{k} \tag{2.31}$$

vorgeschlagen, um eine zu große Variation von λ für $f \gg n$ zu vermeiden.
Das Fehlermaß E_{sc} für den Klassifikationsschritt ist wie folgt definiert:

$$E_{sc} = \frac{\sum\limits_{k} e_{sc}(i,j)}{f} + \frac{n' + k'}{\lambda} \tag{2.32}$$

Dabei gilt, dass $e_{sc}(i,j) = 1$ ist, falls die erkannte Klasse mit der Referenzklasse übereinstimmt, und $e_{sc}(i,j) = 0$ ist, falls die Klassen differieren.
Die gesamte Fehlerrate E ist dann wie folgt definiert:

$$E = w_{sb} \cdot E_{sb} + w_{sc} \cdot E_{sc} \tag{2.33}$$

Die Gewichtsfaktoren w_{sb} und w_{sc} müssen vom Anwender bestimmt werden. Aus diesem Grunde werden meist die Werte E_{sb} und E_{sc} separat angegeben. Auf dieses Fehlermaß wird im Folgenden durch die Bezeichnung *M2* referenziert.

2.4.5 Fehlermaß auf Basis der Segmentgrenzen-Übereinstimmung

Von Eickeler [Eic00] wird ein Fehlermaß vorgeschlagen, das auf der Übereinstimmung der Segmentgrenzen beruht. Dieses Fehlermaß ist eine Erweiterung eines von Gargi [Gar96] vorgeschlagenen Maßes. Für jede eingefügte oder ausgelassene Grenze werden eine feste Anzahl an Strafpunkten definiert, für jede falsch platzierte Grenze wird der Abstand zur richtigen Position als Fehler definiert ($s(i,j) = |\text{Grenze}_i - \text{Grenze}_j|$). Mit Hilfe der Dynamischen Programmierung wird ein Abgleich der Grenzen bestimmt, in der Art, dass das Gesamtstrafmaß am geringsten ist. Die Einfügerate *Ins* (engl.: insertion-rate) kann bestimmt werden, indem die Anzahl der nicht zugewiesenen Grenzen der automatisch erzeugten Daten (n_{ins}) durch die Gesamtzahl der automatisch generierten Grenzen (n_{test}) dividiert wird:

$$Ins = \frac{n_{\text{ins}}}{n_{\text{test}}} \tag{2.34}$$

Die Löschrate *Del* (engl. deletion-rate) wird analog aus der Anzahl der nicht zugewiesenen Grenzen der korrekten Daten (n_{del}) und der Gesamtzahl der Grenzen der korrekten Daten (n_{ref}) ermittelt:

$$Del = \frac{n_{\text{del}}}{n_{\text{ref}}} \tag{2.35}$$

Zur Beurteilung der Genauigkeit *Acc* der gefundenen Grenzen wird die Summe der Abstände korrespondierender Grenzen $d = \sum s(i,j)$ durch die Anzahl der korrespondierenden Grenzen dividiert. *Acc* gibt den durchschnittlichen zeitlichen Versatz korrespondierender Grenzen in Sekunden an.

$$Acc = \frac{\sum s(i,j)}{n_{\text{Grenzen}}} \tag{2.36}$$

Die Erkennungsfehlerrate E_c kann dann aus der Anzahl der korrekt erkannten ($n_{richtig}$) und falsch erkannten (n_{falsch}) Segmente berechnet werden:

$$E_c = \frac{n_{\text{falsch}}}{n_{\text{richtig}} + n_{\text{falsch}}} \tag{2.37}$$

Falls gewünscht kann auch aus den in diesem Fehlermaß definierten Einzelmaßen eine zusammenfassende Fehlerrate E berechnet werden. Allerdings widerspricht dies der Anforderung, genauere Informationen bezüglich der Segmentierung und Klassifikation getrennt voneinander zur Verfügung zu stellen. Um ein einzelnes Maß zu bieten, kann eine Gesamt-Fehlerrate E dennoch mit Hilfe einer gewichteten Summe berechnet werden, wobei die einzelnen Gewichte vom Anwender festzulegen sind:

$$E = \frac{w_{\text{ins}} \cdot n_{\text{ins}} + w_{\text{del}} \cdot n_{\text{del}} + w_{\text{acc}} \cdot d + w_{\text{falsch}} \cdot n_{\text{falsch}}}{n_{\text{ref}}} \tag{2.38}$$

Dieses Fehlermaß wird in dieser Arbeit mit *M3* bezeichnet.
In Abbildung 2.5 werden die drei vorgestellten Fehlermaße graphisch gegenübergestellt und die unterschiedliche Bewertung verdeutlicht.

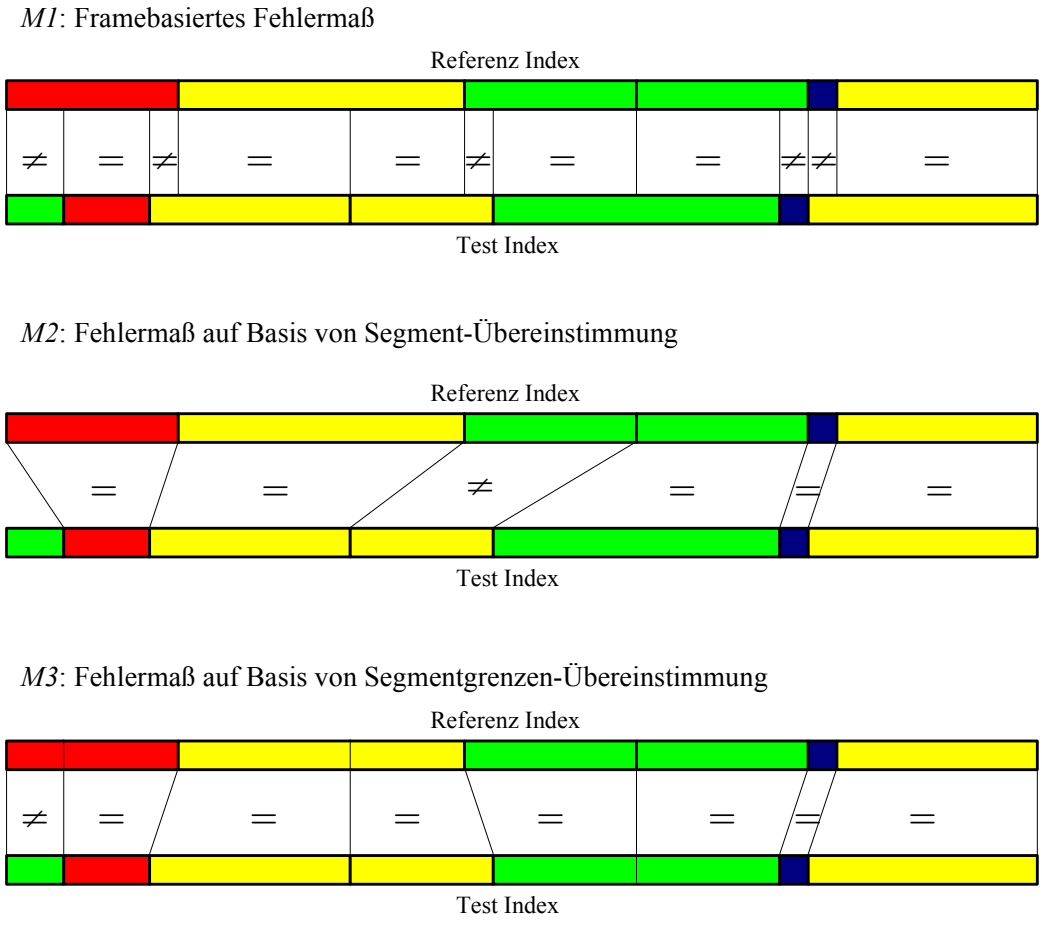


Abbildung 2.5: *Vergleich dreier verschiedener Fehlermaße. Korrespondierende Segmente sind mit gleichen Farben eingefärbt. = bezeichnet eine Übereinstimmung, ≠ eine Nichtübereinstimmung*

Kapitel 3

Segmentierung mit statischen Klassifikationsverfahren

Im folgenden Kapitel werden eine Reihe von Verfahren zur statischen Klassifikation von Mustern vorgestellt. Anschließend wird gezeigt, wie diese Klassifikatoren zur Erkennung und Segmentierung von Gruppenaktionen verwendet werden können. Es wird ferner beschrieben, wie mehrere unterschiedliche Verfahren zu einem leistungsfähigeren Klassifikator kombiniert werden können. Zur Segmentierung werden dann zwei unterschiedliche Verfahren vorgestellt.

3.1 Klassifikation statischer Muster

In diesem Abschnitt werden einige typische und in dieser Arbeit verwendete Verfahren zur Klassifikation statischer Muster oder Merkmale vorgestellt.

3.1.1 Naive-Bayes-Klassifikator

Der Name des *Naive-Bayes*-Klassifikators leitet sich von der Anwendung des Bayes'schen Satzes ab und der stark naiven Annahme, dass beobachtete Merkmale bei gegebener Klasse statistisch voneinander unabhängig sind. Aus diesem Grund wird dieses Model auch *independent feature model* genannt, eine Bezeichnung, die das zugrunde liegende Wahrscheinlichkeitsmodell genauer beschreibt. Numerische Merkmale werden im Allgemeinen als einfache Normalverteilung modelliert [Joh95]. Andere Verteilungen, wie Poisson- oder Gammaverteilungen sind ebenso möglich, aber weniger gebräuchlich. Die Einschränkung der Unabhängigkeit der Merkmale erweist sich häufig als nicht ausreichend, jedoch ist der Naive-Bayes-Klassifikator effizient beim Training und Test und weist eine hohe Transparenz auf. Ziel ist es, die a-posteriori Wahrscheinlichkeit $P(\Omega_\kappa|\boldsymbol{x})$ für einen Vektor $\boldsymbol{x} = (x_1, x_2, \ldots, x_N)^T$ der Klasse Ω_κ zu maximieren. Da diese bedingte Wahrscheinlichkeit direkt nur schwer zu bestimmen ist, kann mit dem Satz von Bayes die Wahrscheinlichkeit wie folgt geschrieben werden:

$$P(\Omega_\kappa|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|\Omega_\kappa) \cdot P(\Omega_\kappa)}{P(\boldsymbol{x})} \tag{3.1}$$

In praktischen Anwendungen ist nur der Zähler von Gleichung 3.1 von Interesse, da der Nenner nicht von der Klasse Ω_κ abhängt und damit für ein gegebenes Muster als konstant betrachtet werden kann. Der Zähler selbst stellt die Verbundwahrscheinlichkeit $P(\Omega_\kappa, \boldsymbol{x})$ dar. Unter Anwendung der Definition der bedingten Wahrscheinlichkeit kann diese Wahrscheinlichkeit geschrieben werden als:

$$\begin{aligned} P(\Omega_\kappa, x_1, x_2, \dots x_N) &= P(\Omega_\kappa) P(x_1, \dots, x_N | \Omega_\kappa) \\ &= P(\Omega_\kappa) P(x_1|\Omega_\kappa) P(x_2, \dots, x_N | \Omega_\kappa, x_1) \\ &= P(\Omega_\kappa) P(x_1|\Omega_\kappa) P(x_2|\Omega_\kappa, x_1) P(x_3, \dots, x_N | \Omega_\kappa, x_1, x_2) \\ &\vdots \end{aligned} \tag{3.2}$$

Aufgrund der statistischen Unabhängigkeit der Merkmale gilt:

$$P(x_i|\Omega_\kappa, x_j) = P(x_i|\Omega_\kappa) \quad \forall i \neq j \tag{3.3}$$

Damit erhält man für die a-posteriori Wahrscheinlichkeit:

$$\begin{aligned} P(x_1, \dots, x_N|\Omega_\kappa) &= P(\Omega_\kappa) P(x_1|\Omega_\kappa) P(x_2|\Omega_\kappa) \dots \\ &= P(\Omega_\kappa) \prod_{i=1}^{N} P(x_i|\Omega_\kappa) \end{aligned} \tag{3.4}$$

Die bedingten Wahrscheinlichkeiten $P(x_i|\Omega_\kappa)$ ergeben sich im diskreten Fall direkt aus der Auszählung der Fälle, in denen das Merkmal x_i einen speziellen Wert einer gegebenen Klasse C_i hat. Die Wahrscheinlichkeit $P(\Omega_\kappa)$ bestimmt sich entweder aus der realen Verteilung der Klassen k, oder es wird eine Gleichverteilung der Klassen angenommen und vereinfachend $P(\Omega_\kappa) = \frac{1}{K}$ gewählt. Für Merkmale mit kontinuierlichem Wertebereich kann auch eine kontinuierliche Wahrscheinlichkeitsdichtefunktion (WDF) $p(x_i = x|\Omega_\kappa)$ zur Modellierung verwendet werden. Im Allgemeinen handelt es sich dabei um eine eindimensionale Normalverteilung $\mathcal{N}(x, \mu_\kappa, \sigma_\kappa)$, die durch die beiden Parameter Mittelpunkt μ_κ und Standardabweichung σ_κ für die jeweilige Klasse Ω_κ definiert ist:

$$p(x_i = x|\Omega_\kappa) = \mathcal{N}(x, \mu_\kappa, \sigma_\kappa) = \frac{1}{\sqrt{2\pi}\sigma_\kappa} \cdot e^{-\frac{(x-\mu_\kappa)^2}{2\sigma_\kappa^2}} \tag{3.5}$$

Zur Entscheidungsfindung in einem Mehrklassenproblem wird die Klasse κ_e mit der höchsten Wahrscheinlichkeit gewählt. Dieses Verfahren wird *Maximum a-posteriori*-Entscheidung (MAP) genannt:

$$\kappa_e = \arg\max_\kappa \{P(\Omega_\kappa) P(\boldsymbol{x}|\Omega_\kappa)\} \tag{3.6}$$

Der Naive-Bayes-Klassifikator lässt sich auch als einfaches Bayessches Netz [Jen96] darstellen. Abbildung 3.1 zeigt den Aufbau eines solchen einfachen Bayesschen Netzes, das als Naive-Bayes-Klassifikator interpretiert werden kann. Diese Darstellung bietet eine einfach zu verstehende, übersichtliche Art, den Naive-Bayes-Klassifikator zu interpretieren. Außerdem können aufgrund der Interpretation als Bayessches Netz alle hier bekannten Verfahren angewendet werden. Auch Erweiterungen sind dadurch problemlos möglich.

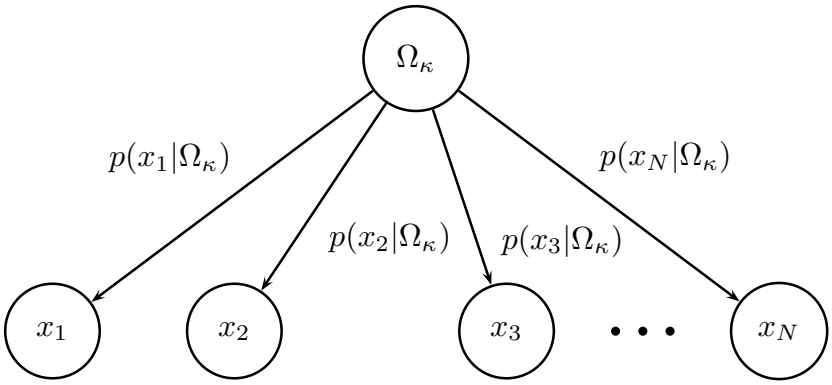


Abbildung 3.1: *Darstellung eines Naive-Bayes-Klassifikators als Bayessches Netz*

3.1.2 Gauß-Mixtur-Modelle

Ein Mixtur-Modell (*Mixture Model*) modelliert eine Wahrscheinlichkeitsverteilung durch additive Überlagerung einer oder mehrerer Einzelverteilungen. Im Unterschied zu dem im vorhergehenden Abschnitt beschriebenen Naive-Bayes-Klassifikator, können mit solch einem Modell auch Abhängigkeiten von Merkmalen untereinander modelliert werden. Mathematisch dargestellt ergibt sich:

$$p(\boldsymbol{x}|\Omega_\kappa) = \sum_{j=1}^{J} w_j h(\boldsymbol{x}|\lambda_j) \tag{3.7}$$

Hierbei bezeichnet $p(\boldsymbol{x}|\Omega_\kappa)$ die zu modellierende Wahrscheinlichkeitsverteilung, J ist die Anzahl der Komponenten (Mixturen) und w_j bezeichnet einen Gewichtskoeffizient, wobei gilt: $0 \leq w_j \leq 1$ und $\sum_{j=1}^{J} w_j = 1$ für alle $j = 1, \ldots, J$. $h(\boldsymbol{x}|\lambda_j)$ bezeichnet eine Wahrscheinlichkeitsverteilung mit einem Parametersatz λ_j. Durch Wahl aller Wahrscheinlichkeitsverteilungen zu Normalverteilungen (vgl. Gleichung (3.5)) ergibt sich ein Gauß-Mixtur-Modell (*Gaussian-Mixture-Modell*, GMM). Für mehrdimensionale Merkmalsvektoren ergibt sich eine multivariate, n-dimensionale Normalverteilung mit dem Mittelwertvektor $\boldsymbol{\mu}$ und der Kovarianzmatrix $\boldsymbol{\Sigma}$:

$$p(\boldsymbol{x}|\Omega_\kappa) = \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_\kappa, \boldsymbol{\sigma}_\kappa) = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}_\kappa|^{1/2}} \exp\left[-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_\kappa)^T \boldsymbol{\Sigma}_\kappa^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_\kappa)\right] \tag{3.8}$$

Wird eine genügend große Anzahl J von Wahrscheinlichkeitsverteilungen, wie sie in Gleichung (3.7) definiert wurden, überlagert, so kann praktisch fast jede gewünschte mehrdimensionale Wahrscheinlichkeitsverteilung nachgebildet werden [Yak70] (vgl. Abb. 3.2). Die Parameter eines GMM lassen sich mit bekannten Trainingsverfahren, wie zum Beispiel dem *Expectation-Maximization*-Algorithmus (EM-Algorithmus) [Dem77], gut bestimmen. GMM können gut generalisieren, solange der Trainingsdatensatz groß genug ist. Das heißt, wenn ausreichend Beispiele vorhanden sind, repräsentieren die gelernten Wahrscheinlichkeitsdichtefunktionen nicht nur die Trainingsdaten, sondern haben auch eine große Aussagekraft für unbekannte Daten. Weiterhin können GMM für beliebig dimensionale Probleme angewendet werden, dazu müssen lediglich multidimensionale WDF trainiert werden.

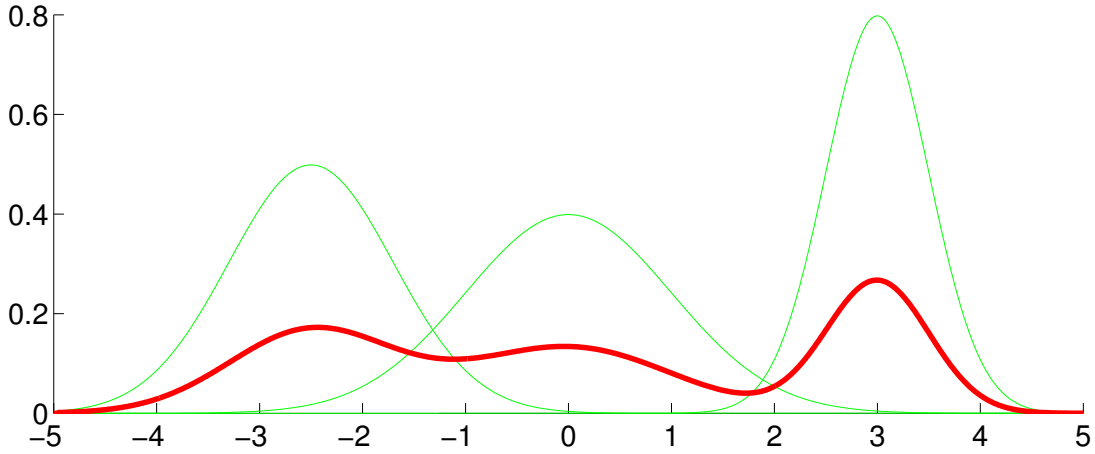


Abbildung 3.2: *Additive Überlagerung mehrerer Normalverteilungen zur Modellierung einer beliebigen Wahrscheinlichkeitsdichtefunktion*

Während der Erkennungsphase sind GMM sehr schnell, da für jedes unbekannte Testdatum lediglich ein posterior Wert einer WDF pro Klasse ausgewertet werden muss, und diese miteinander verglichen werden. Dies lässt sich in der Regel sehr effizient implementieren. GMM werden daher bei vielen Problemen erfolgreich eingesetzt.
Zur Initialisierung der Mittelwerte und Varianzen werden bei unbekannter Klassenzugehörigkeit der Muster Clustering-Verfahren angewendet, wie zum Beispiel k-Means [Mac67]. Danach erfolgt das Training mit dem EM-Algorithmus.
Nachteilig bei der Verwendung von GMM ist allerdings, dass bei zu wenigen Trainingsbeispielen nicht die tatsächliche Verteilung der Daten gelernt wird, sondern lediglich die Trainingsverteilung. Weiterhin handelt es sich bei dem hier verwendeten EM-Training um ein nicht-diskriminatives Verfahren. Beim Lernen wird nicht der Abstand zwischen den Klassen maximiert, sondern jede Wahrscheinlichkeitsdichtefunktion wird für jede Klasse unabhängig optimiert. Das kann für manche Anwendungen nachteilig sein. Es wurden allerdings Verfahren für ein diskriminatives Lernen entwickelt, die diese Nachteile vermeiden sollen [Ala96].

3.1.3 Künstliche Neuronale Netze

Künstliche Neuronale Netze (*Artificial Neural Networks*, KNN) sind ein mächtiges Werkzeug für die Verarbeitung von Daten. Neuronale Netze können aufgrund ihrer weitreichenden Fähigkeiten vielfältig eingesetzt werden, angefangen von der Erkennung von Sprache (vgl. z.B. [Teb95], [Rig96b]) über die Gesichtserkennung [Law97], [Wal04c], [Wal06] bis zur Analyse von Genom-Strukturen [Pet05], [All99]. Nach Jain [Jai96] kann die Funktionalität von KNN in folgende Bereiche eingeteilt werden:

Mustererkennung: Die Aufgabe der Klassifikation besteht darin, einem Eingangsmuster, repräsentiert durch Merkmalvektoren, eine von mehreren zuvor definierten Klassen zuzuordnen. Bekannte Anwendungen beinhalten die Einzelzeichenerkennung, Erkennung von einzelnen Phonemen und medizinische Anwendungen, wie beispielsweise

die Klassifikation von Blutzellen.

Clustering: Bei unbekannter Klassenzugehörigkeit der Trainingsdaten untersuchen KNN die Ähnlichkeit der einzelnen Muster und nehmen dementsprechend eine Gruppeneinteilung vor. Bekannte Anwendungen sind die Datenkompression und die informatorische Datenanalyse.

Funktionsapproximation: Für einen Satz von n Trainingsdaten $(\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n})$ wird angenommen, dass eine Funktion $\mu(\boldsymbol{x})$ die zugehörigen bekannten Funktionswerte $(y_1, y_2, \ldots, y_n)$ erzeugt hat. Die Aufgabe besteht darin, eine Funktion $\hat{\mu}$ zu finden, die die unbekannte Funktion μ möglichst gut approximiert.

Prädiktion: Aus einer Anzahl von bekannten Werten einer Wertereihe, soll auf den nächsten unbekannten Wert einer Größe geschlossen werden. Solche Modelle werden unter anderem bei der Wettervorhersage verwendet.

Optimierung: Ziel ist es, eine möglichst optimale Lösung zu finden, die gegebene Bedingungen berücksichtigt, so dass die Zielfunktion maximal oder minimal wird. Das klassische Beispiel für diese Anwendung ist die Minimierung des Weges eines Handlungsreisenden (Travelling Salesman Problem (TSP)).

Assoziativspeicher: Diese Art von Speicher ermöglichen es dem Anwender, auf bestimmte Adressen bzw. Zellen über den Inhalt zuzugreifen anstelle eines Index. Auf diese Weise können unvollständige oder verrauschte Muster zuverlässig abgerufen werden.

Regelung: Neuronale Netze können auch zur Steuerung eines Regelkreises eingesetzt werden, zum Beispiel bei der adaptiven Geschwindigkeitskontrolle.

Die ersten Überlegungen zu Neuronalen Netzen und deren mathematische Beschreibung gehen auf McCulloch und Pitts [McC43] im Jahre 1943 zurück. Grundlage dieser Arbeit ist der prinzipielle Aufbau des zentralen Nervensystems von Wirbeltieren. Die elementare Einheit zur Verarbeitung von Information ist dabei das Neuron, das über sein zugehöriges Axon und die Synapse elektrische Impulse an benachbarte Neuronen sendet [Rig01a]. Jedes Neuron ist im Mittel mit 10^3 bis 10^4 anderen Neuronen verknüpft. Ein Impuls entsteht, wenn das Potential in der Zelle des Neurons über einen Schwellwert steigt. Das Neuron „feuert“. Die Impulsrate ist dabei abhängig von der kumulativen Eingangserregung. Ein Neuronales Netz besteht also aus Neuronen und ihren gerichteten Verbindungen. Von besonderer Bedeutung für den technischen Einsatz ist die Lernfähigkeit und die massive Prallelität der Neuronen. Dadurch wird eine Klassifikation in hoher Geschwindigkeit ermöglicht.
Die grundlegende Architektur Neuronaler Netze besteht im Allgemeinen aus drei Schichten (sogenannte *Layer*), nämlich einer Eingangsschicht (*Input-Layer*), einer Ausgangsschicht (*Output-Layer*), sowie einer verdeckten Schicht (*Hidden-Layer*) [Rig94]. Die Anzahl der Neuronen in jeder Schicht variiert dabei je nach Aufgabe der KNN und kann gerade in der verdeckten Schicht sehr groß sein. Zur mathematischen Beschreibung der Neuronen und deren Beziehungen schlagen McCulloch und Pitts [McC43] ein einfaches Modell

mit binärem Schwellwert vor. Generell bestehen die im Rechner modellierten Neuronen analog zu ihrem biologischen Vorbild aus einem das Eingangsmuster beschreibenden Vektor $\boldsymbol{x}$, einem Gewichtungsvektor $\boldsymbol{w}$, einem Bias-Wert w_0, einer Propagierungsfunktion $G(x)$, einer meist nichtlinearen Aktivierungsfunktion $F(s) = F(G(\boldsymbol{x}))$ und einer Ausgangsgröße y. Ein solches beispielhafte Neuron ist in Abbildung 3.3 dargestellt. Um eine

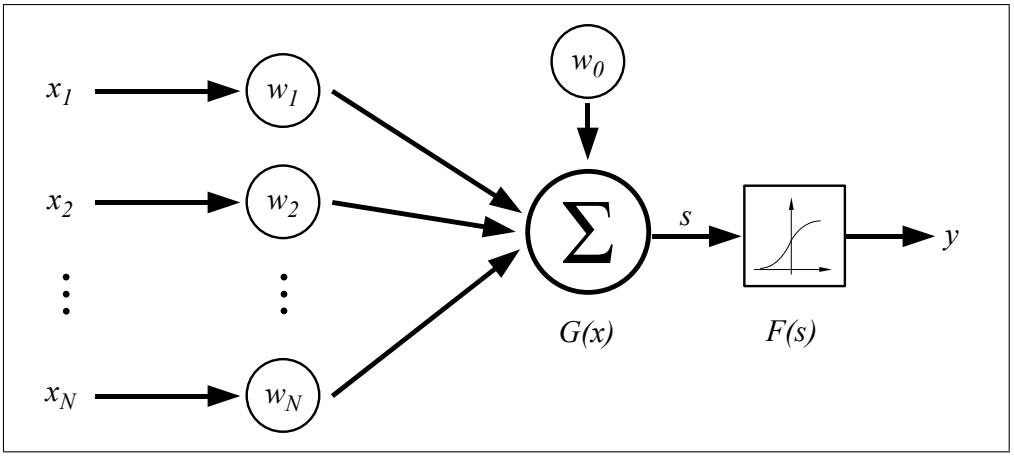


Abbildung 3.3: *Mathematisches Modell eines künstlichen Neurons*

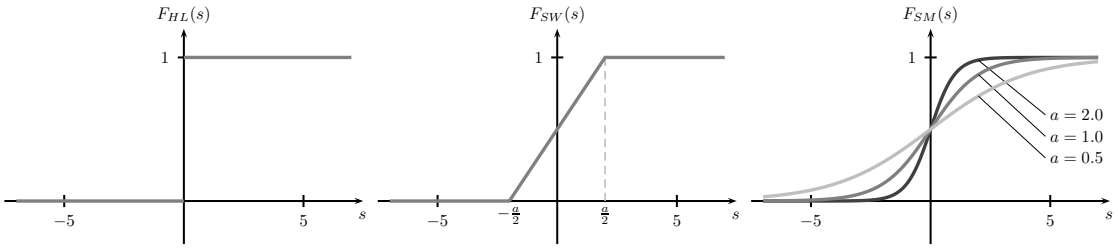
vereinfachte Schreibweise zu erhalten wird der Bias-Wert w_0 in den Gewichtsvektor integriert: $\boldsymbol{w}^T = (w_0, w_1, \ldots, w_N)$. Dementsprechend muss auch der Eingangsvektor mit dem konstanten Wert $x_0 = 1$ erweitert werden, so dass gilt: $\boldsymbol{x}^T = (x_0, x_1, \ldots, x_N)$. Die Propagierungsfunktion $G(x)$ ist im allgemeinen eine gewichtete Summe der Eingänge, selten eine Multiplikationsfunktion, oder eine Funktion, die den euklidischen Abstand zwischen Gewichtungen und Eingangswerten berechnet:

$$s = G(\boldsymbol{x}) = \boldsymbol{w}^T \cdot \boldsymbol{x} = \sum_{i=1}^{N} w_i \cdot x_i + w_0 = \sum_{i=0}^{N} w_i \cdot x_i \tag{3.9}$$

Die Aktivierungsfunktion hat als Variable den Ausgang s der Propagierungsfunktion $G(x)$. Den Ausgang y eines Neurons kann man dann für die hier gewählte Propagierungsfunktion wie folgt angeben:

$$y = F(s) = F(G(\boldsymbol{x})) = F\left(\sum_{i=0}^{N} w_i \cdot x_i\right) \tag{3.10}$$

Typischerweise hat die Aktivierungsfunktion einen Wertebereich zwischen 0 und 1, wobei $\lim_{s\to-\infty} F(s) = 0$ und $\lim_{s\to\infty} F(s) = 1$. Funktionen, die diese gewünschten Anforderungen erfüllen, sind zum Beispiel der harte Begrenzer (HL), eine Schwellwertfunktion (SW), oder eine Sigmoid- (SM)oder Tangenshyperbolikusfunktion. Beispiele solcher Funktionen sind in Abbildung 3.4 dargestellt. Es stehen sowohl stetige als auch nicht stetig differenzierbare Funktionen zur Verfügung. Die Wahl der verwendeten Aktivierungsfunktion hängt dabei von der jeweiligen Aufgabenstellung ab.



$$F_{HL} = \begin{cases} 0 & s \leq 0 \\ 1 & s > 0 \end{cases} \qquad F_{SW} = \begin{cases} 0 & s < -\frac{a}{2} \\ \frac{s}{a} + \frac{1}{2} & |s| \leq \frac{a}{2} \\ 1 & s > \frac{a}{2} \end{cases} \qquad F_{SM} = \frac{1}{1 + e^{-as}} \quad (3.11)$$

Harter Begrenzer | Schwellwert-Funktion | Sigmoid-Funktion

Abbildung 3.4: *Beispiele von Aktivierungsfunktionen in Neuronen*

Mehrschicht-Perzeptron-Netzwerke

Die Architektur eines Neuronalen Netzes ist im Allgemeinen frei wählbar. Grundsätzlich lassen sich zwei grundlegende Architekturen unterscheiden: Die am meisten verwendete Anordnung ist die der Feedforward-Netze. Mehrere Neuronen werden parallel und in mehreren Schichten hintereinander angeordnet. Verbindungen bestehen nur zwischen den Neuronen in eine Richtung, nämlich von einer Schicht zu einer nächst höher (im Sinne von näher an der Ausgansschicht) gelegenen Schicht. Rückwärtsgerichtete Verbindungen sowie Verbindungen zwischen Neuronen derselben Schicht existieren nicht. Als Neuronen werden Perzeptronen verwendet. Aus diesem Grund spricht man von einem Mehrschicht-Perzeptron (*Multilayer-Perceptron*, MLP), das auf dem Gebiet der Mustererkennung weit verbreitet ist und häufig eingesetzt wird. In Abbildung 3.5 ist der prinzipielle Aufbau eines MLP gezeigt. Die unterste Schicht ist die Eingangsschicht und besteht aus N Knoten für die Eingabewerte $x_i,\ i = 1, \ldots, N$ des Merkmalsvektors $\boldsymbol{x}$. Die Eingangsschicht dient nur zur Präsentation des Musters und beinhaltet noch keine Verarbeitung (Gewichtungen oder Summenbildungen). Danach können mehrere verborgene Schichten folgen, in denen die für neuronale Netze typischen Summenbildungen der gewichteten Eingänge stattfinden. Es kann jedoch gezeigt werden, dass für die Definition beliebiger Klassengrenzen im $\mathbb{R}^N$ eine verborgene Schicht ausreichend ist [Rig94]. Die oberste Schicht ist die Ausgangsschicht. An den Ausgängen der Neuronen $y_j,\ j = 1, \ldots, M$, zusammengefasst im Augsgangsvektor $\boldsymbol{y}$, kann man das Ergebnis der Klassifikation ablesen. Zur Zuordnung eines Eingangsmusters zu einer bestimmten Klasse $\Omega_\kappa,\ \kappa = 1, \ldots, K$ muss der Ausgang des KNN entsprechen kodiert werden. Für den Fall von K Klassen werden zum Beispiel auch K Neuronen in der Ausgangsschicht verwendet. Jedes Neuron repräsentiert somit eine Klasse. Da der Wertebereich der Neuronen zwischen 0 und 1 liegt, bedeutet ein Ausgangswert y_i eines Neurons nahe 1 eine bessere Entsprechung zur Klasse κ_i. Für den Fall, dass mehrere Ausgänge hohe Werte annehmen, kann die erkannte Klasse durch eine

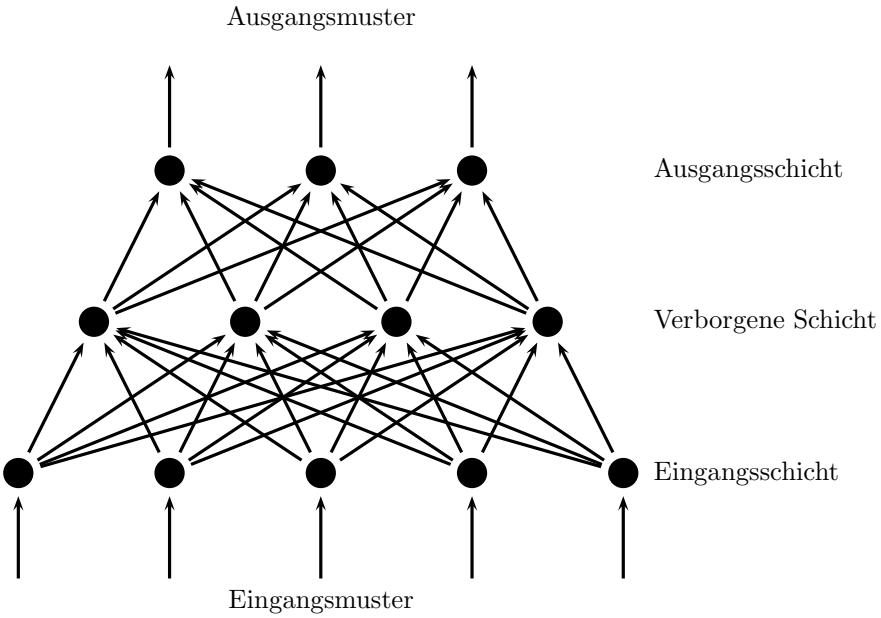


Abbildung 3.5: *Architektur eines Mehrschicht-Perzeptrons mit einer verborgenen Schicht*

Maximumsuche gefunden werden:

$$\kappa_e = \arg\max_{\kappa} y_\kappa \tag{3.12}$$

Soll der Ausgangswert eine Bewertung der Sicherheit der gewählten Klasse beinhalten, so kann man als Aktivierungsfunktion in den Augangsneuronen die sogenannte Softmax-Funktion einsetzen. Diese normiert die Summe der Ausgänge zu eins, und somit können diese als a-posteriori Klassenwahrscheinlichkeiten $P(\Omega_\kappa|\boldsymbol{x})$ interpretiert werden:

$$P(\Omega_\kappa|\boldsymbol{x}) = y_\kappa = \frac{e^{s_\kappa}}{\sum_{j=1}^{K} e^{s_j}} \tag{3.13}$$

Die erkannte Klasse κ_e kann dann wiederum durch Maximumsuche bestimmt werden:

$$\kappa_e = \arg\max_{\kappa} y_\kappa = \arg\max_{\kappa} P(\Omega_\kappa|\boldsymbol{x}) \tag{3.14}$$

Die Bestimmung optimaler Parameter für eine Klassifikation, in diesem Fall der Gewichte des Netzes, erfolgt mit Hilfe eines Lernverfahrens. Dabei werden optimale Gewichtungen aus Lernbeispielen bestimmt.
Aus der Vielzahl der existierenden Lernverfahren wird hier das weit verbreitete Backpropagation-Lernen vorgestellt, ein Gradientenabstiegsverfahren, das zur Gruppe der überwachten Lernverfahren gehört und als Erweiterung der Delta-Regel [Rig94] auf mehrschichtige Netze angewendet wird. Zur leichteren mathematischen Beschreibung des Lernverfahrens können die Gewichte $w_{i,j}$ einer Schicht mit M Neuronen, die jeweils N Gewichte besitzen,

in einer Matrix $\boldsymbol{W}$ zusammengefasst werden:

$$\boldsymbol{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_M] = \begin{bmatrix} w_{1,1} & \cdots & w_{1,M} \\ \vdots & \ddots & \vdots \\ w_{N,1} & \cdots & w_{N,M} \end{bmatrix} \tag{3.15}$$

Hierbei beschreibt das Gewicht $w_{i,j}$ die Gewichtung zwischen dem i-ten Neuron der unteren Schicht und dem j-ten Neuron der oberen Schicht. Damit lassen sich die Ausgänge einer Schicht kompakt schreiben als:

$$\boldsymbol{y}^T = [y_1, y_2, \ldots, y_M] = F(\boldsymbol{x}^T \cdot \boldsymbol{W}) \tag{3.16}$$

Gilt Gleichung (3.16) nur für eine Schicht, so können die Ausgänge einer darüberliegenden Schicht mit Hilfe der Form aus Gleichung (3.17) beschrieben werden:

$$\boldsymbol{y}_2^T = F(\boldsymbol{y}_1^T \cdot \boldsymbol{W}_2) = F(F(\boldsymbol{x}^T \cdot \boldsymbol{W}_1) \cdot \boldsymbol{W}_2) \tag{3.17}$$

Ziel des Lernvorgangs ist es, die freien Parameter des Netzes so anzupassen, dass die erzielten Ausgangswerte bei gegebenem Eingang möglichst nahe an die gewünschten Ausgangswerte herankommen. Zur Bewertung verwendet man üblicherweise den quadratischen Fehler ϵ_{ges}, der sich aus der Summe der Einzelfehler $\epsilon_n = \frac{1}{2}\sum_j (\hat{y}_j - y_j)^2$ der einzelnen Muster n ergibt:

$$\epsilon_{\text{ges}} = \sum_n \epsilon_n = \frac{1}{2} \sum_n \sum_j (\hat{y}_j - y_j)^2 \tag{3.18}$$

Hierbei ist $\hat{y}_j$ der gewünschte Ausgang eines Musters, y_j der tatsächlich erhaltene Ausgang. Um den Wert der Änderung für jedes Gewicht zu erhalten, muss der Gradient von Gleichung (3.18) bezüglich jedes Gewichts gebildet werden. Für die Gewichte in der Ausgangsschicht ist dies direkt möglich, für die der darunter liegenden Schichten muss der Fehler zurückpropagiert werden[1]. Der ganze Backpropagation-Algorithmus kann in sieben Schritte unterteilt werden [Jai96]:

1. Initialisierung der Gewichte mit kleinen Zufallswerten
2. Zufällige Auswahl eines Trainingsmusters $\boldsymbol{x}^{(\mu)}$ aus den Trainingsdaten
3. Vorwärtspropagierung des Signals durch das Netzwerk
4. Berechnung von δ_i^L in der Ausgangsschicht:

$$\delta_i^L = \frac{\partial F(s_i^L)}{\partial s_i^L}[\hat{y}_i^\mu - y_i^L] \tag{3.19}$$

5. Berechnung der δ_i^l in den darunterliegenden Schichten durch Fehler-Propagierung:

$$\delta_i^l = \frac{\partial F(s_i^l)}{\partial s_i^l} \sum_j w_{ij}^{l+1} \delta_j^{l+1} \quad \text{mit } l = (L-1), \ldots, 1 \tag{3.20}$$

[1] Daher der Name *Backpropagation*

6. Aktualisieren der Gewichte w_{ij}^l mit der Lernrate η nach folgender Regel:

$$\Delta w_{ij}^l = \eta \, \delta_i^l \, y_j^{l-1} \tag{3.21}$$

7. Wiederholung ab Schritt 2 für das nächste Muster, solange bis der Fehler der Ausgangsschicht unter einem Schwellenwert liegt, oder eine maximale Anzahl an Iterationen überschritten wird.

Eine ausführliche Herleitung des Algorithmus ist z. B. in [Rig94], [Hay94] und [Sch97a] zu finden. Zur Optimierung der Geschwindigkeit oder Fehlerminimierung wurden Varianten des Backpropagation-Verfahrens entwickelt, wie zum Beispiel die Einführung von Momentum-Termen [Rig94], oder der Resilient Propagation Algorithmus (RPROP), der von Riedmiller und Braun entwickelt wurde [Rie93], [Ige00].

Radiale-Basisfunktionen-Netze

Ähnlich wie bei den Mehrschicht-Perzeptron-Netzen (siehe oben) besteht die Aufgabe der Radialen-Basisfunktionen-Netze (RBF-Netze) darin, Klassifikationsprobleme zu lösen, die nicht linear trennbar sind. RBF-Netze bestehen aus einer Eingangsschicht, einer verdeckten Schicht und einer Ausgangsschicht. Die Neuronen in der verdeckten Schicht werden mit radialen Basisfunktionen modelliert. Die Ausgangsschicht ist linear, die verdeckte Schicht durch die Verwendung von RBF hochgradig nichtlinear. Besteht die verdeckte Schicht aus mehr Neuronen als die Eingangsschicht, so kann man die Ausgänge der verdeckten Schicht als eine Abbildung der (möglicherweise) niedrigdimensionalen Eingänge in einen hochdimensionalen Merkmalsraum interpretieren. Cover hat dargelegt, dass ein komplexes Musterklassifizierungsproblem, das im Ursprungsraum nicht linear trennbar ist, eher in einem hochdimensionalen Raum linear trennbar ist [Cov65]. Aus diesem Grund ist die Ausgangsschicht linear gestaltet. Die am meisten verbreitete radiale Basisfunktion ist die Gauß-Funkion [Rig01b]. Damit ergibt sich als Aktivierungsfunktion:

$$F(\boldsymbol{x}) = e^{-\frac{\|\boldsymbol{x}-\boldsymbol{w}\|}{2\sigma^2}} \tag{3.22}$$

Besteht die verdeckte Schicht aus K Neuronen, ergibt sich für die Aktivierung des n-ten Ausgangs:

$$y_n = \sum_{k=1}^{K} w_{kn} \cdot e^{-\frac{(\boldsymbol{x}-\boldsymbol{w}_{kn})^T \cdot (\boldsymbol{x}-\boldsymbol{w}_{kn})}{2\sigma^2}} \tag{3.23}$$

Hierbei ist $\boldsymbol{x}$ der Eingangsvektor, $\boldsymbol{w}_k$ der Vektor der Gewichtungen von den Eingangsneuronen zum k-ten Neuron der verdeckten Schicht, w_{kn} die Gewichtung vom k-ten Neuron der verdeckten Schicht zum n-ten Neuron der Ausgangsschicht und σ die Streuung. Die Bestimmung der Parameter erfolgt meist mit Hilfe des Least Squares Verfahrens, das u. a. in [Rig01b] und [Hay94] beschrieben ist und auf das hier nicht näher eingegangen werden soll.
Obwohl sich MLP- und RBF-Netze sehr ähnlich sind, gibt es doch mehrere Unterschiede [Hay94]:

- Ein RBF-Netz hat genau eine verborgene Schicht, wohingegen ein MLP auch mehrere verborgene Schichten besitzen kann.
- Typischerweise sind die Neuronen eines MLP in jeder Schicht von gleichem Aufbau. In einem RBF-Netz sind die Neuronen in der verborgenen Schicht von gänzlich anderer Struktur als in der Augsgangsschicht.
- Die verborgene Schicht eines RBF-Netzes ist nichtlinear, die Ausgangsschicht ist linear. Bei einem MLP sind im Normalfall alle Schichten nichtlinear.
- Die Aktivierungsfunktion eines Neurons eines RBF berechnet die euklidische Norm zwischen dem Eingangsvektor und dem Zentrum dieses Neurons. Bei einem MLP wird in der Propagierungsfunktion das innere Produkt des Eingangsvektors und des Gewichtsvektors berechnet.

3.1.4 Support-Vektor-Maschinen

Das Klassifizieren von Mustern mit Hilfe von Support-Vektor-Maschinen (SVM) hat in den letzten Jahren immer mehr an Bedeutung zugenommen [Bur98]. In vielen Bereichen der Mustererkennung werden SVM mit Erfolg eingesetzt, wie zum Beispiel für die Erkennung isolierter handgeschriebener Ziffern [Cor95], zur Sprecheridentifikation [Sch96], zur Detektion von Gesichtern in Bildern [Osu97], in der automatischen Spracherkennung [Sta06] oder zur Kategorisierung von Text [Joa98].

Generalisierungsfähigkeit eines Klassifikators

Die Theorie und mathematische Beschreibung der SVM beruht auf der statistischen Lerntheorie, die von Vapnik ([Vap98] und [Vap00]) ausführlich beschrieben und hergeleitet wird. Hier soll nur knapp auf die Ergebnisse eingegangen werden. Eine detaillierte Beschreibung der Theorie ist auch in einem Tutorial zu SVM [Bur98] zu finden.
Vorhanden sei eine Menge von Merkmalsvektoren $\boldsymbol{x}_i$, $i = 1, \ldots, N$ und die dazugehörigen entsprechenden Klassenzugehörigkeiten $y_i \in \{-1,1\}$. Es wird also nur ein Zweiklassenproblem betrachtet. Der Klassifikator definiert eine Funktion $f(\boldsymbol{x}, \vartheta)$, mit den Eingangsdaten $\boldsymbol{x}$ und den trainierten Parametern ϑ, mit dessen Ausgang die Klassenzugehörigkeit der Eingangsdaten bestimmt werden kann. Im Falle eines Neuronalen Netzes entspricht das Parameterset ϑ den Gewichten aller Schichten des Netzes.
Das empirische Risiko R_{emp} ist definiert als der mittlere Fehler des Klassifikators auf dem Trainingsset [Bur98]:

$$R_{\text{emp}}(\vartheta) = \frac{1}{2N} \sum_{i=1}^{N} |y_i - f(\boldsymbol{x}_i, \vartheta)| \tag{3.24}$$

Für ein binäres Klassifikationsproblem, also $y_i \in \{-1,1\}$ und $f(\boldsymbol{x}_i, \vartheta) \in [-1,1]$, gilt für ein gewähltes η $(0 \leq \eta \leq 1)$ für das zu erwartende Risiko $R(\vartheta)$ [Vap00]:

$$R(\vartheta) \leq R_{\text{emp}}(\vartheta) + \sqrt{\frac{h(\log(2N/h) + 1) - \log(\eta/4)}{N}} \tag{3.25}$$

Hierbei bezeichnet h die sogenannte Vapnik-Chervonenkis (VC) Dimension. Sie ist gleich der Maxmimalzahl von Punkten im Merkmalsraum ($\mathbb{R}^n$), die mit dem jeweiligen Klassifikator auf beliebige Art getrennt werden kann.
Ziel ist es (vgl. [Vap00]), denjenigen Klassifikator zu finden, der die niedrigste obere Schranke für das zu erwartende Risiko $R(\vartheta)$ (Gleichung (3.25), rechte Seite) generiert. Eine mögliche Strategie besteht darin, denjenigen Klassifikator auszuwählen, der die geringste VC-Dimension besitzt.

Linear trennbare Daten

Der einfachste Fall einer Mustertrennung besteht, wenn Daten $\boldsymbol{x}_i$ mit Klassenzugehörigkeit y_i linear trennbar sind. Dann ergibt sich für einen linearen Klassifikator mit den Parametern $\boldsymbol{w}$ und b folgende Bedingungen:

$$\boldsymbol{x} \cdot \boldsymbol{w} + b \geq +1 \quad \text{für } y = +1 \tag{3.26}$$

$$\boldsymbol{x} \cdot \boldsymbol{w} + b \leq -1 \quad \text{für } y = -1 \tag{3.27}$$

$\boldsymbol{w}$ bezeichnet dabei den Normalenvektor zur der Trennebene $\mathcal{H}$, $\frac{|b|}{\|\boldsymbol{w}\|}$ ist der Abstand der Trennebene $\mathcal{H}$ vom Ursprung. Beide Bedingungen können in einer Ungleichung zusammengefasst werden:

$$y_i(\boldsymbol{x}_i \cdot \boldsymbol{w} + b) - 1 \geq 0 \quad \forall i \tag{3.28}$$

In diesem Fall existiert eine Hyperebene $\mathcal{H}$, die die positiven von den negativen Mustern trennt. Punkte, die die Gleichheit von Ungleichung (3.26) erfüllen, liegen auf einer Hyperebene $\mathcal{H}_1 : \boldsymbol{x}_i \cdot \boldsymbol{w} + b = 1$ mit dem Normalenvektor $\boldsymbol{w}$ und dem Abstand $|1 - b|/\|\boldsymbol{w}\|$ vom Ursprung. Genauso liegen Punkte, die die Gleichheit von Ungleichung (3.27) erfüllen, auf einer Hyperebene $\mathcal{H}_2 : \boldsymbol{x}_i \cdot \boldsymbol{w} + b = -1$ mit dem gleichen Normalenvektor $\boldsymbol{w}$ und dem Abstand $|-1-b|/\|\boldsymbol{w}\|$ vom Ursprung. Der Abstand der beiden Hyperebenen (Margin) ist dann $\frac{2}{\|\boldsymbol{w}\|}$. Die beste Trennung der zwei Klassen wird erreicht, wenn der Abstand der Hyperebenen maximal wird. Daher ist es die Aufgabe der Optimierung, den Abstand durch Minimierung von $\|\boldsymbol{w}\|^2$ unter der Berücksichtigung der linearen Randbedingungen von Gleichung (3.28) zu maximieren [Bur98]. Zur Lösung dieses Randwertproblems werden im Allgemeinen Lagrange-Multiplikatoren α_i eingesetzt, was zu folgender zu minimierender Gleichung führt:

$$L = \frac{1}{2}\|\boldsymbol{w}\|^2 - \sum_{i=1}^{N} \alpha_i y_i (\boldsymbol{x}_i \cdot \boldsymbol{w} + b) + \sum_{i=1}^{N} \alpha_i \tag{3.29}$$

Für eine einfachere Berechnung kann unter Berücksichtigung der Bedingungen, die sich bei der Gradientenbildung ergeben ($\frac{\partial L}{\partial \boldsymbol{w}} = 0$ und $\frac{\partial L}{\partial b} = 0$), das folgende duale Problem formuliert und dann minimiert werden [Bur98]:

$$L_D = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j \boldsymbol{x}_i \boldsymbol{x}_j \tag{3.30}$$

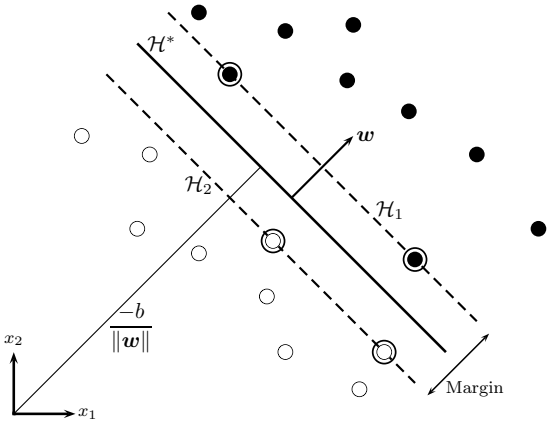


Abbildung 3.6: *Hyperebenen maximale Trenngüte und Mustervektoren in einem zweidimensionalen Raum mit einer linear separierbaren Verteilung*

Zusammen mit den Karush-Kuhn-Tucker Bedingungen kann so eine optimale Lösung gefunden werden. Das genaue Vorgehen ist ausführlich in der Literatur beschrieben [Vap00], [Bur98], [Sch01b].
Trainingsmuster, die nach der Optimierung auf den Ebenen $\mathcal{H}_1$ oder $\mathcal{H}_2$ liegen, werden Stütz-Vektoren oder Support-Vektoren genannt. Diese sind in Abbildung 3.6 eingekreist. Für sie gilt auch, dass die Lagrange-Multiplikatoren $\alpha_i > 0$ sind [Nie03].
Die Klassifikation eines unbekannten Musters $\boldsymbol{x}$ erfolgt nach der Berechnung der optimalen Trennebene durch die Auswertung, auf welcher Seite der Ebene sich das Muster befindet. Dies geschieht durch Bestimmung des Vorzeichens der Bedingungen nach den Gleichungen (3.26) und (3.27), was zu folgender Entscheidungsregel führt:

$$\boldsymbol{x} \in \begin{cases} \Omega_1 & \text{falls} \quad \boldsymbol{w} \cdot \boldsymbol{x} + b \geq 0 \\ \Omega_2 & \text{falls} \quad \boldsymbol{w} \cdot \boldsymbol{x} + b < 0 \end{cases} \tag{3.31}$$

Erweiterung auf nicht linear trennbare Daten

Im Allgemeinen kann man allerdings nicht davon ausgehen, dass die Mustervektoren linear trennbar sind. Daher müssen die obigen Bedingungen erweitert werden, so dass eine in gewissem Sinne optimale Lösung immer noch gefunden werden kann. Dazu führt man sogenannte Schlupfvariablen ξ_i ein:

$$\boldsymbol{x} \cdot \boldsymbol{w} + b \geq +1 - \xi_i \quad \text{für } y = +1 \tag{3.32}$$

$$\boldsymbol{x} \cdot \boldsymbol{w} + b \leq -1 + \xi_i \quad \text{für } y = -1 \tag{3.33}$$

$$\xi_i \geq 0 \quad \forall i \tag{3.34}$$

Dieses Klassifikationsproblem ist in Abbildung 3.7 dargestellt.
Trainingsfehler können jetzt unter Berücksichtigung von zusätzlichen Kosten ξ_i behandelt werden, ohne den im vorherigen Abschnitt vorgestellten Ansatz komplett neu formulieren zu müssen. Die optimale Trennebene kann wie vorhin mit zusätzlichen Lagrange-Multiplikatoren und unter Berücksichtigung einer Fehlerabschätzung $\sum_m \xi_m$ gefunden

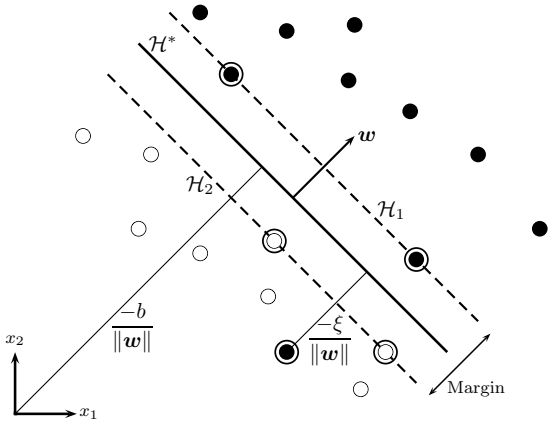


Abbildung 3.7: *Hyperebenen maximale Trenngüte und Mustervektoren in einem zweidimensionalen Raum einer nicht linear separierbaren Verteilung*

werden. Das duale Problem bleibt dabei identisch zu dem im linear trennbaren Fall, allerdings nun mit den Randbedingungen $0 \leq \alpha_i \leq C$ und $\sum_i \alpha_i y_i = 0$, wobei C ein einstellbarer Parameter zur Bestrafung von Trainingsfehlern ist. Auch dieses Vorgehen ist ausführlich in der einschlägigen Literatur beschrieben [Vap00], [Bur98], [Sch01b].

Erweiterung auf nichtlineare Trennfunktionen

Die bisher vorgestellten Lösungsansätze erzeugen immer eine lineare Trennfunktion im Mustervektorraum. Um dennoch nichtlineare Probleme optimal separieren zu können, kann man die Mustervektoren mit Hilfe einer Mapping-Funktion Φ in einen – in der Regel höherdimensionalen – Euklidischen Raum transformieren.

$$\Phi : \mathbb{R}^d \rightarrow \mathbb{R}^{d'} \tag{3.35}$$

Es wird nun eine sogenannte Kernel-Funktion $K(\boldsymbol{x}_1, \boldsymbol{x}_2)$ konstruiert, so dass gilt:

$$K(\boldsymbol{x}_1, \boldsymbol{x}_2) = \Phi(\boldsymbol{x}_1) \cdot \Phi(\boldsymbol{x}_2) \tag{3.36}$$

Da sowohl im Optimierungsproblem (Gleichung (3.30)) als auch im Entscheidungsprozess (Gleichung (3.31)) die Daten nur als Skalarprodukt erscheinen, ist es möglich, die Lösungsansätze aus dem linearen Fall anzuwenden und in dem transformierten hochdimensionalen Raum eine Hyperebene zu finden, die die Datenpunkte linear trennt. Geeignete Kernel-Funktionen können mit Hilfe der Mercer-Bedingungen [Vap00], [Sch01b] gefunden werden. Bekannte, in der Praxis gebräuchliche Funktionen sind:

- Polynomial-Kernel: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = (\boldsymbol{x}_i \cdot \boldsymbol{x}_j + 1)^p$ wobei p den Grad des Polynoms angibt,
- RBF-Kernel / Gauß-Kernel: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = e^{-\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2 / 2\sigma^2}$, mit σ als Standardabweichung der Gaußverteilung,
- Sigmoid-Kernel: $K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \tanh(c\, \boldsymbol{x}_i \cdot \boldsymbol{x}_j - \delta)$, wobei c die Verstärkung, und δ den Offset angibt.

Bei der Entscheidung, welcher Klasse ein unbekanntes Muster $\boldsymbol{x}$ zuzuordnen ist, kommt folgende Entscheidungsfunktion zum Einsatz:

$$y = \sum_{i=1}^{N_s} \alpha_i y(i) K(\boldsymbol{s}_i, \boldsymbol{x}) \tag{3.37}$$

Das Vorzeichen, $\mathrm{sgn}(y)$, bestimmt die Klassenzugehörigkeit, während der Betrag $|y|$ den Abstand zur Hyperebene angibt. $\boldsymbol{s}_i$ bezeichnet diejenigen Trainigsvektoren, die als Support-Vektoren bestimmt wurden.

Erweiterung auf Mehrklassenprobleme

Support-Vektor-Maschinen wurden ursprünglich so konstruiert, dass sie nur ein Zweiklassenproblem lösen können. Für die Trennung von mehr als zwei Klassen gibt es unter anderem folgende Vorschläge [Nie03]:

- „einer gegen alle“: Hier wird das Problem auf k Zweiklassenprobleme zurückgeführt. Für jede Klasse wird ein Klassifikator trainiert, der diese Klasse gegen alle $k-1$ verbleibenden Klassen unterscheidet.

- „ jeder gegen jeden“: Bei diesem Ansatz wird jede Klasse gegen jede andere Klasse differenziert. Es werden also $k(k-1)/2$ Klassifikatoren trainiert. Die endgültige Entscheidung kann durch Zählen der positiven Einzelentscheidungen für eine Klasse getroffen werden. Der Rechenaufwand ist bei dieser Variante höher als in der ersten, da deutlich mehr Klassifikatoren trainiert werden müssen, allerdings wird dieser Aufwand durch die deutlich besseren Ergebnisse gerechtfertigt [Nie03].

- Multi-Layer-SVM / SVM-Bäume: Hier wird versucht, mit Hilfe von Entscheidungsbäumen jeweils zwei Bereiche voneinander zu trennen. Sukzessive wird dabei vom Groben zum Feinen immer weiter unterschieden. Durch den hierarchischen Aufbau wird die Anzahl der notwendigen Klassifikationen stark verringert, so dass es zu maximal $k-1$ Klassifikationen kommt (vgl. [Sch04b], [Sch06]).

- Multi-Klassen-SVM: Hier wird versucht, das Mehrklassenproblem in einem Schritt in einem Modell zu lösen (vgl. z.B. [Lee03], [Wes98], [Liu03]). Da dazu die Theorie erweitert werden muss, und die Lösungsansätze noch nicht ausgereift erscheinen [Nie03], wird dieser Ansatz nicht näher ausgeführt.

3.2 Mehrklassifikatorsysteme

Nicht immer kann für ein Mehrklassenproblem ein einziger optimaler Klassifikator gefunden werden. So kann es sein, dass ein Klassifikator besonders gut eine Klasse von allen anderen trennen kann, aber ein anderer diese Klasse nicht gut separiert, dafür wiederum alle anderen Klassen. Um die Vorteile mehrerer Klassifikatoren zu vereinen, versucht man in vielen Bereichen der Mustererkennung die Ergebnisse mehrerer Klassifikatoren zu

kombinieren (sogenannte „late-fusion“). Auf welche Art und Weise die Ergebnisse optimal fusioniert werden, um ein besseres Gesamtergebnis zu erzielen, ist teilweise noch Gegenstand aktueller Forschung. Hier sollen aufbauend auf den Ergebnissen der in Abschnitt 3.1 vorgestellten Verfahren mögliche Ansätze vorgestellt werden, wie durch die Fusion mehrerer Klassifikatoren die Erkennungsrate eines Gesamtsystems verbessert werden kann.
Es gibt mehrere Gründe, die es rechtfertigen, mehrere Klassifikatoren zu fusionieren und einem einzelnen Klassifikator vorzuziehen:

- **Statistische Gründe:** aus statistischer Sichtweise ist es stabiler, aus einer Menge unterschiedlicher Klassifikatoren (auch schlechten Klassifikatoren) die Ausgabe aus dem Durchschnitt aller zu wählen, als aus diesen einen Klassifikator auszuwählen, der der beste sein könnte.

- **Rechnerische Gründe:** manche Algorithmen verwenden Trainingsverfahren (z. B. Gradientenabstieg), die abhängig von der Initialisierung der Anfangsparameter nicht das globale Minimum, sondern nur eine Näherungslösung finden. Dies kann zu unterschiedlichen lokalen Minima führen. Durch Kombination der Klassifikatoren kann unter Umständen ein gemeinsames Extremum gefunden werden.

- **Repräsentative Gründe:** es ist möglich, dass für ein Problem kein optimale Klassifikator existiert, der alle Klassen optimal trennen kann. Sind zum Beispiel nur lineare Klassifikatoren vorhanden, kann damit kein nichtlineares Problem gelöst werden. Durch die spätere Kombination mehrerer linearer Klassifikatoren können jedoch unterschiedliche Entscheidungsgrenzen erreicht werden. So kann das nichtlineare Problem dennoch gelöst werden, ohne auf einen komplexen Klassifikator zurückgreifen zu müssen.

Eine ausführliche Diskussion verschiedener Fusionstechniken wurde von Kuncheva [Kun01] durchgeführt. Im Folgenden sollen nun die Verfahren kurz vorgestellt werden, die in dieser Arbeit experimentell auf ihre Leistungsfähigkeit untersucht wurden [Rei04a], [Rei04b].

Naive-Bayes Fusion: Diese Art der Fusion wurde von Xu [Xu92] vorgestellt und geht von der Unabhängigkeit der einzelnen Klassifikatoren aus[2]. Für jeden Klassifikator $\mathcal{K}_j$ wird eine Konfusionsmatrix $\boldsymbol{CM}^j$ erzeugt, indem die Trainingsdaten (oder falls vorhanden Validierungsdaten) mit dem Klassifikator $\mathcal{K}_j$ erkannt werden.

$$\boldsymbol{CM}^j = \begin{pmatrix} cm^j_{1,1} & cm^j_{1,2} & \cdots & cm^j_{1,S} \\ cm^j_{2,1} & cm^j_{2,2} & \cdots & cm^j_{2,S} \\ \vdots & \vdots & \ddots & \vdots \\ cm^j_{K,1} & cm^j_{K,2} & \cdots & cm^j_{K,S} \end{pmatrix} \tag{3.38}$$

Die Zeile gibt die wahre Klassenbezeichnung an, während die Spalte die Klasse angibt, auf die der Klassifikator entschieden hat. Der Eintrag $cm^j_{k,s}$ gibt also die Anzahl der Elemente

[2]Diese Annahme ist für reelle Probleme nicht haltbar.

des Datensatzes an, deren wahre Klassenbezeichnung k ist, und die der Klassifikator $\mathcal{K}_j$ der Klasse s zugeordnet hat. Aus dieser Matrix kann nun eine sogenannte Label-Matrix $\boldsymbol{LM}^j$ erzeugt werden, deren Einträge $lm^j_{k,s}$ eine Schätzung der Wahrscheinlichkeit ist, dass die wahre Klassenbezeichnung k ist, wenn der Klassifikator $\mathcal{K}_j$ die Bezeichnung s angibt:

$$lm^j_{k,s} = \hat{P}(k|\mathcal{K}_j(\boldsymbol{x}) = s) = \frac{cm^j_{k,s}}{\sum_k^K cm^j_{k,s}} \tag{3.39}$$

Aufgrund der Unabhängigkeitsannahme der Klassifikatoren kann aus der Label-Matrix ein neuer Wahrscheinlichkeitsvektor μ^i für alle Klassen berechnet werden, indem die Einträge $lm^i_{k,s}$ für alle Klassifikatoren aufmultipliziert werden:

$$\mu_i(\boldsymbol{x}) = \prod_j \hat{P}(i|\mathcal{K}_j(\boldsymbol{x}) = s_j) = \prod_j lm^j_{k,s_j} \quad \forall i \tag{3.40}$$

Aus dem Vektor $\boldsymbol{\mu}$ kann dann durch Maximumsuche die erkannte Klasse bestimmt werden.

Minimum, Maximum, Durchschnitt: Im vorherigen Abschnitt wurde die harte Klassenentscheidung eines Klassifikators verwendet, um durch Kombination zu einer robusteren Klassifikation zu finden. Die im Folgenden vorgestellten Verfahren verwenden nicht die harte Klassenentscheidung sondern eine Bewertung eines jeden Klassifikators über alle Klassen, d.h. jeder Klassifikator gibt mit Hilfe einer Pseudo-Wahrscheinlichkeit an, wie wahrscheinlich jede Klasse ist. Diese „Scores“ werden in einem sogenannten Decision Profile ($\boldsymbol{DP}$) zusammengefasst:

$$\boldsymbol{DP}(\boldsymbol{x}) = \begin{pmatrix} d_{1,1} & d_{1,2} & \cdots & d_{1,S} \\ d_{2,1} & d_{2,2} & \cdots & d_{2,S} \\ \vdots & \vdots & \ddots & \vdots \\ d_{K,1} & d_{K,2} & \cdots & d_{K,S} \end{pmatrix} \tag{3.41}$$

Hierbei bezeichnet der Wert $d_{i,j}$ die Pseudo-Wahrscheinlichkeit des Klassifikators $\mathcal{K}_i$ für die Klasse j. Die Zeilen geben die Scores *eines* Klassifikators für alle Klassen an, während die Spalten die Scores aller Klassifikatoren für eine Klasse angeben.
Basierend auf dem Decision-Profile kann nun eine Kombinationsfunktion $\mathcal{F}$ gewählt werden, um daraus den Entscheidungsvektor $\boldsymbol{\mu}$ zu berechnen. Es gilt dabei:

$$\boldsymbol{\mu} = \mathcal{F}(\boldsymbol{DP}) \tag{3.42}$$

In dieser Arbeit werden folgende Kombinationsfunktionen für die Einträge μ_k verwendet:

- $\mathcal{F}$ = Minimum:

  $$\mu_k(\boldsymbol{x}) = \min_i \{d_{i,k}(\boldsymbol{x})\} \tag{3.43}$$

  Gemäß Gleichung (3.43) wird zunächst spaltenweise das Minimum gesucht, um dann die Zeile mit dem höchsten Eintrag als erkannte Klasse zu wählen.

- $\mathcal{F}$ = Maximum:

$$\mu_k(\boldsymbol{x}) = \max_i \{d_{i,k}(\boldsymbol{x})\} \tag{3.44}$$

  Hier wird als erstes das Maximum in jeder Spalte gesucht, um dann die Zeile mit dem größten Wert als erkannte Klasse auszuwählen.

- $\mathcal{F}$ = Durchschnitt:

$$\mu_k(\boldsymbol{x}) = \frac{\sum_i^K d_{i,k}(\boldsymbol{x})}{K} \tag{3.45}$$

  Wie in Gleichung (3.45) angedeutet, wird der Mittelwert über die Pseudo-Wahrscheinlichkeiten aller Klassifikatoren bezüglich einer Klasse gebildet, bevor das Maximum gesucht wird.

Aus dem vollständigen Vektor $\boldsymbol{\mu}$ kann wieder durch Maximumsuche die erkannte Klasse κ_e gewonnen werden.

$$\kappa_e = \arg\max_\kappa \boldsymbol{\mu} = \arg\max_\kappa [\mu_1, \mu_2, \ldots, \mu_K] \tag{3.46}$$

3.3 Segmentierung mit Sliding-Windows

In diesem Abschnitt werden Verfahren zur automatischen Segmentierung von Videos vorgstellt, wie sie in der vorliegenden Arbeit zum Einsatz kommen. Zunächst wird ein Verfahren erläutert, das ausgehend vom Bayesian Information Criterion (BIC) ein eigenes Maß auf Klassifizierungsebene definiert, und dann in einem Durchgang die Segmentierung und Klassifikation der Daten durchführt. Das zweite Verfahren versucht zuerst, potentielle Segmentgrenzen zu finden und optimiert diese in einem zweiten Schritt [Rei05b].

3.3.1 Integrierter Ansatz

Beim sogenannten integrierten Ansatz zur Segmentierung von Gruppenaktionen wird die Detektion von Segmentgrenzen und die Klassifikation der Segmente in einem kombinierten Schritt durchgeführt. Das Verfahren ähnelt in seiner Vorgehensweise dem des Bayesian Information Criterion, das zuerst von Schwarz vorgestellt wurde [Sch78] und erfolgreich bei der Segmentierung von Audio-Strömen [Zho00] und Sprechern [Tri99] eingesetzt wird aber auch zur Bestimmung von Grenzen bei Gesten [Wal04b], [Zob03]. Der in dieser Arbeit verwendete abgewandelte Algorithmus kann, wie in Abbildung 3.8 gezeigt, als Pseudocode dargestellt werden. Eine graphische Übersicht bietet Abbildung 3.9. Zwei zusammenhängende Zeitfenster mit variabler Breite werden über die Merkmalszeitverläufe geschoben. Die innere Grenze wird dabei in kleinen Schritten von der linken zur rechten Seite geschoben, und in jedem Teilfenster wird ein Merkmalsvektor berechnet (vgl. Abschnitt 2.3.2), der von einem Klassifikator ausgewertet wird. Wenn ein unterschiedliches Klassifikationsergebnis der zwei Teilfenster festgestellt wird, wird angenommen, dass sich an der Position der inneren Grenze eine Segmentgrenze befindet. Wird im ganzen aktuellen Fenster keine

Segmentgrenze gefunden (d.h. die innere Grenze befindet sich fast auf der rechten Fenstergrenze), so wird die rechte Fenstergrenze um ein Intervall nach rechts verschoben, also das Suchfenster vergrößert.

```
(1) Initialisiere das Intervall [a; c]:
        a = 1; b = a + L; c = a + 3L;
(2) Falls K(a, b) ≠ K(b, c) dann
        speichere b als Segmentgrenze
        a = b; b = a + L; c = a + 3L
    sonst
        b = b + 1
(3) Falls (c − b) < L dann
        c = c + 1; b = a + L;
        gehezu (2)
    sonst
        gehezu (2)
```

Abbildung 3.8: *Algorithmus zur Segmentierung von Gruppenaktionen mittels integriertem Ansatz. a bezeichnet die linke Grenze, b die innere Grenze, c ist die rechte Grenze des Zeitfensters, L bezeichnet die minimale Länge einer Gruppenaktion, $K(a, b)$ ist das Klassifikationsergebnis für das Intervall $[a, b]$.*

Der Algorithmus in Abbildung 3.8 wird solange ausgeführt, bis das Ende der Videodatei erreicht ist.
Da die detektierten Grenzen unter Umständen recht ungenau sind, kann zusätzlich in der Umgebung einer jeden potentiellen Grenze eine „Feinsuche“ durchgeführt werden (vgl. Abbildung 3.10). Dazu wird um die zuerst gefundene Grenze b symmetrisch ein Suchfenster mit der Mindestbreite L gelegt. Die Grenzen des Fensters werden mit s und t bezeichnet. Ist das Klassifikationsergebnis des Fensters $[s, t]$ gleich dem des Fensters $[a, b]$ ($K(s, t) = K(a, b)$), so wird das Fenster $[s, t]$ eine Einheit nach rechts verschoben. Gilt allerdings $K(s, t) \neq K(a, b)$, so wird das Fenster $[s, t]$ eine Einheit nach links verschoben. Das Verschieben des Fensters $[s, t]$ erfolgt solange, bis eine Änderung der Klassifikation eintritt, im ersten Fall, bis $K(s, t) \neq K(a, b)$, im zweiten Fall bis $K(s, t) = K(a, b)$. Die Segmentgrenze wird dann bei t liegend angenommen.
Dieser Ansatz erfordert eine hohe Anzahl von Klassifizierungen und beansprucht daher eine hohe Rechenleistung. Dies soll im nachfolgend beschriebenen zweistufigen Ansatz vermieden werden.

3.3.2 Zweistufiger Ansatz

Im Gegensatz zum integrierten Ansatz aus Abschnitt 3.3.1 wird beim zweistufigen Ansatz die Segmentierung von der Klassifikation getrennt und beide Schritte nacheinander ausgeführt. Dadurch kann eine spezialisierte Suche nach optimalen Segmentgrenzen

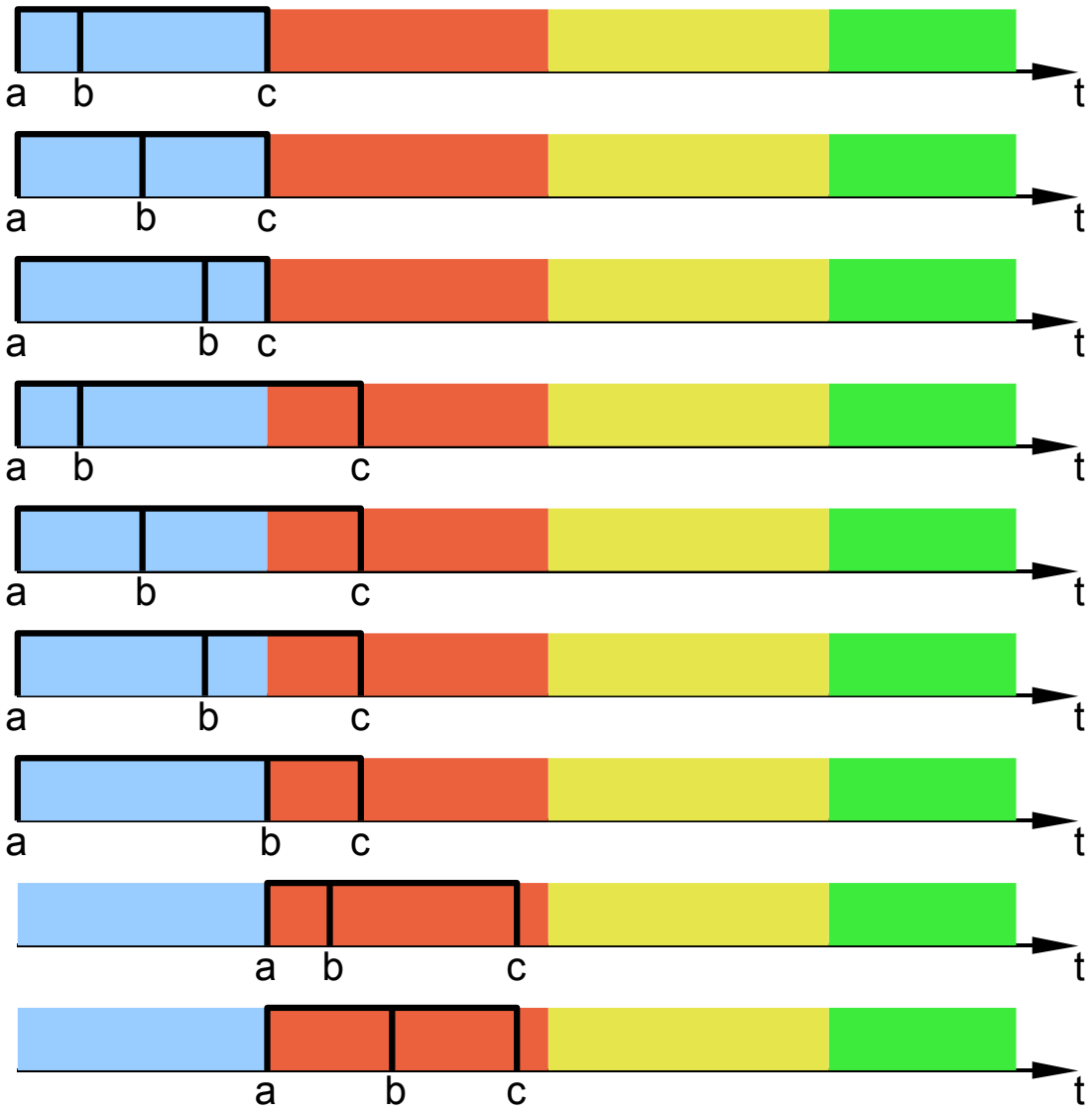


Abbildung 3.9: *Aufeinanderfolgende Schritte mit Sliding-Windows beim integrierten Ansatz zur Segmentierung. Die unterschiedlich eingefärbten Blöcke stellen die zu erkennenden Gruppenaktionen dar.*

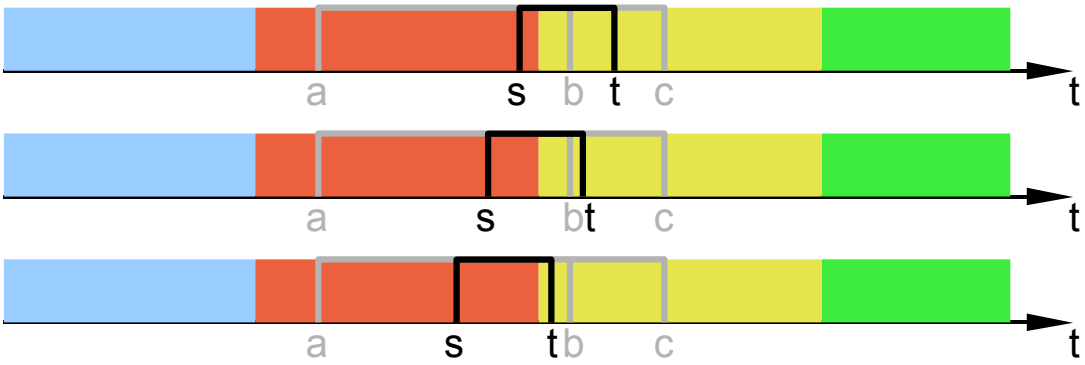


Abbildung 3.10: *Exemplarische Feinsuche zur Verbesserung der Bestimmung der Segmentgrenzen beim integrierten Ansatz*

stattfinden, die einem Optimalitätskriterium genügen. Als Ergebnis sollen sich bessere Segmentierungs- und Erkennungsleistungen einstellen und gleichzeitig die Rechenlast, die bei der aufwändigen integrierten Suche auftritt, vermieden werden.

Suche nach potentiellen Grenzen

Im ersten Schritt werden potentielle Grenzen innerhalb des Merkmalsstroms gesucht. Zu diesem Zweck werden zwei unterschiedliche Verfahren angewendet, der Vergleich von Merkmalsvektoren, und der Vergleich von Klassifikatorergebnissen in zwei zusammenhängenden Zeitfenstern.

Vergleich von Merkmalsvektoren: Die erste Variante der Suche nach den Segmentgrenzen hat als Grundlage einen Vergleich von Merkmalsvektoren in zwei Zeitfenstern. Ähnlich wie im Abschnitt zuvor sind diese Zeitfenster zusammenhängend und werden gleichmäßig über das ganze zu analysierende Video geschoben (vgl. Abbildung 3.11). In den zwei Teilfenstern werden die Merkmalsvektoren $\boldsymbol{f}^{ab}$ bzw. $\boldsymbol{f}^{bc}$ berechnet. Dann werden beide Vektoren auf ihre Ähnlichkeit hin untersucht. Als Distanzmaß findet der City-Block-Abstand Verwendung:

$$d_{f^{ab},f^{bc}} = \sum_i |\boldsymbol{f}_i^{ab} - \boldsymbol{f}_i^{bc}| \tag{3.47}$$

Überschreitet die Distanz $d_{f^{ab},f^{bc}}$ eine festgelegte Schwelle d_{max}, die empirisch ermittelt wurde, so wird die momentane innere Grenze b als potentielle Grenze aufgenommen (siehe Abbildung 3.12). Alle potentiellen Grenzen werden anschließend in Cluster G_i eingeteilt. Dabei beträgt der empirisch festgelegte Mindestabstand zwischen zwei Clustern $T = 5$ Sekunden. Außerdem werden nur Cluster, die eine Mindestanzahl von Grenzen enthalten, weiter berücksichtigt, Cluster mit weniger als $N = 3$ Grenzen werden verworfen.

Vergleich von Klassifikatorergebnissen: Eine zweite Möglichkeit zur Suche von Segmentgrenzen basiert auf den Ergebnissen von Klassifikatoren. Wie bei dem Vergleich der Mustervektoren werden zwei zusammenhängende Zeitfenster über das Video geschoben. In jedem Teilfenster wird der Merkmalsvektor berechnet und (im Unterschied zu oben) durch einen Klassifikator einer Klasse zugeordnet (vgl. auch 3.3.1). Eine mögliche Grenze wird dort vermutet, wenn die Klassifikationsergebnisse im linken und rechten Teilfenster differieren. Die Clusterbildung der gefundenen potentiellen Grenzen wird im gleichen Schritt vollzogen. Solange $K(\boldsymbol{f}^{ab}, \boldsymbol{f}^{bc})$ konstant bleibt, werden neue potentielle Grenzen dem schon bestehenden Cluster G_i hinzugefügt. Ändert sich die zugeordnete Klasse ($K(\boldsymbol{f}_{neu}^{ab}, \boldsymbol{f}_{neu}^{bc}) \neq K(\boldsymbol{f}_{alt}^{ab}, \boldsymbol{f}_{alt}^{bc})$), so wird ein neues Cluster G_{i+1} begonnen. Anschließend werden wieder Cluster, die weniger als eine geforderte Mindestanzahl von Grenzen beinhalten, verworfen.

Bestimmung optimaler Grenzen

Mit Hilfe der dynamischen Programmierung (DP) werden aus allen potentiellen Grenzen in jedem Cluster G_i die Grenzen ausgewählt, die einen Score maximieren. Es wird

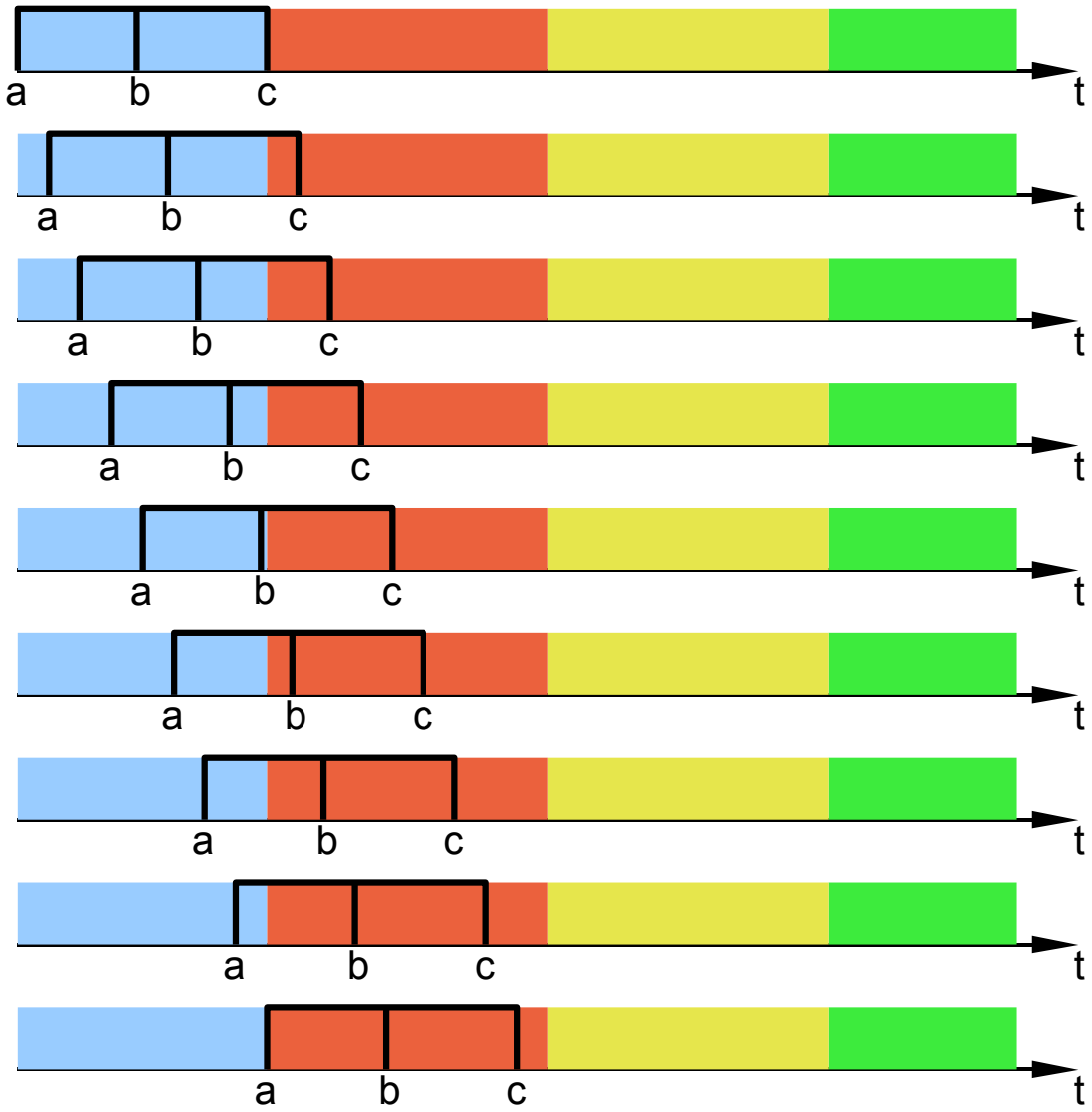


Abbildung 3.11: *Aufeinanderfolgende Schritte mit Sliding-Windows beim zweistufigen Ansatz zur Segmentierung*

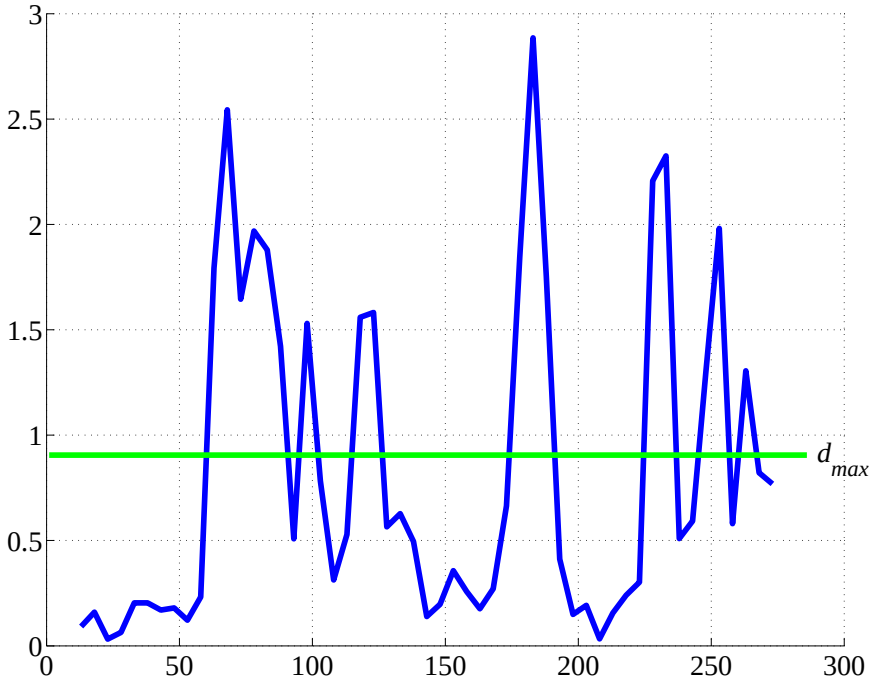


Abbildung 3.12: *Unähnlichkeit der Merkmalsvektoren eines untersuchten Meetings mit eingezeichneter empirisch ermittelter Schwelle d_{max}*

angenommen, dass die Gruppenaktionen voneinander unabhängig sind, und daher jede optimale Segmentgrenze gefunden werden kann, wenn nur der direkte Vorgänger bekannt ist. Die erste und die letzte Grenze sind bekannt (Anfang und Ende eines Videos). Die dazwischenliegenden Grenzen werden so gewählt, dass ein maximaler Gesamtscore erhalten wird. Der Score einer Gruppenaktion wird hier als die Pseudo-Wahrscheinlichkeit bestimmt, mit der ein Klassifikator dem jeweiligen Segment die Klasse zuordnet. Als zusätzliche Beschränkung können nur solche Grenzen gewählt werden, bei deren Wahl die zugehörige Gruppenaktion eine Mindestdauer einhält. Zur Veranschaulichung ist das Auffinden der optimalen Grenzen in Abbildung 3.13 dargestellt. Für jede Grenze $x \in G_i$ wird der Score $s_x(y)$ zu jeder Grenze $y \in G_{i-1},\ i = 2, \ldots, N_G$ berechnet. Dann wird der maximale Score für jedes x ausgewählt:

$$s_{x,max} = \max s_x(y) \tag{3.48}$$

Die Summe aus diesem Score $s_{x,max}$ und dem Gesamtscore bis $i-1$ wird zusammen mit der zugehörigen Vorgängergrenze in einer Score-Matrix $\boldsymbol{SG}^i$ gespeichert:

$$\boldsymbol{SG}^i = \begin{bmatrix} \vdots & \vdots & \vdots \\ x & s_{x,max} + \boldsymbol{SG}^{i-1}_{y,2} & y \\ \vdots & \vdots & \vdots \end{bmatrix}; \tag{3.49}$$

Nachdem alle Cluster abgearbeitet wurden, kann mit Hilfe eines Backtrackings der beste Pfad durch alle Score-Matrizen gefunden werden. Angefangen mit der letzen Score-Matrix

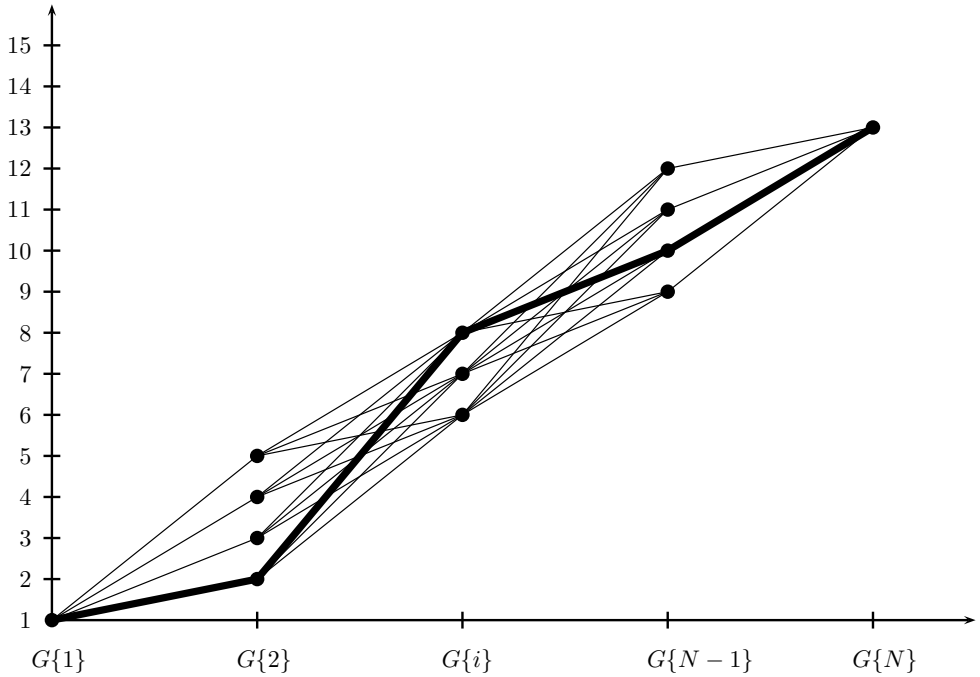


Abbildung 3.13: *Suche nach optimalen Grenzen: Der Pfad mit dem höchsten Score wird durch Backtracking gefunden. Die Abszisse bezeichnet die Cluster mit potentiellen Grenzen, die Ordinate die Nummer der Grenze*

$\boldsymbol{SG}^{N_G}$, die nur eine Grenze enthält, werden die Grenzen ausgewählt, die den größten Gesamtscore bilden. In einem abschließenden Schritt werden dann noch Segmente mit gleicher Gruppenaktion zu einem Segment zusammengefasst.
Aufgrund der Tatsache, dass die Sliding-Windows hier eine feste Fensterbreite besitzen, beansprucht dieser Ansatz weniger Rechenleistung als der integrierte Ansatz.

3.4 Experimente und Ergebnisse

Im Folgenden werden eine Reihe von im Rahmen dieser Arbeit durchgeführten Versuche zur automatischen Modellierung von Gruppenaktionen in Besprechungen vorgestellt. Zum Einsatz kommen die oben dargestellten statischen Klassifikatoren. Zur Evaluierung der Erkennungsleistung werden die in Abschnitt 2.4 vorgestellten Fehlermaße verwendet. Als Daten, die für die folgenden Experimente verwendet wurden, kommen die in Abschnitt 2.1 beschriebenen Meeting-Daten des M4-Corpus zum Einsatz. Als Merkmale werden die semantischen Merkmale aus Abschnitt 2.3.2 verwendet.

3.4.1 Klassifikationsergebnisse

Die Klassifikation der Meeting-Events wird hier zuerst diskutiert, da die automatische Segmentierung die Klassifikation beinhaltet und demnach aufwändiger ist. Die Grenzen der Segmente sind hier also der vorgegebenen „Ground-Truth“ entnommen.

Zunächst wird ein Test durchgeführt, der überprüfen soll, ob die Erkennung von Gruppenaktionen auf Basis von semantischen Ereignissen möglich ist. Dazu werden idealisierte Erkennungsergebnisse von einem Gestenerkenner und Sprecheraktivitätserkenner angenommen, indem die manuell erzeugten Annotationen verarbeitet werden (siehe 2.3.2). Zudem wird die Klassifizierungsaufgabe vereinfacht, indem alle Monologe zusammengefasst werden, sodass nunmehr ein Fünfklassenproblem bearbeitet werden muss (mit den Klassen: *Discussion*, *Monologue*, *Note-taking*, *Presentation*, *White-board*). Die Klassifikationsergebnisse sind in Tabelle 3.1 gezeigt. Verwendet werden fünf verschiedene Klassi-

Klassifikator	**Erkennungsrate** [%]
Naive-Bayes	97,5
GMM	93,9
MLP	98,4
RBF	98,4
SVM	98,4

Tabelle 3.1: *Erkennungsraten statischer Klassifizierung, Merkmale: SEM-M, Lexikon L8 vereinfacht*

fikatoren: Ein Naive-Bayes-Klassifikator, ein GMM mit variabler Anzahl von Mixturen, abhängig von der Anzahl der vorhandenen Trainingsdaten je Klasse, und diagonaler Kovarianzmatrix, ein MLP mit 10 Neuronen in der verborgenen Schicht, ein RBF mit maximal 10 Neuronen und SVM mit Gauß-Kernel.
Die hohen Erkennungraten zeigen, dass eine Klassifikation der Meeting Events basierend auf den Ergebnissen spezialisierter Erkenner (Gesten-, Sprecharaktivitätserkenner) sehr gut möglich ist. Dabei ist in diesem Fall kein signifikanter Unterschied zwischen den verwendeten Klassifikatoren auszumachen. Ein Grund dafür ist darin zu suchen, dass die Datenbank mit 145 Trainingdaten und 122 Testdaten nicht sehr groß ist. Dennoch ist die Größe ausreichend, um das generelle Potential dieser Vorgehensweise aufzuzeigen.

Integration der automatischen Sprechersegmentierung

Einen ersten Ansatz, die manuell erstellten Annotationen durch automatisch generierte Klassifikationsergebnisse zu ersetzen, bieten die mit Hilfe des Mikrophon-Arrays erstellten Sprecheraktivitätssegmentierungen. Da hierbei nicht nach Personen diskriminiert wird, sondern nach dem Ort (im Unterschied zu den Annotationen), muss für die Ereignisse *Presentation* und *Whiteboard* die richtige Person bestimmt werden. Die Sprechereignisse an den beiden Orten werden aus Plausibilitätsgründen dem auftretenden *Mainactor* zugeteilt. Der *Mainactor* ist in der manuellen Annotation enthalten. Auf diese Weise kann also das manuell gelabelte *talking* durch echte Erkennerergebnisse ersetzt werden.
Die Erkennungsergebnisse sind erwartungsgemäß etwas unter denen mit idealisierten Annahmen und sind in Tabelle 3.2 gezeigt. Zusätzlich wurde hier noch die Fusion mehrerer Klassifikatoren nach Gleichung (3.43) miteinbezogen, was allerdings bei der reinen Klassifikation keinen Vorteil bringt. Der Abfall in der Erkennungsleistung bei GMM kann

Klassifikator	**Erkennungsrate** [%]
Naive-Bayes	95,9
GMM	91,8
MLP	96,7
RBF	97,5
SVM	97,5
Fusion	96,7

Tabelle 3.2: *Erkennungsraten statischer Klassifizierung, Merkmale: SEM-M+S, Lexikon L8*

dadurch erklärt werden, dass für ein ausreichendes Training aller Parameter zu wenig Daten zur Verfügung stehen, und das Modell daher zu speziell an die Trainigsdaten angepasst ist. Desweiteren könnte die Initialisierung der Modellparameter nicht optimal erfolgt sein, wodurch beim Training nur ein lokales Extremum gefunden wird.

Klassifikation mit automatisch erzeugten Daten

Werden zur Klassifikation ausschließlich automatisch erzeugte Daten verwendet, so ändert sich das Klassifikationsergebnis nur leicht (siehe Tabelle 3.3). Die Daten werden aus den Ergebnissen einer automatischen Gestenerkennung und einer Sprecheraktivitätserkennung gewonnen (vgl. Abschnitt 2.3.2). Da die Gestenerkennung nicht fehlerfrei arbeitet, sondern eine Klassifikationsrate von etwa 86% besitzt, ist es nur natürlich, dass die Klassifikationsleistung der Erkenner abnimmt, da die Merkmalsvektoren aus diesen fehlerbehafteten Algorithmen hervorgehen. Dennoch ist die Klassifikationsleistung sehr gut, und sinkt im Vergleich zu vollständig manuell generierten Daten (Tabelle 3.1) nur um etwa drei Prozentpunkte absolut ab.

Klassifikator	**Erkennungsrate** [%]
Naive-Bayes	93,4
GMM	91,0
MLP	95,9
RBF	95,1
SVM	95,1

Tabelle 3.3: *Erkennungsraten statischer Klassifizierung, Merkmale: SEM-A, Lexikon L8*

3.4.2 Segmentierungsergebnisse

In diesem Abschnitt werden die Ergebnisse der verschiedenen Segmentierungsverfahren präsentiert. Die Tabellen 3.4 bis 3.13 zeigen die Ergebnisse der drei Fehlermaße (vgl. 2.4)

unter Verwendung der verschiedenen Klassifikatoren und unterschiedlicher Merkmalssets, angefangen von gänzlich manuell erzeugten Merkmalen hin zu vollständig automatisch generierten Merkmalen.
Alle Experimente wurden auf dem Testdatensatz durchgeführt, der aus 29 Meetings mit 139 Segmenten besteht. Die angegebenen Zahlenwerte geben die über alle Testdaten gemittelten Werte an.

Segmentierung unter Verwendung von manuell erzeugten Daten

Um die Leistungsfähigkeit der beschriebenen Segmentierungsverfahren zu evaluieren, werden zunächst möglichst gute Merkmale verwendet, das heißt, die manuell erzeugten Annotationen. Ziel dabei ist, wie auch schon in Kapitel 2.3.2 beschrieben, Ergebnisse von ideal arbeitenden Sprachaktivitäts- und Gestenerkenner zu simulieren. Die Tabellen 3.4 und 3.5 zeigen die Ergebnisse mit den drei verschiedenen Fehlermaßen und unterschiedlichen Merkmalsvektoren. Im ersten Fall ist der Merkmalsvektor wie folgt zusammengesetzt:

$$\boldsymbol{f}_1 = \Bigg[\text{talking}(1),\ \text{talking}(2),\ \text{talking}(3),\ \text{talking}(4),\ \sum_n \text{whiteboard}(n),\ \sum_n \text{presentation}(n),\ \sum_n \text{writing}(n),\ \sum_n \text{standing}(n),\ d^{(2)}_{t_{\text{HR}}}\Bigg] \quad (3.50)$$

Im zweiten Fall kommt ein modifizierter Merkmalsvektor zum Einsatz, der folgendermaßen zusammengesetzt ist:

$$\boldsymbol{f}_2 = \Bigg[d^{(1)}_{t_{HR}},\ \text{talking}(1),\ \text{talking}(2),\ \text{talking}(3),\ \text{talking}(4),\ \sum_n \text{writing}(n),\ \sum_n \text{pointing}(n),\ \sum_n \text{whiteboard}(n),\ \sum_n \text{presentation}(n),\ d^{(2)}_{t_{\text{HR}}}\Bigg] \quad (3.51)$$

Zu erkennen ist, dass die Segmentierung im Großen und Ganzen relativ zuverlässig funktioniert. Die Frame-Fehlerrate liegt unter 20 % mit Ausnahme des MLP im ersten Fall, was aber an der zufälligen Initialisierung der zu trainierenden Parameter liegt. Eine günstigere Ausgangsinitialisierung führt zu weit besseren Ergebnissen, wie im zweiten Fall zu sehen ist, wo nur noch knapp 12 % aller Frames falsch zugeordnet werden. Die beste Frame-Fehlerrate wird mit 8,2 % mit dem Naive-Bayes-Klassifikator erreicht.
Wird das Fehlermaß *M2* betrachtet, so werden auch hier akzeptable Werte erreicht. Die Anzahl der Segmente, die nicht zugeordnet werden können (n' bzw. k'), liegt durchschnittlich pro analysierter Sequenz zwischen 0,0 (alle Segmente haben eine Übereinstimmung) und 0,97 (im Schnitt konnte ein Segment je Meeting nicht zugeordnet werden). Das äußert sich entsprechend auch im Fehlermaß für die Segmentierung (E_{sb}), wenn berücksichtigt wird, dass durchschnittlich fünf Segmente pro Meeting auftreten.
Das dritte Fehlermaß *M3* ist am ehesten mit den bekannten Größen aus der Spracherkennung vergleichbar. Die Raten für die Einfügungen (INS) sind teilweise relativ hoch, es

Klassi-	M1	M2			M3			
fikator	E_f [%]	E_{sb} [%]	E_{sc} [%]	E [%]	INS [%]	DEL [%]	ACC [s]	E_c [%]
BAYES	9,5	20,8	14,3	17,5	8,9	0,8	4,6	11,7
GMM	17,5	33,9	26,3	30,1	13,1	1,5	8,8	21,8
MLP	43,4	69,7	56,9	63,3	17,6	19,7	13,9	42,8
RBF	13,1	28,1	22,5	25,3	11,9	2,1	4,9	14,7
SVM	12,5	30,8	21,8	26,3	11,6	1,5	7,9	16,6

Tabelle 3.4: *Segmentierung mit* integriertem *Ansatz, Merkmale: SEM-M, Merkmalsvektor $\boldsymbol{f}_1$, Lexikon L8*

Klassi-	M1	M2			M3			
fikator	E_f [%]	E_{sb} [%]	E_{sc} [%]	E [%]	INS [%]	DEL [%]	ACC [s]	E_c [%]
BAYES	8,2	18,6	15,2	16,9	8,4	0,0	4,4	12,2
GMM	19,5	36,6	30,6	33,6	16,2	0,8	9,4	24,4
MLP	12,0	24,8	18,9	21,8	10,7	2,2	4,4	15,0
RBF	10,9	22,0	17,3	19,7	9,6	1,3	3,5	12,9
SVM	14,2	34,7	26,8	30,7	16,4	0,7	7,1	20,3

Tabelle 3.5: *Segmentierung mit* integriertem *Ansatz, Merkmale: SEM-M, Merkmalsvektor $\boldsymbol{f}_2$, Lexikon L8*

findet also eine Übersegmentierung statt. Bei Verwendung des Naive-Bayes-Klassifikators kommen aber keinerlei Auslöschungen (DEL) vor. Die Rate der Auslöschungen ist generell sehr niedrig. Zusätzlich angegeben ist die Abweichung der gefundenen Grenzen im Vergleich zur den vorgegebenen Grenzen in Sekunden. Werte um 3 Sekunden sind durchaus in einem akzeptablen Bereich, da auch für einen menschlichen Betrachter der Übergang von einem Meeting-Event zum nächsten nicht immer scharf zu erkennen ist.

Segmentierung unter Verwendung von manuell erzeugten Daten erweitert durch automatische Sprechersegmentierung

Wider Erwarten sind bei Ersetzen der manuellen Sprecheraktivität durch automatisch generierte Ergebnisse die Erkennungsleistungen nicht schlechter als unter „optimalen“ Bedingungen (siehe Tabllen 3.6 bis 3.9). Dies zeigt einerseits die Ausgereiftheit der Algorithmen zur automatischen Sprechersegmentierung, andererseits aber auch die Robustheit der Segmentierungsalgorithmen, die auch mit automatisch generierten Daten, die unter Umständen Fehler aufweisen können, gut zurecht kommen.

Mit diesen Daten können auch die unterschiedlichen Segmentierungsverfahren verglichen werden. Wie zu erwarten war, liefert der integrierte Ansatz durchwegs die besten Ergebnisse. Dies wird allerdings durch einen erhöhten Aufwand in der Rechenzeit erkauft. Die Zeit, die benötigt wird, um mit einem Klassifikator alle Testmeetings zu bearbeiten, be-

Klassi-	**M1**	**M2**			**M3**			
fikator	E_f [%]	E_{sb} [%]	E_{sc} [%]	E [%]	INS [%]	DEL [%]	ACC [s]	E_c [%]
BAYES	10,3	25,2	17,7	21,5	5,7	0,8	5,9	11,6
GMM	18,5	40,6	29,0	34,8	12,0	3,4	10,5	22,9
MLP	14,1	33,1	24,6	28,9	12,2	2,3	8,6	17,5
RBF	11,7	30,9	22,5	26,7	8,3	2,2	6,4	14,2
SVM	14,1	27,6	20,6	24,1	9,5	1,7	6,5	15,8

Tabelle 3.6: *Segmentierung mit* integriertem *Ansatz, Merkmale: SEM-M+S, Merkmalsvektor* $\boldsymbol{f}_1$*, Lexikon L8*

Klassi-	**M1**	**M2**			**M3**			
fikator	E_f [%]	E_{sb} [%]	E_{sc} [%]	E [%]	INS [%]	DEL [%]	ACC [s]	E_c [%]
BAYES	8,2	18,6	15,2	16,9	8,4	0,0	4,4	12,2
GMM	17,3	34,5	26,8	30,6	12,4	2,6	9,1	22,1
MLP	11,5	21,6	16,0	18,8	9,8	1,5	4,2	13,5
RBF	10,9	22,0	17,3	19,7	9,6	1,3	3,5	12,9
SVM	14,2	34,7	26,8	30,7	16,4	0,7	7,1	20,3

Tabelle 3.7: *Segmentierung mit* integriertem *Ansatz, Merkmale: SEM-M+S, Merkmalsvektor* $\boldsymbol{f}_2$*, Lexikon L8*

trägt etwa 27 Minuten[3]. Deutlich schneller wird der zweistufige Ansatz abgearbeitet, der dafür auch leicht schlechtere Ergebnisse produziert. Hierzu wird nur etwa ein Zehntel der Zeit benötigt, im Fall der RBF beispielsweise knapp zwei Minuten. Die Erweiterung mit Dynamischer Programmierung kann die Nachteile in der Erkennungsleistung zum Teil wieder beheben, ohne dabei zu viel Rechenzeit zu beanspruchen. Der zeitliche Mehraufwand beträgt etwa eineinhalb Minuten, so dass die Gesamtbearbeitungszeit für RBF dreieinhalb Minuten ist. Die Ergebnisse erreichen aber nicht ganz die des integrierten Ansatzes. Aufgrund der Tatsache, dass bei Neuronalen Netzen die Ausgänge als Wahrscheinlichkeiten oder Scores interpretiert werden können, wurden nur MLP und RBF für den Test mit Dynamischer Programmierung ausgewählt. Bei den übrigen Klassifikatoren ist zum Teil eine mehr oder weniger aufwändige Nachbearbeitung notwendig. Werden mehrere Klassifikatoren vereint und eine Fusion dieser Erkenner zur Entscheidung herangezogen, so verbessert sich das Ergebnis nur marginal in der Erkennungsrate; bei den Einfügungen und Auslassungen sind sogar leicht schlechtere Ergebnisse zu vermerken. Außerdem verlängert sich die Rechenzeit auf 72 Minuten, da mehrere Klassifikatoren getestet werden müssen. Aufgrund dieses Ergebnisses werden im Folgenden die Mehrklassifikatorsysteme nicht mehr weiter betrachtet.

[3]Getestet wurde auf einem PC (Pentium IV, 2,8 GHz, 2GB RAM) mit dem Betriebssystem Linux (Suse 9.3) unter Matlab 7 (R14).

Klassi-	**M1**	**M2**			**M3**			
fikator	E_f [%]	E_{sb} [%]	E_{sc} [%]	E [%]	INS [%]	DEL [%]	ACC [s]	E_c [%]
BAYES	12,9	41,9	24,8	33,3	9,9	3,3	8,9	19,0
GMM	21,6	42,8	33,6	38,2	10,6	4,1	9,9	23,9
MLP	15,7	29,8	24,9	27,4	9,3	4,4	7,5	19,5
RBF	16,3	42,5	25,9	34,2	10,3	2,4	9,1	19,9
SVM	20,0	33,8	30,1	32,0	10,9	6,3	7,7	23,1
Fusion					9,4	4,9	7,8	17,6

Tabelle 3.8: *Segmentierung mit* zweistufigem *Ansatz, Merkmale: SEM-M+S, Merkmalsvektor* $\boldsymbol{f}_2$*, Lexikon L8*

Klassi-	**M1**	**M2**			**M3**			
fikator	E_f [%]	E_{sb} [%]	E_{sc} [%]	E [%]	INS [%]	DEL [%]	ACC [s]	E_c [%]
MLP	13,9	19,8	14,0	16,9	4,1	4,8	7,1	13,7
RBF	13,0	23,6	17,0	20,3	5,6	3,3	5,1	14,5

Tabelle 3.9: *Segmentierung mit* zweistufigem *Ansatz und aufgesetzter Dynamischer Programmierung, Merkmale SEM-M+S, Merkmalsvektor* $\boldsymbol{f}_2$*, Lexikon L8*

Segmentierung unter ausschließlicher Verwendung der automatischen Sprechersegmentierung

Da die Segmentierung im obigen Abschnitt besser ausfällt, als zu erwarten wäre, wurde ein Experiment unter alleiniger Verwendung der Daten der automatischen Sprechersegmentierung durchgeführt. Die Ergebnisse mit integriertem Ansatz sind in Tabelle 3.10 gezeigt. Im Vergleich zur Segmentierung mit vollständigen Daten (Tabelle 3.7) sind die Ergebnisse erwartungsgemäß schlechter. Die Frame-Fehlerrate (*M1*) steigt um 1 bis 5 Prozentpunkte, die Genauigkeit der detektierten Segmentgrenzen nimmt ab (Fehlermaß *M3*), d. h. die Abweichung steigt von 3,5 bis 9,1 Sekunden auf 8,2 bis 13,1 Sekunden. Ebenso nimmt der Klassifikationsfehler, sowie die Einfügungen und Auslöschungen zu. Daraus ist zu ersehen, dass allein aus der Information, welche Person wann spricht, keine genaue Segmentierung einer Besprechung möglich ist. Insbesondere kann ohne visuelle Information nicht gut zwischen den Gruppenaktionen *presentation* und *white-board* unterschieden werden.

Segmentierung unter Verwendung von automatisch generierten Daten (Sprecheraktivitätserkennung und Gestenerkennung)

Abschließend werden die vorgestellten Verfahren zur Segmentierung einer Besprechung auf möglichst realistischen Merkmalen getestet. Die Merkmale, die hier verwendet werden, wurden ausschließlich automatisch generiert. Der Merkmalsvektor besteht aus semantisch höherwertigen Einheiten, die aus den Ergebnissen der automatischen Sprecher-

Klassi-	**M1**	**M2**			**M3**			
fikator	E_f [%]	E_{sb} [%]	E_{sc} [%]	E [%]	INS [%]	DEL [%]	ACC [s]	E_c [%]
BAYES	13,0	41,7	31,2	36,5	9,4	9,6	8,2	23,3
GMM	13,9	45,8	32,8	39,3	9,9	9,0	9,3	22,7
MLP	16,4	54,5	42,8	48,7	15,6	7,9	13,1	29,3
RBF	13,1	47,8	33,1	40,4	10,7	9,8	9,2	22,0
SVM	15,7	52,0	37,4	44,7	12,6	8,3	12,8	27,3

Tabelle 3.10: *Segmentierung mit* integriertem *Ansatz, Merkmale: automatische Sprechersegmentierung, Lexikon L8*

Klassi-	**M1**	**M2**			**M3**			
fikator	E_f [%]	E_{sb} [%]	E_{sc} [%]	E [%]	INS [%]	DEL [%]	ACC [s]	E_c [%]
BAYES	34,2	38,7	48,1	43,4	14,7	6,2	7,9	39,0
GMM	35,4	55,8	51,5	53,7	22,4	1,7	11,0	38,9
MLP	34,5	37,7	44,8	41,3	11,7	1,7	7,3	34,2
RBF	33,6	27,9	39,0	33,5	6,9	3,0	5,7	31,6
SVM	35,4	51,2	52,3	51,7	17,8	0,8	9,1	35,8

Tabelle 3.11: *Segmentierung mit* integriertem *Ansatz, Merkmale: SEM-A, Lexikon L8*

aktivitätserkennung und der automatischen Gestenerkennung gewonnen werden können (vgl. Abschnitt 2.3.2).

Die Ergebnisse, wie sie in den Tabellen 3.11, 3.12 und 3.13 gezeigt sind, fallen im Vergleich zu den manuell erstellten Merkmalen deutlich ab. Die Frame-Error-Rate verdoppelt sich oder verdreifacht sich sogar. Die Einfügeraten und die Auslöschungsraten verändern sich nicht gravierend, allerdings werden die Grenzen mit einer größeren Abweichung gefunden. Insgesamt ist es zwar möglich, mit Hilfe der vorgestellten Verfahren Besprechungen in Gruppenaktionen zu unterteilen, allerdings hängt die Güte der Segmentierung stark von den zugrunde liegenden Basiserkennern ab. Da hier die gewünschte Genauigkeit momentan

Klassi-	**M1**	**M2**			**M3**			
fikator	E_f [%]	E_{sb} [%]	E_{sc} [%]	E [%]	INS [%]	DEL [%]	ACC [s]	E_c [%]
BAYES	32,7	28,9	37,5	33,2	5,8	5,5	8,4	34,3
GMM	38,1	78,8	62,7	70,8	13,5	0,7	10,2	36,8
MLP	35,0	34,6	42,8	38,7	11,2	2,2	7,5	32,6
RBF	34,1	44,6	41,8	43,2	12,6	1,3	8,9	34,9
SVM	35,8	37,7	44,2	40,9	14,3	2,2	9,0	34,2

Tabelle 3.12: *Segmentierung mit* zweistufigem *Ansatz, Merkmale: SEM-A, Lexikon L8*

Klassi-	M1	M2			M3			
fikator	E_f [%]	E_{sb} [%]	E_{sc} [%]	E [%]	INS [%]	DEL [%]	ACC [s]	E_c [%]
BAYES	33,9	46,5	52,5	49,5	16,5	4,7	6,7	36,6
GMM	36,7	87,6	62,7	75,2	22,0	4,0	21,7	43,9
MLP	37,0	63,5	52,5	58,0	18,7	3,2	16,1	39,0
RBF	35,8	66,3	53,2	59,8	17,4	0,8	16,0	39,7

Tabelle 3.13: *Segmentierung mit* integriertem *Ansatz und Dynamischer Programmierung mit Backtracking, Merkmale: SEM-A, Lexikon L8*

nicht erreicht werden kann, werden in den folgenden Kapiteln weitere Verfahren vorgestellt.

3.5 Kapitelzusammenfassung

In diesem Kapitel wurde statische Klassifikationsverfahren, wie z. B. MLP und SVM, verwendet, um vorgegebene Segmente mit Gruppenaktionen zu identifizieren. Für statische Verfahren ist es jedoch notwendig, die Daten in einem separaten Schritt zu segmentieren. Dafür wurden zwei verschiedene Verfahren vorgestellt: Der integrierte Ansatz kombiniert eine Methode mit gleitenden Fenstern nach dem Prinzip des Bayesian Information Criterion mit Klassifikationsergebnissen; beim zweistufigen Ansatz werden zuerst potentielle Segmentgrenzen definiert, die dann mit Hilfe dynamischer Programmierung optimiert werden. Dabei hat sich gezeigt, dass das integrierte Verfahren im Allgemeinen robustere Ergebnisse liefert als das zweistufige Verfahren. Auch die Auswahl der Merkmale ist von entscheidender Bedeutung für die Leistungsfähigkeit dieses Segmentierungsansatzes. Mit manuell erzeugten semantischen Merkmalen können sehr gute Ergebnisse (FER von 8,2 % mit integriertem Ansatz) erzielt werden. Werden automatisch von einer Sprechersegmentierung und einer Gestenerkennung erzeugte Merkmale verwendet, so steigt die Fehlerrate rapide an (FER von über 30 %). Die Ergebnisse dieses Kapitels zeigen, dass statische Klassifikationsverfahren für eine Erkennung von Gruppenaktionen gut geeignet sind, jedoch ist die vorausgehende Erkennung der semantischen Merkmale kritisch für das Gesamterkennungsergebnis.

Kapitel 4

Segmentierung mit neurobiologischen Klassifikationsverfahren

In diesem Kapitel werden Verfahren zur Segmentierung von Besprechungen vorgestellt, die sich – mehr oder weniger – aus dem biologischen Vorbild der Signalverarbeitung des Menschen ableiten lassen: *Rekurrente Neuronale Netze* (RNN) und erweiterte bzw. abgewandelte Variationen[1]. Zunächst werden die RNN allgemein eingeführt, bevor die spezielle Form der *Long-Short-Term Memory* Netze ausgeführt wird. Den Abschluss dieses Kapitels bildet die Beschreibung eines Modells, das aus den *Neuronalen Feldern* abgeleitet wurde.

4.1 Segmentierung mit Rekurrenten Neuronalen Netzen

Die in Kapitel 3.1.3 vorgestellten vorwärtsgerichteten Neuronalen Netze haben immer nur Verbindungen von einer tiefer gelegenen Schicht zu einer höher gelegenen Schicht. Diese Einschränkung kann aufgehoben werden, wenn man Verbindungen von einer höheren Schicht zu einer darunterliegenden Schicht zulässt. Dadurch entstehen Rückkopplungen, die dem Neuronalen Netz eine Fülle zusätzlicher Eigenschaften verleihen. Aufgrund der Rückkopplungen werden diese Art von Netzen auch Rückgekoppelte Neuronale Netze oder, aus dem englischen Sprachraum kommend, Recurrend Neural Nets (RNN) genannt.

4.1.1 Rückgekoppelte Neuronale Netze

Ein RNN erhält durch die Rückkopplungen ähnliche Eigenschaften wie ein dynamisches System im Zustandsraum, bei dem der Zustandsvektor $\boldsymbol{x}(t)$ rekursiv aus dem Zustandsvektor $\boldsymbol{x}(t-1)$ berechnet wird. Auf diese Weise kann ein Ausgangsmuster berechnet werden, das Werte von vorangegangenen Mustern beinhaltet und berücksichtigt. Werden die Gewichtungen im RNN richtig gewählt, ergibt sich ein Konvergenzverhalten, bei dem ein stabiles Ausgangsmuster entsteht. Bei ungünstiger Parameterwahl dagegen kann es zu Stabilitätsproblemen kommen. In Abbildung 4.1 ist ein mögliches RNN gezeigt, bei dem die Ausgänge wieder auf die Eingänge des Netzes zurückgeführt werden. Dies ist nur eine

[1] Rein vorwärts gerichtete Neuronale Netze sind zwar auch von der Neurobiologie inspiriert, da sie die Topologie jedoch sehr vereinfacht darstellen, wurden sie bereits im vorherigen Kapitel vorgestellt.

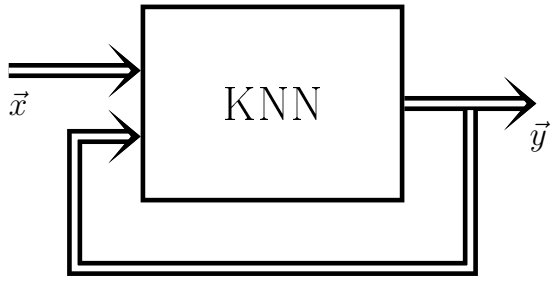


Abbildung 4.1: *Ein rekurrentes Netzwerk ausgehend von einem vorwärtsgerichteten KNN*

mögliche Form der Rückkopplung. Andere Möglichkeiten, eine Schleife in einem Netzwerk zu erzeugen, bestehen in (vgl. [Sch97a]):

- lateralen Interaktionen zwischen den Neuronen einer Schicht,
- Rückkopplungen von einer höheren Schicht zu einer niedrigeren.

Eine sogenannte *Jordan-Struktur* [Jor86] besitzt Rückkopplungen von den Ausgängen zurück zu sogenannten Kontextzellen, die dann im nächsten Zeitschritt die Information an die Neuronen der verborgenen Schicht weitergeben (siehe Abbildung 4.2). Jordan-

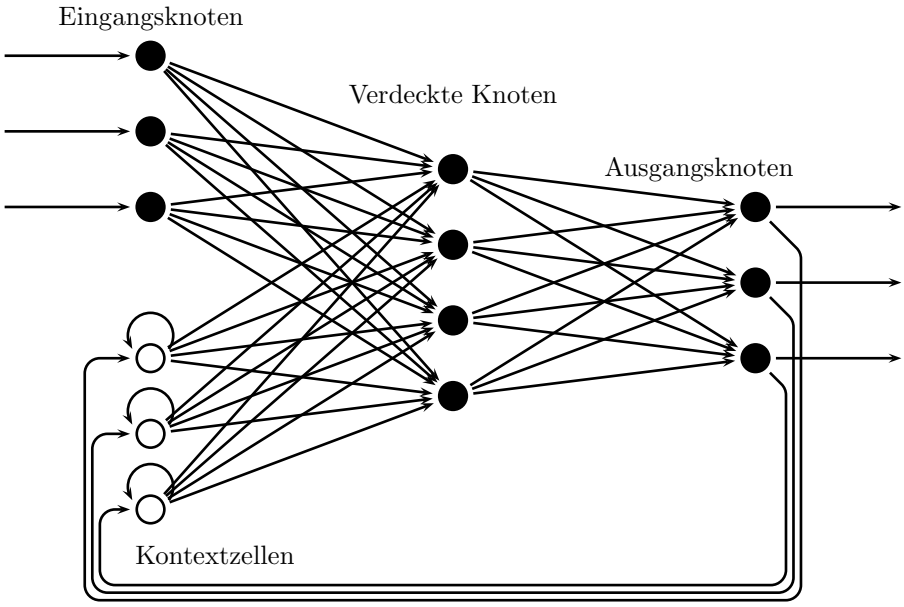


Abbildung 4.2: *Struktur eines Jordan-Netzwerkes. Die Ausgangsschicht wird über Kontextzellen rückgekoppelt. Dadurch können zeitliche Sequenzen generiert werden.*

Netze besitzen normal trainierbare Vorwärtsgewichtungen. Die Kontextzellen sind durch feste Gewichtungen μ eins zu eins mit den Ausgangszellen verbunden. Zusätzlich gibt es feste Gewichtungen λ in den direkten Rückkopplungen der Kontextzellen. Der Vektor der Kontextzellen wird mit $\boldsymbol{s}$ bezeichnet. Damit ergibt sich für den Zeitpunkt k folgender Vektor $\boldsymbol{s}(k)$:

$$\boldsymbol{s}(k) = \lambda \cdot \boldsymbol{s}(k-1) + \mu \cdot \boldsymbol{y}(k-1) \tag{4.1}$$

oder wenn die Rekursion aufgelöst wird:

$$\boldsymbol{s}(k) = \mu \cdot \sum_{n=0}^{k-1} \lambda^n \cdot \boldsymbol{y}(k-n-1) \tag{4.2}$$

Der Vektor der Kontextzellen zum Zeitpunkt k ist die gewichtete Summe der Vektoren aller früheren Zeitpunkte $n = 0, \ldots, k-1$. Aus Gleichung (4.2) ist außerdem zu entnehmen, dass der Einfluss der Vergangenheit von dem Parameter λ abhängt, wobei gilt: $0 \leq \lambda \leq 1$. Je größer dieser „Vergessfaktor“ ist, desto weniger werden alte Muster vergessen. Allerdings nimmt der Einfluss der Vergangenheit exponentiell ab.
Durch Modifikation des Jordan-Netzes kann ein RNN mit ähnlichem Aufbau erzeugt werden: das *Elman-Netzwerk* [Elm90]. Hier werden nicht die Ausgänge zurückgeführt, sondern die verborgene Schicht wird über Kontextzellen zu sich selbst rückgekoppelt. Der grundsätzliche Aufbau eines solchen Elman-Netzwerkes ist in Abbildung 4.3 zu sehen. Die

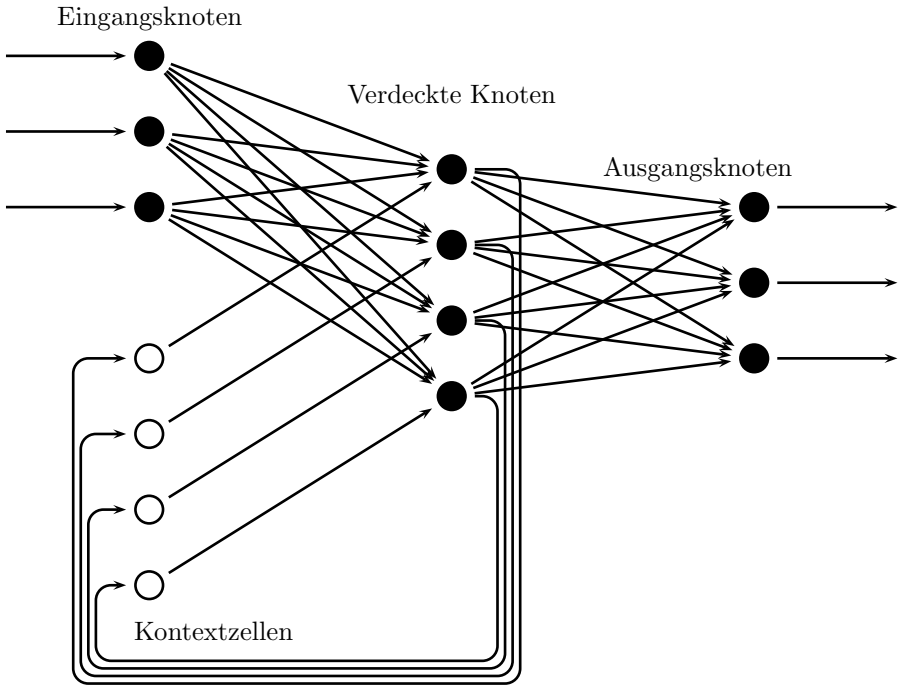


Abbildung 4.3: *Struktur eines Elman-Netzwerkes. Die verborgene Schicht wird über Kontextzellen rückgekoppelt.*

Anzahl der Kontextzellen ist gleich der Anzahl der Neuronen in der verdeckten Schicht, wobei meist die Gewichtungen $\mu = 1$ gesetzt werden. Zudem fehlen die direkten Rückkopplungen der Kontextzellen. Dadurch wird immer nur der Inhalt der verdeckten Schicht zum vorhergehenden Zeitpunkt gespeichert, also nur der letzte Zeitschritt berücksichtigt. Eine weitere Variante stellt die Rückkopplung der Ausgangsknoten zum Ausgang dar (siehe Abbildung 4.4). Auf diese Weise werden die Schichten des Netzwerkes zeitlich hintereinandergefügt. Auf eine verdeckte Schicht, wie beim MLP, zur Erzeugung nicht-linearer Trennebenen kann hier verzichtet werden. Diese Art von RNN wird leicht variiert erfolgreich in der Bestimmung von a-posteriori Wahrscheinlichkeiten in der Spracherkennung eingesetzt [Rob94].

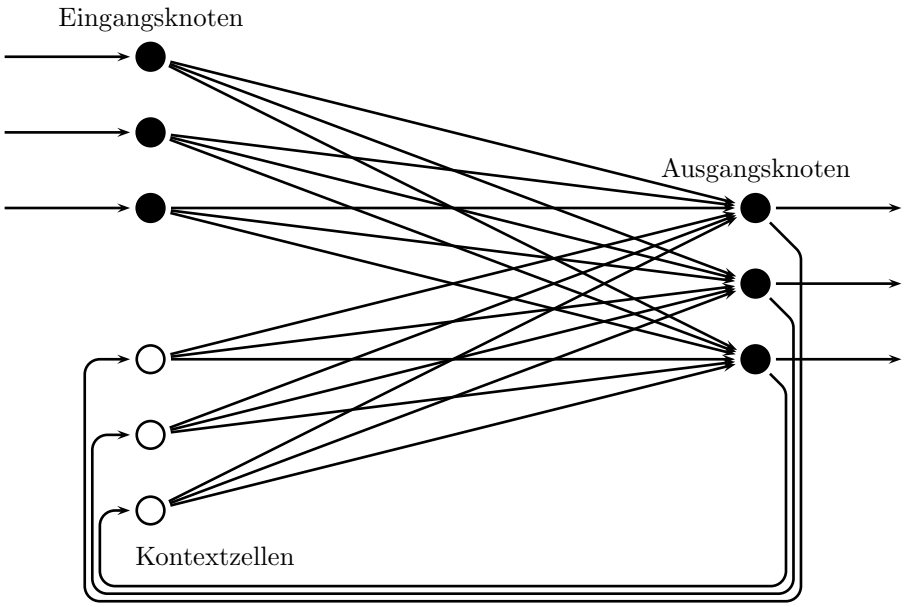


Abbildung 4.4: *RNN mit direkter Rückkopplung der Ausgänge auf die Ausgangsschicht über Kontextzellen*

Für das Training von rückgekoppelten Netzen existieren im Wesentlichen zwei Verfahren [Wil90]:

- BPTT (Backpropagation Through Time),
- RTRL (Real-Time Recurrent Learning).

Der Grundgedankte des BPTT besteht darin, dass sich jedes rekurrente Netzwerk als großes Feedforward-Netz darstellen lässt, wenn man den Ausgang $\boldsymbol{y}(k-1)$ als Eingang in eine höhergelegene Schicht interpretiert, die $\boldsymbol{y}(k)$ generiert. Aus einem einschichtigen rekurrenten Netz mit K Iterationszyklen wird dadurch ein K-schichtiges statisches Netz mit einem Iterationszyklus. Ein solches ausgefaltetes Netz kann dann mit dem Backpropagation-Algorithmus wie ein statisches Netz trainiert werden. Dabei muss beachtet werden, dass die Gewichtungen in allen Schichten identisch sein müssen, und in jeder Schicht neue Eingänge hinzukommen. Für eine genaue mathematische Beschreibung des Verfahrens wird auf die einschlägige Literatur verwiesen (z. B. [Rig01b], [Wil90]).
Das RTRL ist eine Möglichkeit, unendliche Sequenzen zu trainieren, indem die Gewichtungen nach jedem Zeitschritt angepasst werden, und der Fehler nicht rückwärts durch das Netz propagiert werden muss. Diese Methode ist auch für das online-Lernen geeignet. Allerdings beansprucht dieses Lernverfahren große Mengen von Arbeitsspeicher und Rechenkapazität. Daher wird, sofern möglich, auf alternative Trainingsalgorithmen zurückgegriffen. Zur genaueren Beschreibung mit einer ausführlichen mathematischen Behandlung sei wieder auf die weiterführende Literatur verwiesen (z. B. [Rig01b], [Wil90], [Sch93]).
Neben diesen zwei grundlegenden Trainingsverfahren für RNN existiert eine Vielzahl von weiteren abgeleiteten und optimierten Algorithmen, auf die in dieser Arbeit nicht eingegangen werden soll.

4.1.2 Long Short-Term Memory Neuronale Netze

Rekurrente Neuronale Netze eignen sich aufgrund der Rückkopplungen für das Lernen von korrelierten Sequenzen. Wie schon im vorherigen Abschnitt erwähnt, nimmt der Einfluss vergangener Muster exponentiell ab. Dies hat Hochreiter in einer ausführlichen Analyse [Hoc91] untersucht. Durch den exponentiell abklingenden Einfluss werden in etwa nur die letzten zehn angelegten Eingangsmuster berücksichtigt. Dadurch werden Zeitabstände, die größer als zehn sind, vergessen. Manchmal kann von kurzen Beispielen auf lange Sequenzen generalisiert werden, häufig ist das allerdings nicht der Fall.
Aus diesem Grund schlägt Hochreiter vor, einen konstanten Fehlerfluss einzuführen. Dies führt zu dem sogenannten Long Short-Term Memory (LSTM) [Hoc97]. Damit wird es möglich, Zeitabstände über 1000 zu berücksichtigen. Dabei bleibt das „Kurzzeitgedächtnis“ erhalten und die Komplexität steigt nur minimal.

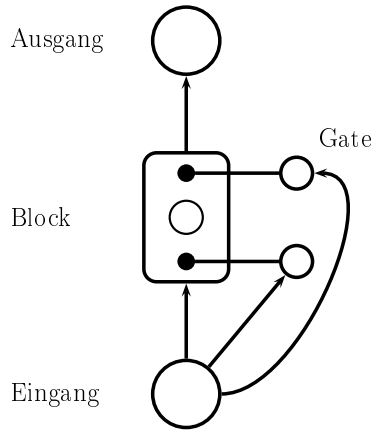


Abbildung 4.5: *Einfachstes LSTM-Netzwerk mit einem Eingang, einem Ausgang und einem LSTM-Block anstelle der verborgenen Schicht.*

In Abbildung 4.5 ist das einfachst mögliche LSTM-Netz gezeigt. Es besteht aus einem Eingang, einem Ausgang und einem Memory-Block anstelle der verborgenen Neuronen. Jeder Block hat zwei sogenannte Gates. Typischerweise gibt es gewichtete Verbindungen von der Eingangsschicht zu den LSTM-Blöcken und den Gates. Zusätzlich gibt es rekurrente Verbindungen zwischen den LSTM-Blöcken und den Gates. Schließlich existieren Verbindungen von den Blöcken zu der Ausgangsschicht.
Die verborgene Schicht eines konventionellen RNN wird durch LSTM-Blöcke ersetzt, die aus einer oder mehreren LSTM-Zellen bestehen. Eine solche Zelle ist in Abbildung 4.6 gezeigt.
Im Zentrum einer LSTM-Zelle befindet sich ein einfaches lineares Neuron mit einem Selbstübergang mit dem Gewicht 1,0. Liegt kein Eingang an, so sorgt diese Verbindung dafür, dass der gegenwärtige Zustand zum nächsten Zeitschritt erhalten bleibt. Diese Architektur wird auch konstantes Fehler-Karussell genannt[2]. Die Gates sind dafür zuständig, zu entscheiden, welche Information im Fehler-Karussell gespeichert wird, beziehungsweise wann diese Information ausgegeben wird.

[2]englisch: Constant Error Carousel (CEC)

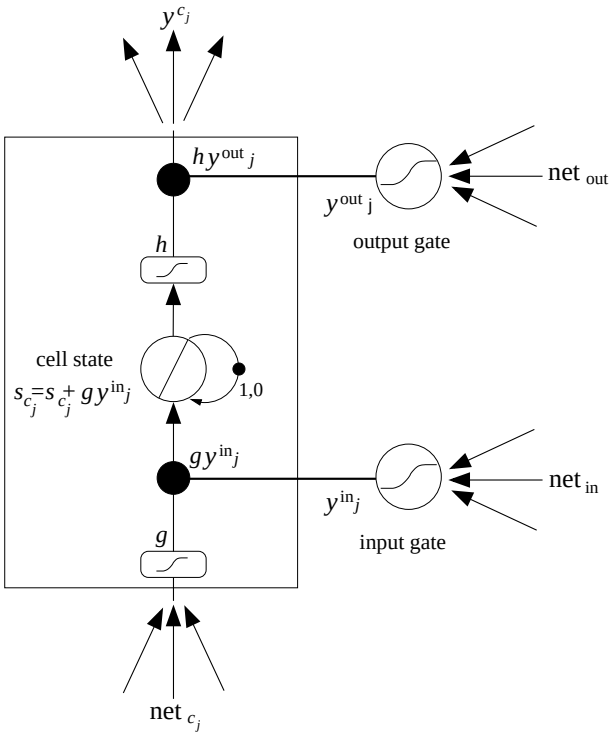


Abbildung 4.6: *Eine LSTM-Zelle. Im Zentrum das „Fehler-Karussell“ mit konstantem Gewicht* 1,0. *Die Input- und Output-Gates steuern den Zugang zur bzw. Ausgang der LSTM-Zelle.*

Der Input net_{c_j} für eine LSTM-Zelle wird über eine nichtlineare Funktion $g(x)$, die typischerweise eine logarithmische Funktion darstellt, in das innere der Zelle geleitet. Dort wird er mit dem Ausgang y^{in_j} des Input-Gates multipliziert. Die Aktivierungsfunktion des Input-Gates wird so gewählt, dass sie einen Wertebereich von 0 bis 1 besitzt. Dadurch kann dieser Wert entscheiden, ob der Input net_{c_j} zum Zentrum der Zelle durchgelassen wird, oder nicht. Ist der Wert von y^{in_j} nahe 0, kann nichts zu dem Speicherelement durchdringen. Auf ähnliche Weise steuert der Ausgang y^{out_j} des Output-Gates, wann die Zelle einen Ausgabewert $y^{c_j} > 0$ emittiert. Da der interne Zellzustand durch eine lineare Einheit mit unbegrenztem Ausgangswertebereich repräsentiert wird, muss der Ausgang mit einer geeigneten Funktion $h(x)$ auf einen sinnvollen Wertebereich, typischerweise $[-1; 1]$, beschränkt werden. Der Ausgang $y^{c_j}(t)$ zum Zeitpunkt t der Zelle c_j kann damit folgendermaßen beschrieben werden:

$$y^{c_j}(t) = y^{\text{out}_j}(t)\; h(s_{c_j}(t)) \tag{4.3}$$

Der interne Zustand s_{c_j} ist dabei gegeben als:

$$\begin{aligned} s_{c_j}(0) &= 0 \\ s_{c_j}(t) &= s_{c_j}(t-1) + g(\text{net}_{c_j}(t))\; y^{\text{in}_j}(t) \qquad \forall t > 0 \end{aligned} \tag{4.4}$$

Die verschiedenen Elemente der LSTM-Zelle haben unterschiedliche Verantwortungsbereiche: Das Input-Gate entscheidet, was gespeichert wird, die Zelle speichert die Information

und das Output-Gate bestimmt, wann Information weitergegeben wird. Dadurch können hervorstechende Ereignisse beliebig lange gespeichert werden und später abgerufen werden. Durch mehrere LSTM-Blöcke ist es einem Netzwerk dann möglich, Ereignisse mit unterschiedlichen Zeitskalen zu behandeln.

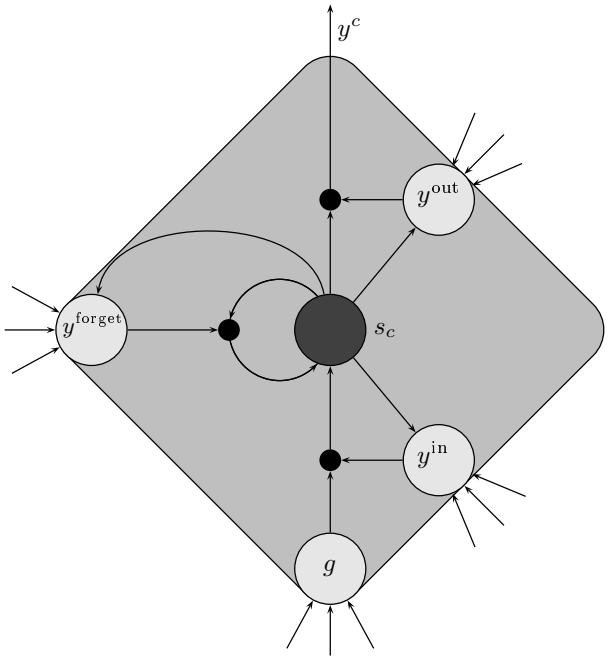


Abbildung 4.7: *Erweiterung der LSTM-Zelle durch ein weiteres Gate (Forget-Gate), welches das gezielte Vergessen von Information ermöglicht.*

Aufgrund des konstanten Fehlerflusses im Inneren der Zelle kann eine einmal gelernte Information nie mehr vergessen werden. Manchmal ist es aber gewünscht, eine Zelle zurückzusetzen. Aus diesem Grund kann die LSTM-Zelle mit einem zusätzlichen Gate erweitert werden, dem sogenannten Forget-Gate [Ger01a]. Dieses kann direkt auf das Fehler-Karussell zugreifen und ein gezieltes Vergessen ermöglichen (siehe Abbildung 4.7). Somit berechnet sich der interne Zustand zu

$$s_{c_j}(t) = y^{\text{forget}_j} \cdot s_{c_j}(t-1) + g(\text{net}_{c_j}(t)) \cdot y^{\text{in}_j}(t) \qquad \forall t > 0 \tag{4.5}$$

indem Gleichung (4.4) um den Wert y^{forget_j} erweitert wird.
Die Netzwerktopologie eines LSTM-Netzwerkes besteht in der Regel aus einer Eingangsschicht, einer verborgenen Schicht und einer Ausgangsschicht. Die verborgene Schicht ist vollständig mit sich selbst verknüpft und enthält LSTM-Zellen und Gates. Zusätzlich können hier auch „konventionelle“ Neuronen enthalten sein. Alle Einheiten (mit Ausnahme der Gates) haben direkte Verbindungen zu allen Einheiten in der übergeordneten Schicht. Mehrere LSTM-Zellen können zu einem LSTM-Block zusammengefasst werden, indem sie sich das Input- und Output-Gate teilen (siehe Abbildung 4.8). Dadurch können verteilte Informationen leichter gespeichert werden, da sie auf mehrere Zellen aufgeteilt werden können. Zudem ergibt sich unter Umständen ein leichter Vorteil in der Komplexität, da sich die Anzahl der Knoten reduziert.

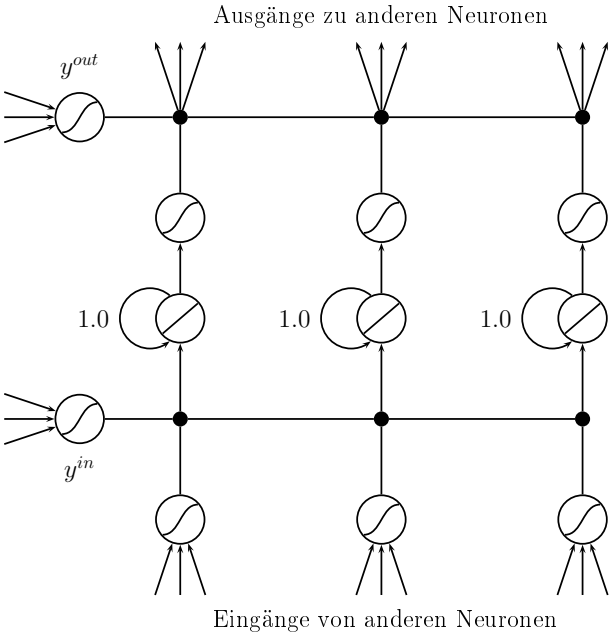


Abbildung 4.8: *LSTM-Block, in dem mehrere LSTM-Zellen sich ein Input- und Output-Gate teilen*

Das Training eines LSTM-Netzes erfolgt über eine Kombination aus RTRL und BPTT. Eine genaue Beschreibung des Verfahrens ist bei Hochreiter und Schmidhuber [Hoc97] zu finden und wird hier nicht weiter erläutert.

Die Segmentierung einer Merkmalssequenz erfolgt implizit durch die zeitlich nacheinander erfolgende Zuordnung eines Frames zu einer Gruppenaktion. Dabei können unter Umständen jedem Zeitschritt unterschiedliche Klassen zugeordnet werden, so dass eine starke Übersegmentierung auftreten kann. Um diese zu vermeiden, können die Ergebnisse mit einem Filter nachbearbeitet werden, oder aber, es werden andere Segmentierungsverfahren vorgeschaltet, so dass LSTM-Netze nur für die Klassifizierung eingesetzt werden.

LSTM-Netzwerke wurden und werden erfolgreich in einer Vielzahl von Anwendungen eingesetzt, unter anderem zum Lernen von kontextfreien und kontextsensitiven Sprachen [Ger01b], zur automatischen Komposition von Musik [Eck02], in der automatischen Spracherkennung [Gra05] und zum Reinforcement-Lernen [Bak01].

4.2 Segmentierung mit einem Neuronalen Feldern

Dieser Abschnitt gibt zunächst einen Überblick über mögliche oder bereits vorhandene Einsatzgebiete von Neuronalen Feldern, beschreibt danach die mathematischen Grundlagen, zeigt die Nähe zu Rekurrenten Neuronalen Netzen auf und entwickelt deren spezielle, in dieser Arbeit verwendete, Form. Anschließend werden durchgeführte Experimente und deren Ergebnisse präsentiert.

4.2.1 Überblick

Neuronale Felder sind ein mathematisches Modell, mit dessen Hilfe versucht wird, die Vorgänge im menschlichen Gehirn zu erklären und nachzubilden. Darauf aufbauend werden Systeme entwickelt, die die besonderen Fähigkeiten der Neuronalen Felder nutzen, um zu zeigen, dass spezielle Probleme mit dieser Art von Modellen gelöst werden können. Krekelberg [Kre92] zeigt, dass unter bestimmten Umständen ein Neuronales Feld einen „Winner-takes-it-all“ Algorithmus emuliert. Chang et al. [Cha93] weisen nach, dass es mit Hilfe von kontinuierlichen neuronalen Systemen möglich ist, Muster zu speichern, bzw. verrauschte Muster zu verbessern. Ebenso sei dieses Modell invariant gegenüber bestimmten Transformationen. Dass es möglich ist, die Parameter von Neuronalen Feldern mit Hilfe von Optimierungsmethoden zu bestimmen, zeigen Igel et al. [Ige01]. Eine erste konkretere Anwendung wird von Giese [Gie00] beschrieben, wobei biologische Bewegungsmuster wie zum Beispiel „gehen“ und „rennen“ mit Hilfen von Neuronalen Feldern unterschieden werden können. Auch für die frühe Integration von auditivem und visuellem Kanal können Neuronale Felder eingesetzt werden [Sch04a]. Andere Einsatzgebiete sind beispielsweise im Fahrzeug im Bereich der Fahrerassistenz [Ahr99], [Ede01], oder um Fahrverhalten zu modellieren [Lee01]. Desweiteren werden Neuronale Felder im Bereich der Robotik eingesetzt, um das Verhalten von Robotern zu beschreiben [Ber99], [Men00], oder um Roboter zu steuern [Ste97], [Ste00b], bzw. diese anthropomorpher zu gestalten [Ste00a].
Trotz der sehr allgemeinen Beschreibung haben sich Neuronale Felder im Bereich der Mustererkennung bisher nicht etablieren können. Gerade für die Klassifikation statischer Muster scheint ein so hochgradig dynamisches Modell nicht geeignet und nur mit tiefgreifenden Veränderungen des Modells möglich. Daher wird in dieser Arbeit erstmalig versucht, durch geeignete Anpassung der Modellbeschreibung, ein Modell aus der Theorie der Neuronalen Felder zu entwickeln, um damit eine Segmentierung von Besprechungen zu erreichen (vgl. auch [Rei05c]).
Studien über Neuronale Felder und deren mathematische Beschreibung existieren schon geraume Zeit (vgl. [Wil73], [Ama77], [Erm98]). Jedoch wurden sie, wohl aufgrund der erst jetzt zur Verfügung stehenden Rechenleistung, bisher kaum für reale Anwendungen verwendet. Erst in neuerer Zeit haben Forschergruppen dieses Modell wieder entdeckt. Einen Überblick über aktuell vorhandene Arbeiten geben unter anderem Jirsa et al. [Jir01], Vogels et al. [Vog05] und Coombes [Coo05].

4.2.2 Modellbeschreibung

Allen Arbeiten über Neuronale Felder gemeinsam ist die Beschreibung der Aktivität der Neuronen mit Hilfe einer integralen Differentialgleichung:

$$\tau \frac{\partial u(x,t)}{\partial t} = -u(x,t) + \int w(|x-y|) f[u(y)] \mathrm{d}y + h + s(x,t) \tag{4.6}$$

Hierbei beschreibt $u(x,t)$ das durchschnittliche Membranpotential zum Zeitpunkt t einer Neuronenverteilung an der Stelle x. Die mittlere Aktivität wird durch die Transferfunktion

$f[u(t)]$ beschrieben, die eine beliebige Gestalt annehmen kann. Der Term $s(x,t)$ bezeichnet den Eingang für die Neuronenverteilung. Die Stärke der Konnektivität zwischen den Neuronen im Abstand $|y|$ zum Neuron x beschreibt $w(|x-y|)$, eine Funktion des relativen Abstands zweier Neuronen. τ bezeichnet die Zeitkonstante der Dynamik und $h \geq 0$ den Ruhelevel, auf den der Ausgang $u(x,t)$ des Neuronalen Feldes zurückfällt, wenn keine Werte am Eingang anliegen. Als Konnetivitätsfunktion wird häufig eine Kombination aus Exponentialfunktionen verwendet, die als sogenannte „Mexican Hat“ - Funktion bekannt ist und in Abbildung 4.9 gezeigt ist.

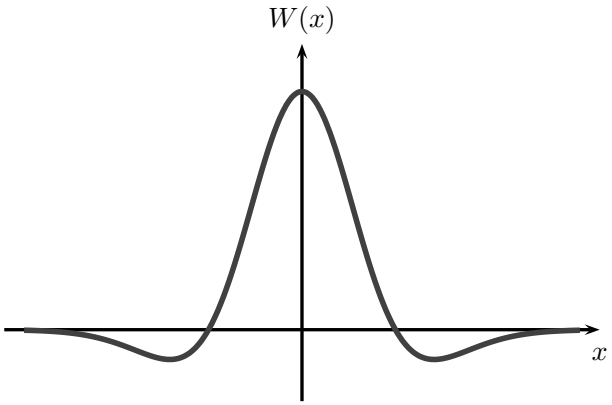


Abbildung 4.9: *„Mexican Hat“ - Funktion*

Dieses Modell kann, wie oben kurz erwähnt, für diverse Aufgaben eingesetzt werden, bei denen die Dynamik der Neuronalen Felder vorteilhaft ist. In dieser Arbeit wird allerdings nur die Idee, bzw. der räumliche Aufbau übernommen, um aus den Neuronalen Feldern ein Modell zu entwickeln, das in der Lage ist, Meetings in semantische Abschnitte zu untergliedern.

Das Grundkonzept basiert auf der Idee, die Eingangsmerkmale einer Besprechung nicht, wie sonst üblich, sequentiell dem Modell anzubieten, sondern alle Merkmale als Ganzes in paralleler Weise. Die Merkmale an den Eingängen werden in die richtige zeitliche Reihenfolge gebracht, in N Frames zusammengefasst und dem Modell dargeboten. Sodann liegt im Idealfall am Ausgang eine Aktivitätsverteilung an, die die Segmentierung der Besprechung repräsentiert. Abbildung 4.10 verdeutlicht diese Idee. Die berechneten Ausgänge geben an, wann und in welcher Reihenfolge welche Gruppenaktionen auftreten.

Werden die Eingangsdaten in der vorgestellten Weise gehandhabt, so müssen die zeitlichen Verknüpfungen nicht mehr berücksichtigt werden, und die Zeitabhängigkeit kann eliminiert werden. Aus dem Eingangsterm $s(x,t)$ wird also $s(x)$. Desweiteren sind für eine Klassifikation nur stabile Zustände des Modells von Interesse, d. h. die Aktivität der Neuronen ändert sich nicht mehr. Dies ist in Gleichung (4.6) der Fall, wenn $\frac{\partial u(x,t)}{\partial t} = 0$. Berücksichtigt man diese zwei Einschränkungen, so kann die Gleichung für die Aktivität eines Neuronalen Feldes wie folgt geschrieben werden:

$$u(x) = \int w(|x-y|) f[u(y)] dy + h + s(x) \tag{4.7}$$

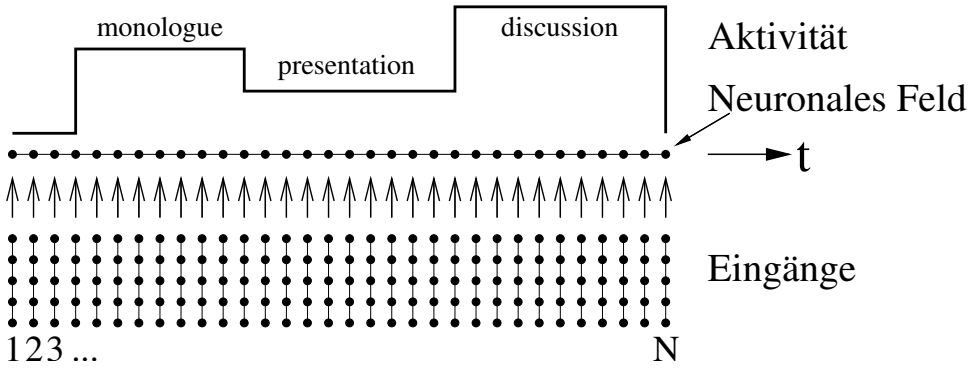


Abbildung 4.10: *Schematische Zeichnung des Modells nach der Idee der Neuronalen Felder*

Unter Beachtung der Tatsache, dass Rechner nur diskrete Werte verarbeiten können und keinen kontinuierlichen Zahlenraum, muss Gleichung (4.7) diskretisiert werden. Wird also das Integralzeichen durch ein Summenzeichen ersetzt, erhält man:

$$\boldsymbol{u}(k) = \sum_{\xi=1}^{N} w(|k-\xi|) f[u_\xi(k-1)] + h + s(k) \tag{4.8}$$

Der Index ξ läuft dabei von 1 bis N, der Anzahl aller Neuronen des Feldes.
Untersucht man den Aufbau von Gleichung (4.8), so fällt auf, dass eine gewisse Ähnlichkeit mit Rekurrenten Neuronalen Netzen (RNN) vorhanden ist. So hat zum Beispiel die erste Schicht eines Elman-Netzwerkes [Elm90] folgende Gleichung:

$$a^1(k) = f\left[\sum_\xi W_\xi^L a_\xi^1(k-1) + \sum_\xi W_\xi^I s_\xi(k)\right] + h \tag{4.9}$$

Hierbei bezeichnet W^L die Gewichtsmatrix, $s(k)$ den Eingang und h ist der Bias. Setzt man nun für die Aktivierungsfunktion f die Identitätsfunktion $f(x) = x$ und fasst man die Summe der Eingänge zusammen zu $s(k) = \sum_\xi W_\xi^I s_{\xi(k)}$, so ergibt sich die folgende Gleichung, in der einzelne Terme ihre Entsprechung in der Gleichung des Neuronalen Feldes haben:

$$a^1(k) = \sum_\xi W_\xi^I a^1(k-1) + h + s(k) \tag{4.10}$$

Unter diesen Annahmen ist es möglich ein äquivalentes Rekurrentes Neuronales Netz zu definieren, ein Paralleles Rekurrentes Neuronales Netz (PRNN), das in etwa die gleiche Architektur besitzt, wie das Modell, das aus dem Neuronalen Feld abgeleitet wurde. Der Vorteil dabei ist, dass alle für RNN bekannte Trainingsalgorithmen ohne weitere Probleme angewendet werden können. Abbildung 4.11 zeigt die Architektur des PRNN, wie es in dieser Arbeit zur Segmentierung und Klassifikation herangezogen wird. Generell sind Verbindungen von allen Neuronen zu allen anderen Neuronen möglich. Jedoch sind in dem vorgestellten Modell nur Neuronen in einer Umgebung miteinander verknüpft. Als Koppelbreite wird die Anzahl der Neuronen definiert, zu denen von einem Neuron aus

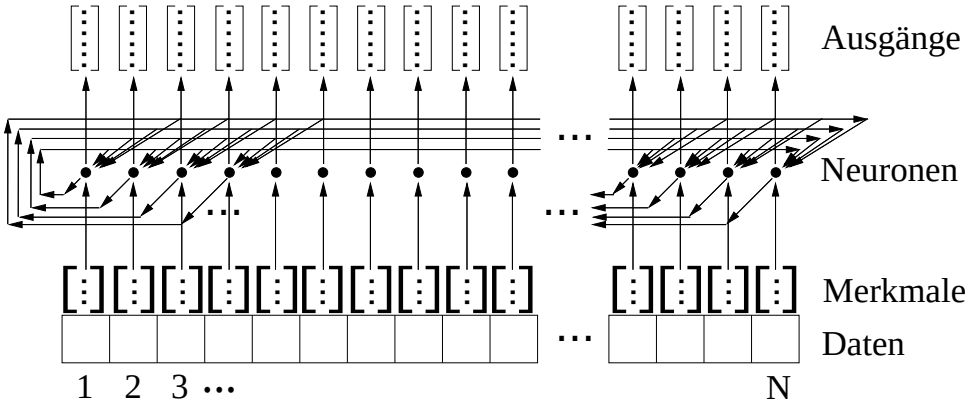


Abbildung 4.11: *Architektur eines Parallelen Rekurrenten Neuronalen Netzes, abgeleitet von Neuronalen Feldern (nicht alle Verknüpfungen sind eingezeichnet)*

rekursive Verknüpfungen bestehen. Diese Rückkopplungen simulieren die hemmende Einwirkung benachbarter Neuronen, wie es bei den Neuronalen Feldern der Fall ist.
Die Besonderheit dieses Ansatzes besteht darin, dass alle Merkmale eines gesamten Meetings parallel dem Modell übergeben werden. Im Gegensatz zu herkömmlichen RNN mit einer zeitlichen Anordnung, erlaubt der räumliche Aufbau von PRNN eine gegenseitige Einflussnahme der Merkmale sowohl vom Anfang des Meetings auf Aktionen am Ende einer Besprechung, als auch umgekehrt.

4.3 Experimente und Ergebnisse

In den folgenden Abschnitten werden die Ergebnisse der Versuche mit den rückgekoppelten Modellen präsentiert. Zunächst werden RNN, wie sie üblicherweise verwendet werden, mit LSTM verglichen. Anschließend werden PRNN, die aus den Neuronalen Feldern abgeleitet wurden, untersucht.

4.3.1 Ergebnisse mit Rekurrenten Netzen

Für die Segmentierung von Besprechungen werden hier die in Kapitel 4.1.2 vorgestellten Modelle eingesetzt. In diesen Experimenten werden die semantischen Daten aus Abschnitt 2.3.2 verwendet. Zum Vergleich mit den LSTM-Netzen werden RNN verwendet, wie sie in Kapitel 4.1.1 beschrieben werden. Die Anzahl der Eingangsknoten beträgt stets sieben, während die Anzahl der Ausgangsknoten acht beträgt, da acht verschiedene Klassen unterschieden werden sollen. Die Kodierung der Klassen erfolgt binär, das heißt, jeder Ausgangsknoten kann einen Wert zwischen null und eins annehmen. Derjenige Knoten mit der höchsten Aktivität bestimmt die Zugehörigkeit zur Klasse. Untersucht werden bei den LSTM-Netzen unterschiedlich viele LSTM-Blöcke in der verborgenen Schicht mit variierenden Größen. Zur Bewertung der Klassifikationsleistung findet die Frame-Fehlerrate (siehe Abschnitt 2.4.1) Anwendung. In Tabelle 4.1 sind die Frame-Fehlerraten von LSTM-Netzen mit einem LSTM-Block und einer LSTM-Zelle bis zu acht LSTM-Blöcken mit

LSTM-Blöcke	**LSTM-Zellen**	**FER** [%]	**LSTM-Blöcke**	**LSTM-Zellen**	**FER** [%]
1	1	45,2	5	1	15,1
1	2	48,1	5	2	51,6
1	3	70,4	5	3	11,7
1	4	70,4	5	4	18,8
2	1	29,7	6	1	12,8
2	2	52,0	6	2	68,5
2	3	49,5	6	3	23,3
2	4	28,9	6	4	14,7
3	1	27,8	7	1	19,6
3	2	23,5	7	2	13,1
3	3	86,0	7	3	88,8
3	4	17,6	7	4	32,2
4	1	12,3	8	1	33,2
4	2	18,3	8	2	30,1
4	3	42,3	8	3	99,1
4	**4**	**11,5**	8	4	55,8

Tabelle 4.1: *Frame-Fehlerraten (FER) unterschiedlicher LSTM-Netzwerke mit variierter Anzahl und Größe der LSTM-Blöcke, Merkmale SEM-A, Lexikon L8*

vier Zellen zu sehen. Auffällig ist, dass kein offensichtlicher Zusammenhang zwischen der Fehlerrate und der Anzahl der Blöcke bzw. Zellen zu bestehen scheint. Auch ist die Bandbreite der Ergebnisse sehr groß. Das beste Ergebnis mit einer FER von 11,5 % wird bei vier LSTM-Blöcken mit jeweils vier LSTM-Zellen erhalten. Das schlechteste Resultat, bei dem fast kein Frame richtig erkannt wird, liegt bei 99,1 % wenn acht Blöcke mit jeweils drei Zellen eingesetzt werden.

Werden die Bereichsgrenzen der Gruppenaktionen vorgegeben und wiederum die Frames durch das LSTM-Netz bestimmt, so kann mit Hilfe eines Mehrheitsentscheids bestimmt werden, welche Aktion innerhalb eines Segments erkannt wurde. Dadurch erhält man eine Erkennungsrate, die für die gleichen LSTM-Kombinationen wie oben in Tabelle 4.2 gezeigt ist. Auch hier lässt sich wiederum keine Abhängigkeit der Erkennungsrate zur Struktur des LSTM-Netzes ablesen.

Das beste Ergebnis wird auch hier mit vier LSTM-Blöcken mit einer Größe von jeweils vier Zellen erreicht.

Vergleicht man nun die Erkennungsleistung von LSTM-Netzen mit der herkömmlicher rekurrenter Netze, so kann man feststellen, dass die LSTM-Netze eine deutlich bessere Leistung erbringen als andere Topologien von RNN. In Tabelle 4.3 ist das jeweils beste Ergebnis angegeben, das mit der jeweiligen Art von RNN erzielt werden konnte. Das

LSTM-Blöcke	LSTM-Zellen	Erkennungsrate [%]	LSTM-Blöcke	LSTM-Zellen	Erkennungsrate [%]
1	1	59,6	5	1	90,8
1	2	56,0	5	2	47,7
1	3	34,9	5	3	95,4
1	4	34,9	5	4	85,3
2	1	75,2	6	1	94,5
2	2	50,5	6	2	39,5
2	3	53,2	6	3	82,6
2	4	75,2	6	4	89,0
3	1	76,2	7	1	87,1
3	2	80,7	7	2	92,7
3	3	14,7	7	3	11,0
3	4	89,9	7	4	78,0
4	1	95,4	8	1	69,7
4	2	90,8	8	2	77,0
4	3	63,3	8	3	2,8
4	**4**	**96,3**	8	4	44,0

Tabelle 4.2: *Erkennungsraten verschieden großer LSTM-Netzwerke bei vorgegebenen Grenzen der Gruppenaktionen und Bestimmung der Klasse aufgrund eines Mehrheitsentscheids innerhalb eines Segments, Merkmale: SEM-A, Lexikon L8*

schlechte Abschneiden anderer Varianten von RNN deutet auf suboptimale Parameter der Modelle hin. Die Klassifikationsrate von 96,3 % übertrifft sogar die der statischen Klassifikatoren (vgl. Tabelle 3.3), bei denen die höchste Erkennungsrate bei 95,9 % lag.

4.3.2 Ergebnisse mit Parallelen Rekurrenten Neuronalen Netzen

Das in Kapitel 4.2 vorgestellte Modell des PRNN, das vom Aufbau einem neuronalen Feld nachempfunden ist, wird nun auf die Aufgabe, Besprechungen in semantische Einheiten, die Gruppenaktionen, zu unterteilen, angewendet (siehe dazu auch [Rei05a], [Rei05c]).
Als Daten werden hier die Merkmale, die aus der automatischen Sprechersegmentierung gewonnen werden, und Global Motion Features (vgl. 2.3.1) verwendet. Alternativ kommen auch die semantischen Merkmale, die Ergebnisse der automatischen Sprechersegmentierung und der Gestenerkennung, zum Einsatz.
Das Parallele Rekurrente Neuronale Netz hat folgenden Aufbau: Für jeden Frame i eines Videos mit N Frames gibt es ein Neuron im RNN. Der Eingangsvektor besteht bei Verwendung der Low-Level Merkmale aus sechs bzw. zwölf Einträgen (aus der Sprechersegmentierung bzw. Global Motion Merkmalen), je nachdem, ob ein unimodaler oder mul-

Netzart	**Erkennungsrate** [%]
Jordan-Netz	35,2
Elman-Netz	51,2
Direkte Rückkopplung	90,7
LSTM	96,3

Tabelle 4.3: *Vergleich der Erkennungsraten von verschiedenen rekurrenten neuronalen Netzen bei vorgegebenen Segmentgrenzen, Merkmale: SEM-A, Lexikon L8*

timodaler Ansatz verfolgt wird. Der Ausgang des RNN ist binär kodiert. Daher besteht die Ausgangsschicht aus $8 \cdot N$ Neuronen, da acht verschiedene Klassen unterschieden werden. Für jeden der N Frames wird die erkannte Gruppenaktion durch den Neuronenausgang mit dem höchsten Wert bestimmt.
Die Merkmale werden mit einer Rate von fünf Hertz erzeugt. Dadurch ergibt sich unter der Annahme, dass eine Besprechung eine Länge von circa fünf Minuten besitzt, eine Gesamtzahl von $5\text{min} \cdot 60\frac{s}{\text{min}} \cdot 5\frac{1}{s} = 1500$ Frames. Ein RNN mit einer solch großen Anzahl von Eingängen ist rechentechnisch nicht mehr handhabbar. Daher müssen die Daten reduziert werden. Dies geschieht durch einfache Mittelung der Merkmale über die Anzahl der zusammenzufassenden Frames. Als praktischer Nebeneffekt stellt sich dabei ein, dass auf diese Weise zudem eine gewisse Mindestlänge einer Gruppenaktion bestimmt werden kann. Werden zum Beispiel Merkmale von zwei Sekunden zusammengefasst, kann auch keine erkannte Gruppenaktion kürzer als zwei Sekunden sein.
Es wurden verschiedene Experimente mit einer unterschiedlichen Anzahl von Eingängen

sec.	**FER (ST)** [%]	**FER (GM)** [%]	**FER (ST+GM)** [%]
2	42,6	59,0	53,1
4	42,4	62,2	46,8
6	40,3	62,7	44,3
8	42,1	54,3	49,8
10	37,1	52,4	42,8
12	40,3	54,3	44,8
14	42,0	52,4	41,8
16	39,5	49,6	46,9
18	39,0	55,5	45,7
20	33,3	51,1	43,8

Tabelle 4.4: *Frame-Fehlerraten (FER) von Parallelen Rekurrenten Neuronalen Netzen mit verschiedener Mindestlänge der Gruppenaktion bei einer festen Koppelbreite von 3 Neuronen, die erste Spalte zeigt die Mindestlänge einer Gruppenaktion in Sekunden, die weiteren Spalten die Frame-Fehlerraten. Merkmale:Sprechersegmentierung (ST),Global Motion Merkmalen (GM) und beide Modalitäten (ST+GM), Lexikon L8*

und Zeitschritten (d. h. Anzahl der Sekunden, über die die Merkmale zusammengefasst werden) durchgeführt. Zunächst wurden nur Versuche mit einer Modalität (Sprechersegmentierung) durchgeführt. In Tabelle 4.4 sind die Ergebnisse bei fester Koppelbreite und verschiedener Mindestlänge einer Gruppenaktion gezeigt.

Das beste Ergebnis wird hier erzielt, wenn die Mindestdauer einer Gruppenaktion 20 Sekunden beträgt. Die Frame-Fehlerrate beträgt dabei nur 33,3 %. Eine längere Mindestdauer zu wählen ist nicht sinnvoll, da die Gruppenaktionen teilweise nicht länger sind. Eine Abhängigkeit zwischen der Frame-Fehlerrate und der Mindestdauer lässt sich nicht ablesen.

Werden die gleichen Versuche nur mit Global Motion Merkmalen durchgeführt, so erhält man bei der gleichen Mindestlänge deutlich schlechtere Ergebnisse. Eine Abhängigkeit von Mindestdauer und Frame-Fehlerrate ist auch hier nicht auszumachen. Insbesondere liegt hier das Minimum bei 16 Sekunden Mindestlänge mit einer Frame-Fehlerrate von 49,6 %. Die niedrigste FER mit 46,3 % konnte bei einer Mindestlänge von 20 Sekunden und keiner Koppelung unter benachbarten Neuronen erreicht werden. Untersucht man

N	**FER (ST)** [%]	**FER (GM)** [%]	**FER (ST+GM)** [%]
1	39,3	46,3	45,5
2	34,2	55,5	45,0
3	33,3	51,1	43,8
4	40,5	53,2	41,8
5	36,0	51,2	39,5
6	40,7	48,2	44,5
7	42,6	54,4	45,8
8	36,8	48,0	42,5
9	39,7	50,1	40,6
10	39,3	50,2	46,2
11	41,4	58,6	43,8
12	41,6	51,2	44,8
13	40,1	67,0	44,3
14	36,0	50,0	41,9
15	35,8	49,7	43,7
17	38,2	46,4	49,1
19	38,2	46,4	49,1
20	38,2	46,4	49,1

Tabelle 4.5: *Ergebnisse der Versuche mit unterschiedlicher Koppelbreite (d. h. Anzahl der Neuronen, die sich gegenseitig beeinflussen können). Die minimale Länge einer Gruppenaktion beträgt* 20 *Sekunden. Die erste Spalte bezeichnet die Koppelbreite (N), die anderen Spalten jeweils die Frame-Fehlerraten. Merkmale: Sprechersegmentierung (ST), Global Motion Merkmale (GM) und beide Modalitäten (ST+GM), Lexikon L8*

die Frame-Fehlerrate in Abhängigkeit der Koppelbreite, so lässt sich auch hier keine eindeutige Tendenz feststellen (siehe Tabelle 4.5). Die besten FER werden mit Merkmalen der Sprechersegmentierung mit einer Koppelbreite von $N = 3$ Neuronen erzielt, mit Global Motion Merkmalen mit $N = 1$. Die Kombination beider Modalitäten erreicht eine minimale FER bei einer Koppelbreite von $N = 5$ Neuronen.
In allen Fällen hat sich gezeigt, dass die Verwendung der Daten der Sprechersegmentierung alleine bessere Ergebnisse erzielt, als die frühe Fusion der Merkmale aus Sprechersegmentierung und Global Motion Merkmalen. Dies legt den Schluss nahe, dass entweder die Merkmale, insbesondere die Global Motion Merkmale, für diese Aufgabe nicht geeignet sind, oder aber, dass das vorgestellte Modell mit den zusätzlichen Informationen nicht zurechtkommt.
Um die Erkennungsleistung besser mit anderen Verfahren vergleichen zu können, werden auch Experimente zur Erkennung von Gruppenaktionen in vordefinierten Zeitfenstern durchgeführt (also nur Klassifikation, keine Segmentierung). Die resultierende Gruppenaktion wird in den Zeitfenstern durch einen Mehrheitsentscheid bestimmt. Tabelle 4.6 zeigt die Erkennungsraten, die das Modell auf unterschiedlichen Merkmalssets erreicht.

Merkmale	**Mindestlänge**	**Koppelbreite**	**Erkennungsrate** [%]
Low-Level Merkmale (LOW)	6	7	60,4
Sprechersegmentierung	20	3	67,3
Global Motion Merkmale	4	9	57,8
Sprechersegmentierung + Global Motion Merkmale	4	8	66,0
semantische Merkmale (SEM-A)	1	16	65,1

Tabelle 4.6: *Erkennungsraten von PRNN mit verschiedenen Merkmalssets, Lexikon L8*

Da die Ergebnisse im Schnitt deutlich schlechter ausfallen, als mit anderen Methoden, die auf den gleichen Daten angewendet werden (siehe Tabelle 4.7 und z. B. Tabellen 3.3, 3.10 oder 4.3), wurden weitere Untersuchungen mit diesem Modell nicht durchgeführt. Es scheint, als sei die vorgestellte Architektur eher für anders geartete Probleme, wie in der Literatur gezeigt, anwendbar.

Klassifikator	**Erkennungsrate** [%]
MLP	95,9
LSTM	96,3
RNN-parallel	65,1

Tabelle 4.7: *Vergleich unterschiedlicher NN-Architekturen, Merkmale: SEM-A, Lexikon L8*

4.4 Kapitelzusammenfassung

In diesem Kapitel wurden Rekurrente Neuronale Netze mit unterschiedlichen Architekturen vorgestellt und verglichen. Long Short-Term Memory Neuronale Netze haben sich hierbei als sehr effizient in der Klassifizierung von Gruppenaktionen bei gegebenen Grenzen erwiesen. Es konnte eine Erkennungsrate von 96,3 % erreicht werden. Der in dieser Arbeit neu präsentierte Ansatz des Parallelen Rekurrenten Neuronalen Netzes (PRNN), der die zeitlich aufeinanderfolgenden Merkmale in eine räumliche Verteilung transformiert, konnte die gewünschte Leistung nicht erbringen. Die höchste erzielbare Erkennungsrate lag bei 65,1 %. Deshalb wurden keine weiteren Experimente mit PRNN durchgeführt.

Kapitel 5

Segmentierung mit dynamischen Klassifikationsverfahren

In den vorangegangenen Kapiteln wurden Mustererkennungsverfahren vorgestellt, die für ein Segment mit vorgegebenen Segmentgrenzen die Klassenzugehörigkeit bestimmen konnten. In Kapitel 3.3 wurde für diese Verfahren eine Methodik eingeführt, um die Segmentgrenzen automatisch zu bestimmen. In diesem Kapitel werden zwei Klassifikatoren erläutert, die die Segmentierung und die Klassifikation in einem Dekodierungsschritt vereinen: Hidden-Markov-Modelle und Hidden Conditional Random Fields.

5.1 Segmentierung mit Hidden-Markov-Modellen

Hidden-Markov-Modelle (HMM) sind mächtige und flexible Klassifikatoren, die bei einer Vielzahl von Klassifikationsaufgaben zum Einsatz kommen. Sie werden zum Beispiel erfolgreich in der Spracherkennung [ST95], in der Handschrifterkennung [Rig96a] oder in der Gestenerkennung [Wu99] eingesetzt. Im Umfeld der Besprechungsanalyse werden HMM zum Beispiel zur Erkennung von individuellen Aktionen der Teilnehmer verwendet [Wal04b].
Die besonderen Vorteile von HMM gegenüber statischen Klassifikatoren bestehen darin, dass zum einen durch geeignete Modellierung Streuungen innerhalb ähnlicher Merkmalsvektoren zugelassen werden, zum anderen aufgrund der Struktur eine temporale Variabilität ermöglicht wird, um unterschiedlich lange Musterfolgen zu generieren.
Ein HMM besteht aus zwei überlagerten Zufallsprozessen. Der erste Zufallsprozess besteht aus einer Markov-Kette, die mit Zuständen und Übergangswahrscheinlichkeiten beschrieben werden kann. Ein zweiter Zufallsprozess erzeugt zu jedem Zeitpunkt eine Beobachtung gemäß einer Wahrscheinlichkeitsverteilung. Die jeweils zu einem Zustand gehörigen Mustervektoren werden bei kontinuierlichen HMM durch multivariate Normalverteilungen beschrieben, bei diskreten HMM durch diskrete Wahrscheinlichkeiten.
Da ein HMM nur Merkmalsequenzen einer Klasse Ω_κ nachbilden kann, werden für die Unterscheidung von K Klassen K HMM benötigt. Es wird bei der Erkennung einer Mustersequenz dasjenige Modell λ_{κ_e} gesucht, das die größte Produktionswahrscheinlichkeit für eine zu generierende Beobachtung $\boldsymbol{O} = (\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T)$ erzeugt. Es wird also das Modell gewählt, das die größte a-posteriori Wahrscheinlichkeit bei gegebener Beobachtungsse-

quenz $\boldsymbol{O}$ besitzt:

$$\kappa_e = \arg\max_{\kappa} P(\lambda_\kappa|\boldsymbol{O}) \tag{5.1}$$

Da die Wahrscheinlichkeit $P(\lambda_\kappa|\boldsymbol{O})$ nicht direkt bestimmt werden kann, wird mit Hilfe des Satzes von Bayes die Wahrscheinlichkeit wie folgt umgeformt:

$$P(\lambda_\kappa|\boldsymbol{O}) = \frac{P(\boldsymbol{O}|\lambda_\kappa)\,P(\lambda_\kappa)}{P(\boldsymbol{O})} \tag{5.2}$$

Da $P(\boldsymbol{O})$ nicht von dem Modell λ_κ abhängt, kann dieser Term bei der Maximumfindung vernachlässigt werden. Die Bestimmung der a-priori Wahrscheinlichkeit $P(\lambda_\kappa)$ ist je nach Einsatzgebiet nur schwer oder gar nicht möglich. Häufig geht man jedoch bei einer Anzahl von K Klassen von einer Gleichverteilung aller Modelle aus, so dass $P(\lambda_\kappa) = 1/K$ gilt. Dann trägt dieser Term ebenfalls nicht mehr zur Maximumbildung bei, sodass sich Gleichung (5.1) auch schreiben lässt als

$$\kappa_e = \arg\max_{\kappa} P(\boldsymbol{O}|\lambda_\kappa) \tag{5.3}$$

Zur Klassifikation muss also nur die deutlich leichter zu bestimmende Wahrscheinlichkeit $P(\boldsymbol{O}|\lambda_\kappa)$ anstelle von $P(\lambda_\kappa|\boldsymbol{O})$ ermittelt werden.

5.1.1 Modellbeschreibung

Ein HMM λ lässt sich mit den folgenden Parametern beschreiben: Die Menge der Zustände des HMM sei gegeben durch $\mathcal{S} = \{s_1, \ldots, s_{N_S}\}$. Die Anzahl der Zustände beschreibt N_S. Zu jedem Zeitpunkt t ist immer nur ein Zustand aktiv. Der aktive Zustand $q_t \in \mathcal{S}$ hängt aufgrund der Markovbedingung nur von dem direkt vorausgehenden Zustand q_{t-1} ab. Eine Zustandsfolge $\boldsymbol{q}$ der zu den Zeitpunkten $t = \{1,2,\ldots,T\}$ aktiven Zustände q_t ist definiert als

$$\boldsymbol{q} = q_1, q_2, \ldots, q_T \;; \quad q_t \in \mathcal{S} \tag{5.4}$$

Den Übergang eines Zustands in einen anderen Zustand beschreibt die Wahrscheinlichkeit $P(q_t|q_{t-1})$. Alle Übergangswahrscheinlichkeiten können in eine Matrix $\boldsymbol{A}$ der Dimension $N_S \times N_S$ geschrieben werden. Dabei müssen die Einträge den Randbedingungen $\sum_j a_{ij} = 1$ und $a_{ij} > 0$ für alle a_{ij} genügen.

$$\boldsymbol{A} = [a_{ij}]_{N_S \times N_S} \quad \text{mit} \quad a_{ij} = P(q_t = s_j|q_{t-1} = s_i) \tag{5.5}$$

Zusätzlich muss angegeben werden, in welchen Zustand mit der Generation eines Musters begonnen wird. Dies geschieht mit dem Zustandseinsprungsvektor $\boldsymbol{\pi} = [\pi_i]_{N_S \times 1}$:

$$\pi_i = P(q_1 = s_i)\;, \quad \sum_i \pi_i = 1 \tag{5.6}$$

In Abbildung 5.1 ist ein Links-Rechts-Modell gezeigt, bei dem der Einsprungsvektor $\boldsymbol{\pi}$ nur an der ersten Stelle einen von Null verschiedenen Eintrag hat. Die A-Matrix des Links-Rechts-Modells hat eine obere Dreiecksform. Ist die A-Matrix voll besetzt, sind also auch

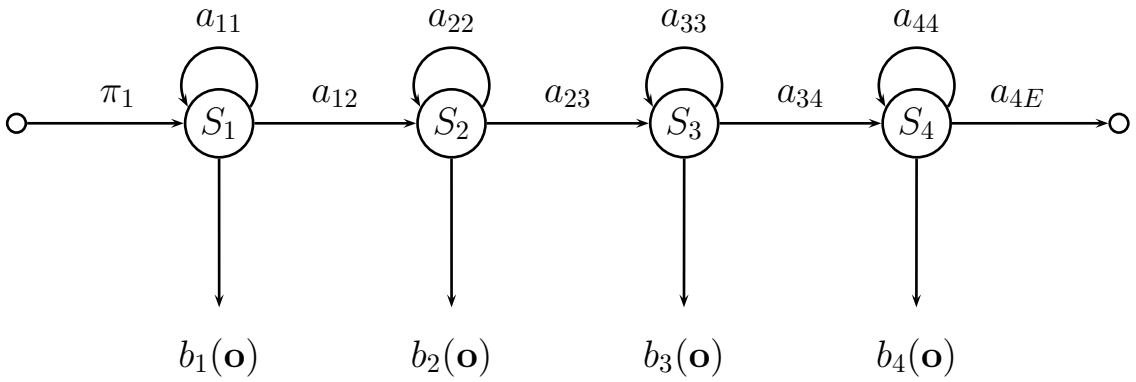


Abbildung 5.1: *Links-Rechts-HMM mit vier Zuständen*

Rücksprünge erlaubt und von jedem Zustand jeder andere Zustand erreichbar, so spricht man von einem *ergodischen* HMM.
Ein Beobachter sieht während des Durchlaufens einer Zustandsfolge nur die emittierten Observationen $\boldsymbol{O} = \boldsymbol{o}_1, \ldots, \boldsymbol{o}_T$, während die Abfolge der tatsächlich eingenommenen Zustände $\boldsymbol{q}$ verborgen bleibt. Dies erklärt auch das Präfix „Hidden“, da nicht unmittelbar nachvollziebar ist, welche Zustandsfolge eine Beobachtung generiert hat. Dabei gilt, dass die Produktion einer Observation nur vom aktuellen Zustand abhängt. Dies setzt die implizite Annahme der Unabhängigkeit der Beobachtungen voraus. Damit gilt $P(\boldsymbol{o}_t|\boldsymbol{o}_1, \ldots, \boldsymbol{o}_{t-1}, q_1, \ldots, q_t) = P(\boldsymbol{o}_t|q_t)$. Die Beobachtungswahrscheinlichkeit kann somit geschrieben werden als

$$\boldsymbol{B} = [b_i]_{N_S \times 1} \,, \quad \text{mit} \quad b_i(\boldsymbol{o}) = P(\boldsymbol{o}_t|q_t) \tag{5.7}$$

Durch die Angabe der Parameter $\boldsymbol{\pi}, \boldsymbol{A}$ und $\boldsymbol{B}$ kann ein HMM λ vollständig beschrieben werden.

$$\lambda = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{B}) \tag{5.8}$$

Die Beobachtungswahrscheinlichkeit kann diskret oder kontinuierlich gestaltet sein. Diese beiden typischen Verteilungen werden im Folgenden vorgestellt.

Diskrete HMM

Beschreiben die Beobachtungswahrscheinlichkeiten diskrete Symbole, so kann zu jedem Zeitpunkt ein Zustand s_i mit der Wahrscheinlichkeit $b_j(m)$ eine Beobachtung v_m aus dem Symbolvorrat $\mathcal{V} = \{v_1, \ldots, v_M\}$ erzeugen, d. h. es gilt:

$$b_i(m) = P(\boldsymbol{o}_t = v_m|q_t = s_i) \tag{5.9}$$

Kontinuierliche HMM

Bei Verwendung von kontinuierlichen HMM wird die Wahrscheinlichkeitsdichtefunktion durch Überlagerung von mehreren Normalverteilungen $\mathcal{N}$ erzeugt. Wählt man die Anzahl

der Gauß-Mixturen hoch genug, so kann im Prinzip fast jede gewünschte Dichtefunktion approximiert werden (vgl. Kapitel 3.1.2).

$$b_i(\boldsymbol{o}_t) = P(\boldsymbol{o}_t|s_i) = \sum_{m=1}^{N_M} c_{im}\,\mathcal{N}(\boldsymbol{o}_t|\boldsymbol{\mu}_{im}, \boldsymbol{\Sigma}_{im}) \tag{5.10}$$

$$\mathcal{N}(\boldsymbol{o}_t|\boldsymbol{\mu}_{im}, \boldsymbol{\Sigma}_{im}) = \frac{1}{\sqrt{(2\pi)^{N_M}|\boldsymbol{\Sigma}_{im}|}} \cdot e^{-\frac{1}{2}(\boldsymbol{o}_t-\boldsymbol{\mu}_{im})^T\boldsymbol{\Sigma}_{im}^{-1}(\boldsymbol{o}_t-\boldsymbol{\mu}_{im})} \tag{5.11}$$

Die K Verteilungen werden jeweils mit einem Gewicht c_{ik} beaufschlagt und addiert. Die Gewichte unterliegen dabei der Bedingung:

$$c_{im} \geq 0, \qquad \sum_{m=1}^{N_M} c_{im} = 1 \tag{5.12}$$

In dieser Arbeit werden zur Segmentierung von Gruppenaktionen stets kontinuierliche HMM eingesetzt.

5.1.2 Berechnung der Produktionswahrscheinlichkeit

Soll eine Beobachtungssequenz $\boldsymbol{O}$ in einer Klassifizierungsaufgabe einer bestimmten Klasse zugeordnet werden, so ist es notwendig, die Produktionswahrscheinlichkeit $P(\boldsymbol{O}|\lambda)$ eines HMM zu berechnen. Dasjenige HMM, das die größte Produktionswahrscheinlichkeit liefert, bestimmt die Klassenzugehörigkeit. Für die Pfadwahrscheinlichkeit, die Emission $\boldsymbol{O}$ mit der Abfolge der Zustände $\boldsymbol{q} = q_1, \ldots, q_T$ zu beobachten, gilt:

$$\begin{aligned} P(\boldsymbol{O}, \boldsymbol{q}|\lambda) &= \pi_{q_1} b_{q_1}(\boldsymbol{o}_1) \cdot a_{q_1 q_2} b_{q_2}(\boldsymbol{o}_2) \cdots a_{q_{t-1} q_t} b_{q_t}(\boldsymbol{o}_t) \cdots a_{q_{T-1} q_T} b_{q_T}(\boldsymbol{o}_T) \\ &= \pi_{q_1} b_{q_1}(\boldsymbol{o}_1) \cdot \prod_{t=2}^{T} a_{q_{t-1} q_t} b_{q_t}(\boldsymbol{o}_t) \end{aligned} \tag{5.13}$$

Die Wahrscheinlichkeit $p(\boldsymbol{O}|\lambda)$ ist die Summe der Wahrscheinlichkeiten aller Pfade, die zur gewünschten Beobachtung $\boldsymbol{O}$ führen:

$$P(\boldsymbol{O}|\lambda) = \sum_{\boldsymbol{q}\in\mathcal{Q}} P(\boldsymbol{O}, \boldsymbol{q}|\lambda) \tag{5.14}$$

Gleichung (5.13) in Gleichung (5.14) eingesetzt ergibt:

$$P(\boldsymbol{O}|\lambda) = \sum_{\boldsymbol{q}\in\mathcal{Q}} \pi_{q_1} b_{q_1}(\boldsymbol{o}_1) \cdot \prod_{t=2}^{T} a_{q_{t-1} q_t} b_{q_t}(\boldsymbol{o}_t) \tag{5.15}$$

Hierbei beschreibt $\mathcal{Q}$ die Menge aller möglichen Pfade durch das Modell λ. Diese Methode der Berechnung der Produktionswahrscheinlichkeit ist aufgrund der Komplexität praktisch nicht umsetzbar. Es müssen näherungsweise $2T \cdot N^T$ Berechnungen durchgeführt werden [Rab89]. Deshalb wird in der Praxis oftmals der Vorwärts-Algorithmus eingesetzt, mit dessen Hilfe die Komplexität auf N^2T Schritte reduziert werden kann.

Anders als bei der Berechnung der Wahrscheinlichkeit nach Gleichung (5.15) wird beim Vorwärts-Algorithmus die Wahrscheinlichkeit eines Teilpfades nur einmal berechnet. Als Vorwärts-Wahrscheinlichkeit $\alpha_t(i)$ wird zunächst definiert:

$$\alpha_t(i) = P(\boldsymbol{o}_1, \boldsymbol{o}_{2,} \ldots, \boldsymbol{o}_t, q_t = s_i|\lambda) \tag{5.16}$$

Das ist die Wahrscheinlichkeit, dass die Teilbeobachtung $\{\boldsymbol{o}_1, \boldsymbol{o}_{2,} \ldots, \boldsymbol{o}_t\}$ gemacht wurde und sich das HMM im Zustand s_i befindet. Diese Vorwärts-Wahrscheinlichkeit lässt sich nach der Initialisierung rekursiv berechnen:

1. Initialisierung:
$$\alpha_1(i) = \pi_i b_i(\boldsymbol{o}_1) \tag{5.17}$$

2. Induktion:
$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N_S} \alpha_t(i)\, a_{ij}\right] b_j(\boldsymbol{o}_{t+1}), \quad 1 \leq t \leq T-1,\ 1 \leq j \leq N_S \tag{5.18}$$

3. Abschluss:
$$P(\boldsymbol{O}|\lambda) = \sum_{i=1}^{N_S} \alpha_T(i) \tag{5.19}$$

Dieser Algorithmus stellt eine effiziente Methode für die Berechnung der Wahrscheinlichkeit $P(\boldsymbol{O}|\lambda)$ dar.

5.1.3 Training von HMM

Werden die Parameter λ eines HMM geeignet gewählt, so sind HMM in der Lage, Merkmalssequenzen robust zu klassifizieren, ohne durch Störungen in den Merkmalen beeinträchtigt zu werden. Die Bestimmung der Parameter $\lambda = (\boldsymbol{\pi}, \boldsymbol{A}, \boldsymbol{b})$ erfolgt häufig mit dem Baum-Welch Algorithmus, der hier in seinen Grundzügen vorgestellt werden soll.
Zur Bestimmung der geschätzten Parameter $\overline{\lambda} = (\overline{\pi}, \overline{\boldsymbol{A}}, \overline{\boldsymbol{b}})$ wird neben der Vorwärts-Wahrscheinlichkeit $\alpha_t(i)$ zusätzlich die Rückwärts-Wahrscheinlichkeit $\beta_t(i)$ benötigt, die angibt, mit welcher Wahrscheinlichkeit eine Restbeobachtung $\{\boldsymbol{o}_{t+1}, \boldsymbol{o}_{t+2}, \ldots, \boldsymbol{o}_T\}$ emittiert wird, wenn sich das HMM zum Zeitpunkt t im Zustand s_i befindet:

$$\beta_t(i) = P(\boldsymbol{o}_{t+1}, \boldsymbol{o}_{t+2}, \ldots, \boldsymbol{o}_T|q_t = s_i, \lambda) \tag{5.20}$$

Diese Rückwärts-Wahrscheinlichkeit lässt sich analog zu den Vorwärtswahrscheinlichkeiten rekursiv berechnen:

1. Initialisierung
$$\beta_T(i) = 1, \quad 1 \leq i \leq N_S \tag{5.21}$$

2. Induktion
$$\beta_t(i) = \sum_{j=1}^{N_S} a_{ij} b_j(\boldsymbol{o}_{t+1}) \beta_{t+1}(j), \quad t = T-1, T-2, \ldots, 1, \quad 1 \leq i \leq N_S \tag{5.22}$$

Zusätzlich werden die ergänzende Variablen $\gamma_t(i) = P(q_t = s_i|\boldsymbol{O}, \lambda)$, die Wahrscheinlichkeit, dass sich das HMM zum Zeitpunkt t im Zustand s_i befindet, und die zusätzliche Variable $\xi_t(i,j) = P(q_t = s_i, q_{t+1} = s_j|\boldsymbol{O}, \lambda)$, die Wahrscheinlichkeit, dass sich das HMM zum Zeitpunkt t im Zustand s_i und zum Zeitpunkt $t+1$ im Zustand s_j befindet, definiert:

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{P(\boldsymbol{O}|\lambda)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{N_S} \alpha_t(i)\beta_t(i)} \tag{5.23}$$

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij}\, b_j(\boldsymbol{o}_{t+1})\, \beta_{t+1}(j)}{P(\boldsymbol{O}|\lambda)} = \frac{\alpha_t(i)\, a_{ij}\, b_j(\boldsymbol{o}_{t+1})\, \beta_{t+1}(j)}{\sum_{i=1}^{N_S} \alpha_t(i)\beta_t(i)} \tag{5.24}$$

Damit können Schätzformeln für die einzelnen Parameter formuliert werden [Rab89]:

$$\overline{\boldsymbol{\pi}}_i = \text{„erwartete Verweildauer im Zustand } s_i \text{ zum Zeitpunkt } (t=1)\text{“} = \gamma_1(i) \tag{5.25}$$

$$\overline{a}_{ij} = \frac{\text{„erwartete Anzahl von Sprüngen von } s_i \text{ nach } s_j\text{“}}{\text{„erwartete Sprünge aus } s_i\text{“}} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \tag{5.26}$$

$$\overline{b}_t(k) = \frac{\text{„erwartete Verweildauer im Zustand } s_j \text{ mit Beobachtung von } v_k\text{“}}{\text{„erwartete Verweildauer im Zustand } s_j\text{“}} = \frac{\sum_{\substack{t=1 \\ o_t = v_k}}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)} \tag{5.27}$$

Da sich die Gleichungen (5.25) bis (5.27) auch durch Ableitung aus dem EM-Algorithmus[1] [Dem77] ergeben, ist sichergestellt, dass $P(\boldsymbol{O}|\overline{\lambda}) \geq P(\boldsymbol{O}|\lambda)$ gilt. Durch iterative Anwendung der Gleichungen (5.25) bis (5.27) wird ein Parametersatz λ gefunden, der die Wahrscheinlichkeit $P(\boldsymbol{O}|\lambda)$ maximiert. Zu beachten ist, dass beim Training ein lokales Maximum gefunden wird, welches nicht das globale Maximum sein muss. Die hier gezeigten Formeln gelten für diskrete Beobachtungen; für kontinuierliche HMM können entsprechende Gleichungen aufgestellt werden [Rab89]. Eine Herleitung der Gleichungen ist unter anderem bei Bilmes [Bil98] zu finden.

5.1.4 Automatische Segmentierung mit dem Viterbi-Algorithmus

Die bisher erwähnten Verfahren zur Bestimmung von Produktionswahrscheinlichkeiten setzten eine vorsegmentierte Merkmalssequenz voraus. Durch eine geschickte Verkettung mehrerer HMM ist es jedoch möglich, eine in gewissem Sinne optimale Zuordung unterschiedlicher HMM auf unterschiedliche Abschnitte einer unsegmentierten Merkmalssequenz vorzunehmen.

[1] Der *E*xpectation-*M*aximization-Algorithmus berechnet iterativ verbesserte Parameter in zwei Schritten: der Erwartungswert der zu optimierenden Größe wird aus alten Parametern berechnet (E-Schritt), daraus wird eine verbesserte Schätzung der Parameter abgeleitet (M-Schritt).

Das Vorgehen entspricht weitgehend dem in der automatischen Spracherkennung. Die kleinsten Einheiten bei dem hier behandelten Problem der Erkennung von Gruppenaktionen stellen die Gruppenaktionen selbst dar. Können die Modelle der einzelnen Gruppenaktionen unter Umständen noch isoliert trainiert werden, so müssen bei der Dekodierung alle Modelle berücksichtigt werden. Dies geschieht dadurch, dass alle HMM hintereinander gehängt werden, d. h. es wird ein Übergang vom letzten Zustand eines HMM zum ersten Zustand der anderen HMM erzeugt. Mit diesem Super-HMM können prinzipiell alle Kombinationen von Gruppenaktionen gebildet werden. Die Besetzung der neu erzeugten Übergänge mit geeigneten Wahrscheinlichkeiten erfolgt meist unter Verwendung einer Grammatik und einer Statistik des Trainingsmaterials. Mit Hilfe des Viterbi-Algorithmus [Vit67] kann die optimale Zustandssequenz aller HMM, und damit die optimale Zerlegung der Sequenz in Gruppenaktionen bestimmt werden. Das Ziel besteht darin, einen optimalen Pfad durch das Super-HMM zu finden. Die Dekodierung erfolgt dabei in ähnlicher Weise wie die Berechnung der Produktionswahrscheinlichkeit, wobei neben der Berechnung einer vorwärts gerichteten Größe $\delta_t(i)$, definiert als

$$\delta_t(i) = \max_{q_1,\ldots,q_{t-1}} P[q_1, q_2, \ldots, q_{t-1} = i, \boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_t|\lambda] \tag{5.28}$$

auch eine Information über den Vorgängerzustand in einem Array $\psi_t(j)$ festgehalten werden muss. Der Viterbi-Algorithmus kann wie folgt dargestellt werden:

1. Initialisierung:

$$\delta_1(i) = \pi_i \, b_i(\boldsymbol{o}_1), \quad 1 \leq i \leq N_S \tag{5.29}$$

$$\psi_1(t) = 0, \quad 1 \leq i \leq N_S \tag{5.30}$$

2. Rekursion:

$$\delta_t(j) = \max_{1\leq i\leq N_S} \left[\delta_{t-1}(i)\, a_{ij}\right] b_i(\boldsymbol{o}_t), \quad 2 \leq t \leq T,\ 1 \leq j \leq N_S \tag{5.31}$$

$$\psi_t(j) = \arg\max_{1\leq i\leq N_S} \left[\delta_{t-1}(i)\, a_{ij}\right], \quad 2 \leq t \leq T,\ 1 \leq j \leq N_S \tag{5.32}$$

3. Abschluss:

$$P^* = \max_{1\leq i\leq N_S} \delta_T(i) \tag{5.33}$$

$$q_T^* = \arg\max_{1\leq i\leq N_S} \delta_T(i) \tag{5.34}$$

4. Backtracking:

$$q_t^* = \psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \ldots, 1 \tag{5.35}$$

Durch das Backtracking am Ende des Viterbi-Algorithmus kann auf die aktiven Zustände des Super-HMM und damit auf die erkannte Sequenz von Klassen zurückgeschlossen werden.

5.2 Segmentierung mit Hidden Conditional Random Fields

In diesem Kapitel wird das Konzept der *Hidden Conditional Random Fields* (HCRF) eingeführt, die dann auch zur Segmentierung von Besprechungen eingesetzt werden. HCRF können als verallgemeinernde Erweiterung von Hidden-Markov-Modellen angesehen werden. Im Folgenden werden zunächst die Grundlagen von Conditional Random Fields (CRF) erläutert. Anschließend werden die Modelle um verborgenen Zustände erweitert zu HCRF. Danach werden die Ergebnisse bei der Anwendung auf das Problem der Meeting-Segmentierung erläutert.

5.2.1 Conditional Random Fields

In vielen Bereichen der Signalverarbeitung und Mustererkennung besteht die Aufgabe, einer Sequenz von beobachteten Daten Bezeichnungen zuzuordnen, sogenannte Labels. Diese Aufgabe besteht zum Beispiel in der Bioinformatik, Computerlinguistik und Spracherkennung. Ein Beispiel für die Aufgabe der Zuordnung eines Lables zu einer Beobachtung ist der Einsatz in der natürlichen Sprachverarbeitung (vgl. [Wal04a]), bei dem zu jedem beobachteten Wort eines Satzes die zugehörige Wortbedeutung innerhalb dieses Satzes bestimmt wird (also Substantive, Verben, Adjektive usw.), zum Beispiel (nach [Gut05]):

```
Beobachtung: [Der],     [rote],      [Ball],   [platzt],     [.]
Zuordnung: [Artikel], [Adjektiv], [Substantiv], [Verb], [Satzzeichen]
```

Aufgrund dieser Zuordnung der Funktion eines Wortes kann der Inhalt eines Satzes unter Umständen besser erfasst und automatisch ausgewertet werden.
Für das Tagging von Sequenzen werden Conditional Random Fields (CRF) [Laf01] eingesetzt, eine Variante eines ungerichteten graphischen Modells. Eine von Gunawardana [Gun05] entwickelte Erweiterung der CRF um verborgene Zustände zu Hidden Conditional Random Fields ermöglicht es, dass nicht zu jeder einzelnen Beobachtung ein Label vergeben werden muss, sondern dass ganze Sequenzen mit einem Label bezeichnet werden können. Dadurch ist es möglich, HCRF auf die in dieser Arbeit beschriebene Aufgabe der Meeting Event Segmentierung und Erkennung anzuwenden.
Im nächsten Abschnitt sollen zunächst ungerichtete Graphische Modelle als Grundlage von CRF und auch HCRF erläutert werden.

Ungerichtete Graphische Modelle

Ungerichtete Graphische Modelle, auch Markov Random Fields genannt, bestehen aus einem Graph $G = (V, E)$, wobei V eine endliche Menge von Knoten und E eine endliche Menge von ungerichteten Kanten darstellt[2]. Die Knoten V repräsentieren kontinuierliche

[2] V steht für *vertex*, engl. für Knoten, E für *edge*, engl. für Kante

oder diskrete Zufallsvariablen. Eine Kante $e \in E$ ist eine Menge von zwei Elementen $\{v_1, v_2\}$ mit $v_1, v_2 \in V$ und $v_1 \neq v_2$. Ein beispielhaftes Modell ist in Abbildung 5.2 gezeigt.

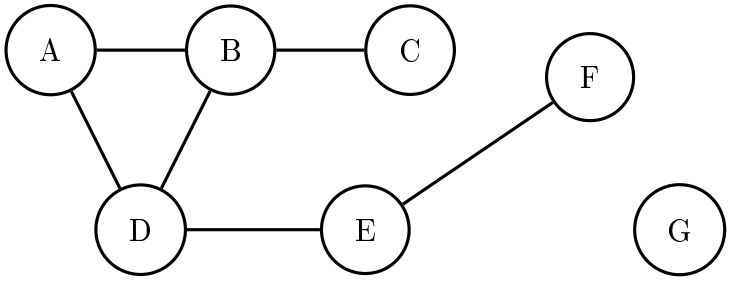


Abbildung 5.2: *Ein ungerichteter Graph*

Die Struktur des Graphen legt die Beziehungen zwischen den Knoten untereinander fest. Knoten, die nicht durch eine Kante verbunden sind, sind voneinander unabhängig.
Beziehungen von Knoten in einem ungerichteten Graph können über sogenannte Potentialfunktionen ausgedrückt werden. Jede Potentialfunktion Ψ_C wird auf einem Subset der Knoten, einer Clique C definiert. Eine Clique C ist eine Teilmenge $C \subseteq V$, mit der Eigenschaft, dass alle Kanten direkt paarweise miteinander verbunden sind; es gilt also: $\{c_i, c_j\} \in E, \forall c_i, c_j \in C$, mit $i \neq j$. Um die Unabhängigkeit der Knoten zu berücksichtigen, muss sichergestellt sein, dass unabhängige Knoten nicht in der selben Potentialfunktion vorkommen. Deshalb verwendet man Potentialfunktionen, die auf Knoten definiert sind, die eine maximale Clique bilden. Eine Clique heißt maximal, wenn sie selbst nicht mehr in einer anderen Clique enthalten ist. Im einfachsten Fall eines Markov-Modells, einer Kettenstruktur, wird eine Potentialfunktion immer auf zwei benachbarten Knoten definiert (vgl. Abbildung 5.3). $\boldsymbol{X}$ entspricht einer Eingabesequenz, die Variablen Y_i bilden

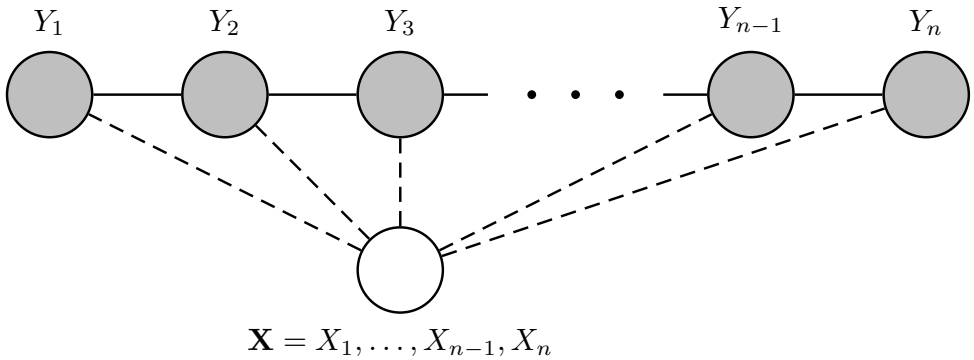


Abbildung 5.3: *Graphische Struktur eines einfachen CRF mit Kettenstruktur. Variablen von schattierten Knoten werden vom Modell erzeugt.*

eine gleich lange Ausgabesequenz. Die Potentiale sind Funktionen, mit dem Wertebereich $\mathbb{R}^+$. Das Ergebnis der Potentialfunktionen sind keine Wahrscheinlichkeiten, da der Wertebereich nicht beschränkt ist. Jedoch gilt, dass je höher der Wert eines Potentials ist, desto besser diese Belegung ist. Aus den Potentialen kann die Wahrscheinlichkeit einer Variablenbelegung berechnet werden [Cli90]. Das Produkt der Potentialfunktionen wird

dazu normiert. Es gilt:

$$P(Y_1, Y_2, \ldots, Y_n) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \Psi_C(Y_C) \tag{5.36}$$

Dabei ist $\mathcal{C}$ die eindeutige Menge aller maximalen Cliquen von G. Für den Normierungsfaktor Z gilt:

$$Z = \sum_{Y_C} \prod_{C \in \mathcal{C}} \Psi_C(Y_C) \tag{5.37}$$

Das Maximum-Entropie Prinzip

Die Wahl der Potentialfunktionen basiert nach Lafferty et al. [Laf01] im Wesentlichen auf dem Prinzip der maximalen Entropie. Es wird diejenige Wahrscheinlichkeitsverteilung gesucht, welche die Entropie [Sha48]

$$H(p) = \sum_i -p_i \log p_i \tag{5.38}$$

unter der Berücksichtigung von Randbedingungen maximiert. Es kann gezeigt werden ([Ber96], [Del95], [Rat97]), dass die parametrische Form der Wahrscheinlichkeitsverteilung wie folgt aussieht:

$$P(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \exp\left(\sum_k \lambda_k f_k(\boldsymbol{x}, \boldsymbol{y})\right) \tag{5.39}$$

Dabei ist $Z(\boldsymbol{x})$ ein Normierungsfaktor und die λ_k können als Gewichtungsfaktoren für die Merkmalsfunktionen f_k aufgefasst werden. $\boldsymbol{x}$ und $\boldsymbol{y}$ beschreiben eine Beobachtungssequenz bzw. die dazu gehörige Ausgabesequenz.

Potentialfunktionen für CRFs

Aus dem Prinzip der Maximum-Entropie folgt, dass die Potentialfunktionen eines CRF eine Form besitzen müssen, die ähnlich wie Gleichung (5.39) aussieht. Daher kann jede Potentialfunktion allgemein definiert werden als:

$$\Psi_{Y_C}(\boldsymbol{y}_c) = \exp\left(\sum_k \lambda_k f_k(c, \boldsymbol{y}_c, \boldsymbol{x})\right) \tag{5.40}$$

Legt man eine Kettenstruktur zugrunde, wie in Abbildung 5.3, eine Struktur, die häufig für die Modellierung von sequenziellen Daten verwendet wird, so kann man die Potentialfunktionen aufteilen:

$$\Psi_{Y_C}(\boldsymbol{y}_c) = \exp\left(\sum_k \lambda_k f_k(\boldsymbol{y}_{i-1}, \boldsymbol{y}_i, \boldsymbol{x}) + \sum_k \mu_k g_k(\boldsymbol{y}_i, \boldsymbol{x})\right) \tag{5.41}$$

Der erste Teil, $f_k(\boldsymbol{y}_{i-1}, \boldsymbol{y}_i, \boldsymbol{x})$ beschreibt die Beziehungen der ganzen observierten Sequenz mit den Werten an Position i und $i-1$ der zugehörigen Ausgabesequenz, der rechte Teil,

$g_k(\boldsymbol{y}_i, \boldsymbol{x})$, ist ein Merkmal der Bezeichnung an der Position i und der Beobachtungssequenz. Damit kann die Wahrscheinlichkeit $P(\boldsymbol{y}|\boldsymbol{x})$ wie folgt geschrieben werden:

$$P(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{Z(\boldsymbol{x})} \exp\left(\sum_i \sum_k \lambda_k f_k(\boldsymbol{y}_{i-1}, \boldsymbol{y}_i, \boldsymbol{x}) + \sum_i \sum_k \mu_k g_k(\boldsymbol{y}_i, \boldsymbol{x})\right) \tag{5.42}$$

mit der Normierungskonstanten

$$Z(\boldsymbol{x}) = \sum_{\text{alleModelle}} \exp\left(\sum_i \sum_k \lambda_k f_k(\boldsymbol{y}_{i-1}, \boldsymbol{y}_i, \boldsymbol{x}) + \sum_i \sum_k \mu_k g_k(\boldsymbol{y}_i, \boldsymbol{x})\right) \tag{5.43}$$

CRF als Lösung des Label Bias Problems

Aufgrund der definierten Wahrscheinlichkeitsverteilungen eines Conditional Random Field besteht das Label Bias Problem hier nicht (vgl. [Laf01], [Wal02]). Dieses kann bei anderen probabilistischen Modellen auftreten. Durch die Normalisierung der Wahrscheinlichkeitsverteilungen in jedem Zustand ist es möglich, dass Teile des Eingangsmusters bei der Klassifikation nicht berücksichtigt werden, und stattdessen immer das während des Trainings am häufigsten beobachtete Ergebnis zurückgeliefert wird.
Conditional Random Fields sind allerdings so gestaltet, dass keine Normalisierung der Wahrscheinlichkeitsverteilungen nötig ist. Jeder Potentialfunktion steht die komplette Eingangssequenz zur Verfügung. Dadurch kann vermieden werden, dass sich das Modell vorzeitig auf eine Lösung beschränkt.

5.2.2 Hidden Conditional Random Fields

Conditional Random Fields sind gut geeignet für Aufgaben, bei denen die Ausgabesequenz ebenso lang ist wie die Eingangssequenz. Dies ist zum Beispiel beim „Part-of-speech“ Tagging der Fall (vgl. [McC00], [Laf01]). Jedem Symbol der Eingangssquenz wird ein Symbol der Ausgangssequenz zugeordnet. Bei allgemeinen Klassifikationsaufgaben, beispielsweise in der automatischen Spracherkennung, ist diese Voraussetzung nicht gegeben. Hier ist die genaue Verteilung von Eingangsdaten und Ausgangssymbolen „verborgen“. Aus diesem Grund werden hier vor allem HMM eingesetzt, die mit Hilfe des Viterbi-Algorithmus eine optimale Zuordnung der Zustände auf die Eingangssequenz durchführen können.
Erweitert man CRF nach einem Vorschlag von Quattoni et al. [Qua05] und Gunawardana et al. [Gun05] um verborgene Zustände, so erhält man Hidden Conditional Random Fields (HCRF). Eine Veranschaulichung als Graphisches Modell ist in Abbildung 5.4 gegeben. Im Unterschied zu HMM besitzen HCRF keine normierten Wahrscheinlichkeitsverteilungen für die Zustandsübergänge und die Ausgabewahrscheinlichkeiten. Aufgrund der verborgenen Zustände ist es möglich, ähnlich wie mit HMM, Sequenzen zu klassifizieren, ohne dass die Ausgabesequenzen die gleiche Länge besitzen wie die Eingabesequenzen.
HCRF werden mittlerweile in vielen Bereichen der Mustererkennung erfolgreich eingesetzt. Quattoni et al. [Qua05] verwendet solche Modelle zur Erkennung von Objekten. HCRF werden auch zur Gestenerkennung herangezogen [Qua06], [Wan06]. In der Sprachverarbeitung werden HCRF im Bereich der Phonemerkennung eingesetzt [Gun05].

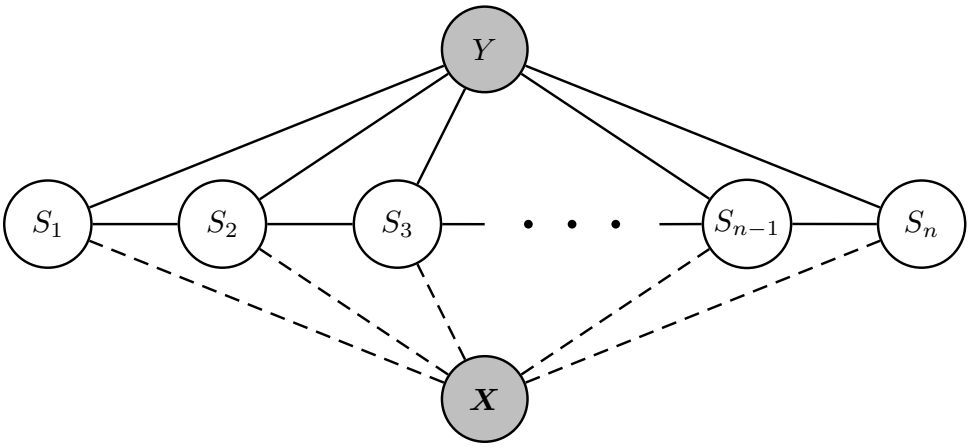


Abbildung 5.4: *Struktur eines HCRF als Graphisches Modell. Schattierte Knoten sind beobachtbar, während weiße Knoten verborgen (hidden) sind (nach [Qua05]).*

Modellbeschreibung

Das HCRF-Modell beschreibt die bedingte Wahrscheinlichkeit einer Klasse Ω_κ unter der Beobachtung $\boldsymbol{o} = (o_1, o_2, \ldots, o_T)$:

$$p(\Omega_\kappa|\boldsymbol{o};\boldsymbol{\lambda}) = \frac{1}{Z(\boldsymbol{o};\boldsymbol{\lambda})} \sum_{\boldsymbol{q}\in\mathcal{Q}} \exp\{\boldsymbol{\lambda}\cdot\boldsymbol{f}(\kappa,\boldsymbol{q},\boldsymbol{o})\} \tag{5.44}$$

Der Vektor $\boldsymbol{q} = (q_1, q_2, \ldots, q_T)$ beschreibt dabei eine verborgene Abfolge der Zustände, vergleichbar mit denen bei HMM. $\mathcal{Q}$ ist die Menge aller möglichen Pfade durch ein Modell. $\boldsymbol{\lambda}$ stellt den Parametervektor dar, den man auch als Gewichtsvektor auffassen kann. $\boldsymbol{f}$ ist der Vektor der ausreichenden Statistik, oder wie er im Kontext der CRF genannt wird, der Merkmalsvektor[3]. Die Funktion $Z(\boldsymbol{o};\boldsymbol{\lambda})$ dient zur Normierung, sodass das Modell eine normalisierte Wahrscheinlichkeit ausgibt, und ist definiert als:

$$Z(\boldsymbol{o};\boldsymbol{\lambda}) = \sum_{\kappa}\sum_{\boldsymbol{q}\in\mathcal{Q}} \exp\{\boldsymbol{\lambda}\cdot\boldsymbol{f}(\kappa,\boldsymbol{q},\boldsymbol{o})\} \tag{5.45}$$

Dem Vorschlag von Gunawardana [Gun05] folgend wird der Merkmalsvektor $\boldsymbol{f}$ so gewählt, dass die gleichen statistischen Abhängigkeiten modelliert werden wie es bei HMM der Fall

[3]In diesem Zusammenhang bezeichnet der Begriff „Merkmalsvektor“ den Vektor, der die statistischen Abhängigkeiten des Modells ausreichend beschreibt, nicht den Vektor der durch die Vorverarbeitung gewonnenen Signaldaten. Dieser Vektor wird in diesem Abschnitt als „Beobachtungsvektor“ bezeichnet.

ist. Dies führt zur folgenden Form:

$$f_{\kappa'}^{(LM)}(\kappa, \boldsymbol{q}, \boldsymbol{o}) = \delta(\kappa = \kappa') \qquad \forall \kappa' \tag{5.46a}$$

$$f_{ij}^{(Tr)}(\kappa, \boldsymbol{q}, \boldsymbol{o}) = \sum_{t=1}^{T} \delta(q_{t-1} = s_i)\, \delta(q_t = s_j) \qquad \forall s_i, s_j \tag{5.46b}$$

$$f_{i}^{(Occ)}(\kappa, \boldsymbol{q}, \boldsymbol{o}) = \sum_{t=1}^{T} \delta(q_t = s_i) \qquad \forall s_i \tag{5.46c}$$

$$f_{i}^{(M1)}(\kappa, \boldsymbol{q}, \boldsymbol{o}) = \sum_{t=1}^{T} \delta(q_t = s_i)\, o_t \qquad \forall s_i \tag{5.46d}$$

$$f_{i}^{(M2)}(\kappa, \boldsymbol{q}, \boldsymbol{o}) = \sum_{t=1}^{T} \delta(q_t = s_i)\, o_t^2 \qquad \forall s_i \tag{5.46e}$$

Hierbei bezeichnet $\delta(q_t = s_i)$ die Delta-Funktion, die gleich 1 ist, wenn $q_t = s_i$ gilt, andernfalls 0. $f_{\kappa}^{(LM)}$ ist 1, wenn die Klasse Ω_κ auftritt. Die Übergangsmerkmale $f_{ij}^{(Tr)}$ zählen die Anzahl der Übergänge von Zustand s_i nach s_j der Zustandsabfolge $\boldsymbol{q}$, während die Belegungsmerkmale $f_i^{(Occ)}$ die Häufigkeit des Zustandes s_i zählen. Die ersten und zweiten Momente $f_i^{(M1)}$ und $f_i^{(M2)}$ beschreiben die Summe bzw. die Summe der Quadrate der Beobachtungen, die mit dem Zustand s_i verbunden sind.

Da die beschreibende Statistik nur paarweise Nachbarschaften berücksichtigt, genügt die Zustandsabfolge der Markov-Bedingung. Aus diesem Grund können Verfahren der Dynamischen Programmierung, wie der Vorwärts-Rückwärts-Algorithmus und der Viterbi-Algorithmus, wie bei HMM angewendet werden.

5.2.3 HCRF als Generalisierung von HMM

Legt man wie oben beschrieben dem HCRF-Modell eine Kettenstruktur zugrunde und definiert den Merkmalsvektor derart, dass die Markov-Bedingung eingehalten wird, so kann man zeigen, dass bei geeigneter Wahl der Parameter $\boldsymbol{\lambda}$ ein HCRF äquivalent zu einem HMM ist (vgl [Gun05]). Da die Parameter in einer Exponentialfunktion stehen, wird beispielsweise $\lambda_{\kappa}^{(LM)}$ mit dem Logarithmus der Klassenauftrittswahrscheinlichkeit $P(\Omega_\kappa)$ gleichgesetzt. Gleiches gilt auch für die Zustandsübergänge $\lambda_{ij}^{(Tr)}$, die dem Logarithmus der Zustandsübergangswahrscheinlichkeiten a_{ij} entsprechen. Für die bei HMM häufig verwendete Normalverteilung mit diagonalen Kovarianzmatrizen, d. h. $\sigma_{ij} = 0$ für $i \neq j$, ergibt

sich folgende Umformung:

$$\begin{aligned}
\mathcal{N}(\boldsymbol{o}|\boldsymbol{\mu},\boldsymbol{\sigma}) &= \frac{1}{(2\pi)^{\frac{N}{2}}\sqrt{\prod_{i=1}^{N}\sigma_i^2}} \cdot e^{-\frac{1}{2}\sum_{i=1}^{N}\frac{(o_i-\mu_i)^2}{\sigma_i^2}} \\
&= e^{-\frac{1}{2}\log\left[(2\pi)^N\prod_{i=1}^{N}\sigma_i^2\right]} \cdot e^{-\frac{1}{2}\left[\sum_{i=1}^{N}\frac{o_i^2}{\sigma_i^2}+\sum_{i=1}^{N}\frac{-2o_i\mu_i}{\sigma_i^2}+\sum_{i=1}^{N}\frac{\mu_i^2}{\sigma_i^2}\right]} \\
&= e^{-\frac{1}{2}\left\{\log\left[(2\pi)^N\prod_{i=1}^{N}\sigma_i^2\right]+\sum_{i=1}^{N}\frac{\mu_i^2}{\sigma_i^2}\right\}} + e^{\sum_{i=1}^{N}\frac{\mu_i}{\sigma_i^2}o_i} + e^{-\frac{1}{2}\sum_{i=1}^{N}\frac{1}{\sigma_i^2}o_i^2} \\
&= e^{\lambda^{(Occ)}} + e^{\boldsymbol{\lambda}^{(M1)}\boldsymbol{o}} + e^{\boldsymbol{\lambda}^{(M2)}\boldsymbol{o}^2}
\end{aligned} \tag{5.47}$$

Damit ergibt sich folgende Korrespondenz der Parameter von HCRF und HMM:

$$\lambda_{\kappa}^{(LM)} = \log P(\Omega_{\kappa}) \qquad \forall\kappa \tag{5.48a}$$

$$\lambda_{ij}^{(Tr)} = \log a_{ij} \qquad \forall i,j \tag{5.48b}$$

$$\lambda_{i}^{(Occ)} = -\frac{1}{2}\left(\log 2\pi\sigma_i^2 + \frac{\mu_i^2}{\sigma_i^2}\right) \qquad \forall i \tag{5.48c}$$

$$\lambda_{i}^{(M1)} = \frac{\mu_i}{\sigma_i^2} \qquad \forall i \tag{5.48d}$$

$$\lambda_{i}^{(M2)} = -\frac{1}{2\sigma_i^2} \qquad \forall i \tag{5.48e}$$

Hierbei bezeichnet a_{ij} die Übergangswahrscheinlichkeiten, μ_i und σ_i die Mittelwerte bzw. Standardabweichung der Beobachtung und $P(\Omega_\kappa)$ die Klassenwahrscheinlichkeit.

Gleichung (5.44) ergibt mit dem Merkmalsvektor $\boldsymbol{f}$ aus Gleichung (5.46) eine gültige bedingte Wahrscheinlichkeit für jede Belegung des Parametervektors $\boldsymbol{\lambda}$. Allerdings entspricht nicht jeder Wert dem eines HMM. So kann zum Beispiel $\lambda_i^{(M2)}$ nicht-negativ sein, und die $\lambda_i^{(Occ)}$ und $\lambda_{ij}^{(Tr)}$ können Gewichte beinhalten, die einen speziellen Zustand oder Zustandsübergang betonen oder abschwächen. Indem HMM die gleichen Abhängigkeiten durch die gleichen statistischen Bindungen modellieren, bilden sie also eine Untermenge der von HCRF modellierbaren bedingten Wahrscheinlichkeiten. Aufgrund der großen Ähnlichkeit, können HCRF auch als abgeänderte HMM aufgefasst werden, wobei der größte Unterschied die fehlende Normierung darstellt. Die Übergänge $\lambda_i^{(Tr)}$ beinhalten, anders als bei HMM, zusätzlich Gewichtungen. Es kann zudem sein, dass die Exponentialfunktion von Gleichung (5.47) nicht normierbar ist. So stellen dann die „Emissionswahrscheinlichkeiten“ keine ordentliche Wahrscheinlichkeitsverteilung über der Emission $\boldsymbol{o}$ dar. Es können damit Verteilungen mit „negativen Varianzen“ gebildet werden [Gun05].

5.2.4 Berechnung der Modellwahrscheinlichkeit

Für die Berechnung der Modellwahrscheinlichkeit $p(\Omega_\kappa|\boldsymbol{O};\boldsymbol{\lambda})$ kann wie bei den HMM der Vorwärtsalgorithmus eingesetzt werden. Es wird eine Vorwärtswahrscheinlichkeit $\alpha_t(i)$ definiert als

$$\alpha_t(i) = P(\boldsymbol{o}_1\boldsymbol{o}_2\ldots\boldsymbol{o}_t, q_t = s_i|\boldsymbol{\lambda}) \tag{5.49}$$

das heißt, die Wahrscheinlichkeit der partiellen Beobachtungssequenz $\boldsymbol{O}_{1:t} = \boldsymbol{o}_1, \boldsymbol{o}_2, \ldots \boldsymbol{o}_t$ bis zum Zeitpunkt t *und* Zustand S_i zum Zeitpunkt t, unter der Bedingung des Modells λ. Die Wahrscheinlichkeiten $\alpha_t(i)$ können induktiv berechnet werden (vgl. Gleichungen (5.17) bis (5.19)):

1. Initialisierung:

$$\alpha_1(i) = e^{\lambda_{01}^{(Tr)}} \cdot e^{\left(\lambda_i^{(Occ)} + \boldsymbol{\lambda}_i^{(M1)}\cdot\boldsymbol{o}_1 + \boldsymbol{\lambda}_i^{(M2)}\cdot\boldsymbol{o}_1^2\right)}; \qquad \forall i \in S \tag{5.50}$$

2. Induktion:

$$\alpha_{t+1}(j) = \left(\sum_{i=1}^{N_S} \alpha_t(i) e^{\lambda_{ij}^{Tr}}\right) \cdot e^{\left(\lambda_j^{(Occ)} + \boldsymbol{\lambda}_j^{(M1)}\cdot\boldsymbol{o}_{t+1} + \boldsymbol{\lambda}_j^{(M2)}\cdot\boldsymbol{o}_{t+1}^2\right)}; \quad 1 \le t \le T-1, 1 \le j \le N_S \tag{5.51}$$

3. Abschluss:

$$\tilde{P}(\Omega_\kappa|\boldsymbol{O},\boldsymbol{\lambda}) = \sum_{i=1}^{N_S} \alpha_T(i) \tag{5.52}$$

Abschließend müssen die Modellwahrscheinlichkeiten $\tilde{P}(\Omega_\kappa|\boldsymbol{O},\boldsymbol{\lambda})$ normiert werden, so dass sich eine echte Wahrscheinlichkeitsverteilung ergibt:

$$P(\Omega_\kappa|\boldsymbol{O},\boldsymbol{\lambda}) = \frac{\tilde{P}(\Omega_\kappa|\boldsymbol{O},\boldsymbol{\lambda})}{\sum_{\kappa=1}^{K} \tilde{P}(\Omega_\kappa|\boldsymbol{O},\boldsymbol{\lambda})} \tag{5.53}$$

Die Rückwärtswahrscheinlichkeiten $\beta_t(i) = P(\boldsymbol{o}_{t+1}, \boldsymbol{o}_{t+2}, \ldots, \boldsymbol{o}_T|q_t = s_i, \boldsymbol{\lambda})$, die für das Training wichtig sind, können analog dazu wie folgt berechnet werden:

1. Initialisierung:

$$\beta_T(i) = 1; \qquad 1 \le i \le N_S \tag{5.54}$$

2. Induktion:

$$\beta_t(i) = \sum_{j=1}^{N_S} e^{\lambda_{ij}^{(Tr)}} \cdot e^{\lambda_i^{(Occ)} + \boldsymbol{\lambda}_i^{(M1)}\cdot\boldsymbol{o}_t + \boldsymbol{\lambda}_i^{(M2)}\cdot\boldsymbol{o}_t^2} \cdot \beta_{t+1}(j) \tag{5.55}$$

5.2.5 Training von HCRF

Die Parameter der HCRF sind mit Hilfe von ausreichenden Trainingsdaten mit geeigneten Trainingsverfahren zu bestimmen. Ein EM-Training wie bei HMM ist hier aufgrund der fehlenden Normierung der Parameter, und dem damit verbundenen Fehlen der Randbedingungen nicht möglich. Aus diesem Grund greift man auf gradientenbasierte Verfahren zurück [Mah06].
Das Ziel beim Gradiententraining besteht darin, die Log-Likelihood $\mathcal{L}$ der Trainingsdaten

$$\mathcal{L}(\lambda) = \sum_{n=1}^{N} \log p(\Omega_\kappa^{(n)} | \boldsymbol{o}^{(n)}; \lambda) \tag{5.56}$$

zu maximieren. Dazu ist es notwendig, den Gradienten $\nabla\mathcal{L}(\lambda)$ zu bestimmen. Für jede präsentierte Trainingssequenz wird der Gradient berechnet und werden die Parameter nach folgender Regel verändert:

$$\lambda^{(n+1)} = \lambda^{(n)} + \eta^{(n)} \nabla_\lambda \log p(\Omega_\kappa^{(n)} | \boldsymbol{o}^{(n)}; \lambda^{(n)}) \tag{5.57}$$

Hierbei ist $\eta^{(n)}$ die Lernrate, die angibt, mit welcher Schrittweite die Parameter verändert werden.
Um beim Training nicht ein Modell zu bevorzugen, werden in dieser Arbeit HCRF mit Maximum Mutual Information (MMI) trainiert (vgl. [Bah86], [Dro05]). Im Unterschied zu Maximum Likelihood Training (ML) werden beim MMI die Parameter gesucht, mit denen die bestmöglichen Trennbarkeit der zu unterscheidenden Klassen erreicht wird. Die Beschreibung des Trainings folgt der von Warakagoda [War94]:
Alle Modelle, die mit MMI trainiert werden sollen, werden zusammengefasst in ein Set $\Lambda = \{\lambda_\kappa,\ 1 \leq \kappa \leq K\}$. K bezeichnet dabei die Anzahl der Klassen. Ziel des Trainings ist es, die Unsicherheit einer Klasse Ω_κ bezüglich der Beobachtungssequenz $\boldsymbol{o}$ zu minimieren. Diese Aufgabe ist äquivalent zur Minimierung der Information

$$I(\Omega_\kappa | \boldsymbol{o}, \lambda_\kappa) = -\log p(\Omega_\kappa | \boldsymbol{o}, \lambda_\kappa) \tag{5.58}$$

in Bezug auf λ_κ. Gleichung (5.58) kann mit Hilfe des Satzes von Bayes umgeschrieben werden, so dass die zu minimierende Zielfunktion E_{MMI} wie folgt lautet:

$$\begin{aligned} E_{\text{MMI}} &= -\log p(\Omega_\kappa | \boldsymbol{o}, \lambda_\kappa) \\ &= -\log \frac{p(\Omega_\kappa, \boldsymbol{o} | \lambda_\kappa)}{p(\boldsymbol{o} | \lambda_\kappa)} \\ &= -\log \frac{p(\Omega_\kappa, \boldsymbol{o} | \lambda_\kappa)}{\sum_\kappa p(\Omega_\kappa, \boldsymbol{o} | \lambda_\kappa)} \end{aligned} \tag{5.59}$$

Dies entspricht genau dem negativen Logarithmus der Modellwahrscheinlichkeit aus Gleichung (5.53).

Zur einfacheren Beschreibung sei im Folgenden der Zähler mit L_κ und der Nenner mit L_{tot} bezeichnet:

$$L_\kappa := p(\Omega_\kappa, \boldsymbol{o}|\lambda_\kappa) = \sum_{i=1}^{N_S} \alpha_t(i)\beta_t(i)\Big|_\kappa \tag{5.60}$$

$$L_{\text{tot}} := \sum_\kappa p(\Omega_\kappa, \boldsymbol{o}|\lambda_\kappa) = \sum_\kappa \left[\sum_{i=1}^{N_S} \alpha_t(i)\beta_t(i)\right]_\kappa \tag{5.61}$$

so dass gilt:

$$E_{\text{MMI}} = -\log \frac{L_\kappa}{L_{\text{tot}}} \tag{5.62}$$

Die Parameter seien mit ϑ bezeichnet. Dann gilt für einen beliebigen neuen Parameter:

$$\vartheta^{\text{neu}} = \vartheta^{\text{alt}} - \eta \left[\frac{\partial E_{\text{MMI}}}{\partial \vartheta}\right]_{\vartheta=\vartheta^{\text{alt}}} \tag{5.63}$$

Zu Berechnen ist nun also die partielle Ableitung von E_{MMI} nach den Parametern ϑ. Für beliebiges ϑ gilt:

$$\frac{\partial E_{\text{MMI}}}{\partial \vartheta} = \frac{1}{L_{\text{tot}}}\frac{\partial L_{\text{tot}}}{\partial \vartheta} - \frac{1}{L_\kappa}\frac{\partial L_\kappa}{\partial \vartheta} \tag{5.64}$$

Unter Verwendung der Berechnungsvorschrift für die Vorwärtswahrscheinlichkeit $\alpha_t(i) = b_j(o_t)\sum_{i=1}^{N_S} \alpha_{t-1}(i)a_{ij}$ (vgl. [Rab89]), kann die Ableitung nach den Übergangsparametern durch Anwendung der Kettenregel wie folgt ermittelt werden:

$$\frac{\partial L}{\partial a_{ij}} = \sum_{t=1}^{T} \frac{\partial L}{\partial \alpha_t(j)} \cdot \frac{\partial \alpha_t(j)}{\partial a_{ij}} \cdot \frac{\partial a_{ij}}{\partial \lambda_{ij}^{(Tr)}} \tag{5.65}$$

Mit

$$\frac{\partial \alpha_t(j)}{\partial a_{ij}} = b_j(o_t)\alpha_{t-1}(i) \tag{5.66}$$

und

$$\frac{\partial L_{\text{tot}}}{\partial \alpha_t(j)} = \beta_t(j) \tag{5.67}$$

bzw.

$$\frac{\partial L_\kappa}{\partial \alpha_t(j)} = \beta_t(j) \cdot \delta_{k\kappa} \tag{5.68}$$

gilt:

$$\frac{\partial L_{\text{tot}}}{\partial a_{ij}} = \sum_{t=1}^{T} \beta_t(j) \cdot b_j(o_t) \cdot \alpha_{t-1}(i) \tag{5.69}$$

$$\frac{\partial L_\kappa}{\partial a_{ij}} = \delta_{k\kappa} \sum_{t=1}^{T} \beta_t(j) \cdot b_j(o_t) \cdot \alpha_{t-1}(i) \tag{5.70}$$

Hierbei ist $\delta_{k\kappa} = 1$, falls die Klasse k der Sequenz $\boldsymbol{o}$ dem Modell κ entspricht und 0 sonst. Damit kann nun die Ableitung nach $\lambda_{ij}^{(Tr)}$ vollständig beschrieben werden als:

$$\frac{\partial E_{MMI}}{\partial \lambda_{ij}^{(Tr)}} = \left[\frac{1}{L_{\text{tot}}} - \frac{\delta_{k\kappa}}{L_\kappa}\right] \sum_{t=1}^{T} \alpha_{t-1}(i) \cdot \beta_t(j) \cdot b_j(o_t) \cdot e^{\lambda_{ij}^{(Tr)}} \tag{5.71}$$

Analog zu dem oben beschriebenen Vorgehen zur Ermittlung der Ableitung nach den Übergangsparametern können die Ableitungen nach den übrigen Parametern ermittelt werden ($\lambda^{(x)}$ steht für einen beliebigen Parameter):

$$\frac{\partial L}{\partial \lambda^{(x)}} = \frac{\partial L}{\partial \alpha_t(j)} \cdot \frac{\partial \alpha_t(j)}{\partial b_j(o_t)} \cdot \frac{\partial b_j(o_t)}{\partial \lambda^{(x)}} \tag{5.72}$$

Es gilt (analog zu oben):

$$\frac{\partial L_{\text{tot}}}{\partial \alpha_t(j)} = \beta_t(j) \qquad \frac{\partial L_\kappa}{\partial \alpha_t(j)} = \delta_{k\kappa}\beta_t(j) \tag{5.73a}$$

$$\frac{\partial \alpha_t(j)}{\partial b_j(o_t)} = \frac{\alpha_t(j)}{b_j(o_t)} \qquad \frac{\partial b_j(o_t)}{\partial \lambda_i^{(Occ)}} = b_j(o_t) \tag{5.73b}$$

$$\frac{\partial b_j(o_t)}{\partial \boldsymbol{\lambda}_i^{(M1)}} = b_j(o_t) \cdot \boldsymbol{o}_t \qquad \frac{\partial b_j(o_t)}{\partial \boldsymbol{\lambda}_i^{(M2)}} = b_j(o_t) \cdot \boldsymbol{o}_t^2 \tag{5.73c}$$

Unter Verwendung von Gleichung (5.64), (5.72) und (5.73) können die Ableitungen nach den Parametern $\lambda^{(x)}$ wie folgt ausgedrückt werden:

$$\frac{\partial E_{\text{MMI}}}{\partial \lambda_i^{(Occ)}} = \left[\frac{1}{L_{\text{tot}}} - \frac{\delta_{k\kappa}}{L_\kappa}\right] \alpha_t(i) \cdot \beta_t(i) \tag{5.74a}$$

$$\frac{\partial E_{\text{MMI}}}{\partial \lambda_l^{(M2)}} = \left[\frac{1}{L_{\text{tot}}} - \frac{\delta_{k\kappa}}{L_\kappa}\right] \alpha_t(i) \cdot \beta_t(i) \cdot o_t^l \tag{5.74b}$$

$$\frac{\partial E_{\text{MMI}}}{\partial \lambda_l^{(M2)}} = \left[\frac{1}{L_{\text{tot}}} - \frac{\delta_{k\kappa}}{L_\kappa}\right] \alpha_t(i) \cdot \beta_t(i) \cdot (o_t^l)^2 \tag{5.74c}$$

Damit sind nun alle Ableitungen, die für die Verbesserung der Parameter nötig sind, bestimmt.

Varianten des Trainings

Für das Update der Parameter beim Gradientenverfahren sind mehrere Methoden bekannt: Eine Variante besteht im direkten Update nach der Präsentation einer Sequenz nach Gleichung (5.57). Unter Umständen kann es aber günstiger sein, nicht nach jeder Sequenz ein Parameterupdate durchzuführen, sondern erst nachdem alle Trainingssätze verarbeitet wurden. Dann wird der Gradient über alle Sequenzen gemittelt (vgl. [Dro05]). Eine weitere Variante ist das sogenannte Resilient Propagation Verfahren (RProp), das zuerst von Riedmiller und Braun vorgestellt wurde [Rie93]. Hierbei wird für die aktuelle

Parameteränderung zusätzlich die letzte Änderung mit einbezogen. Es wird auch nicht mehr eine konstante Lernrate η verwendet, sondern diese wird je nach Gestalt des Gradienten angepasst. Für das Update der Parameter gilt:

$$\lambda_i^{(k+1)} = \lambda_i^{(k)} + \eta_i^{(k)} \operatorname{sgn}\left(\frac{\partial \mathcal{L}(\boldsymbol{\lambda}^{(k)})}{\partial \lambda_i}\right) \tag{5.75}$$

Die aktuelle Lernrate ist abhängig von den Vorzeichen des aktuellen Gradienten und des vorangegangenen Gradienten:

$$\eta_i^{(r+1)} = \begin{cases} 0.5 \cdot \eta_i^{(r)} & \text{falls} \quad \operatorname{sgn}\left(\nabla_i \mathcal{L}(\boldsymbol{\lambda}^{(k)}) \cdot \nabla_i \mathcal{L}(\boldsymbol{\lambda}^{(k-1)})\right) < 0 \\ 1.2 \cdot \eta_i^{(r)} & \text{falls} \quad \operatorname{sgn}\left(\nabla_i \mathcal{L}(\boldsymbol{\lambda}^{(k)}) \cdot \nabla_i \mathcal{L}(\boldsymbol{\lambda}^{(k-1)})\right) > 0 \end{cases} \tag{5.76}$$

Die Lernrate kann also adaptiv an die jeweilige Situation angepasst werden. Dadurch konvergiert das Trainingsverfahren oftmals schneller als der normale Gradientenabstieg.

Anmerkung zur Implementierung

Skalierung: Ähnlich wie bei HMM kommt es auch bei HCRF während der Berechnung der Produktionswahrscheinlichkeit sehr schnell zu einem Unterschreiten der Rechengenauigkeit, da eine Vielzahl von Zahlenwerten zwischen 0 und 1 multipliziert werden. Dadurch wird das Produkt so klein, dass ein Unterlauf stattfindet. Abhilfe verschafft in diesem Fall eine Skalierung, die nach jedem Rechenschritt durchgeführt wird. Wie zum Beispiel Levinson et al. [Lev83] erläutern, wird ein Skalierungskoeffizient

$$c_t = \left[\sum_{i=1}^{N} \alpha_t(i)\right]^{-1} \tag{5.77}$$

derart festgelegt, dass gilt:

$$\sum_{i=1}^{N} c_t \alpha_t(i) = 1, \qquad 1 \leq t \leq T \tag{5.78}$$

Es kann dann gezeigt werden, dass gilt:

$$\prod_{t=1}^{T} c_t = \frac{1}{P} \tag{5.79}$$

Aufgrund von Zahlenbereichsüberschreitungen kann dieses Produkt nicht berechnet werden, sondern nur der logarithmierte Wert davon:

$$\log P = -\sum_{t=1}^{T} \log c_t \tag{5.80}$$

Dieser Wert ist allerdings ausreichend zur Bestimmung einer Klassenzugehörigkeit, da der Logarithmus monoton ist.

Addition logarithmierter Größen: Eine Alternative oder komplementäre Methode, die Wahrscheinlichkeiten unter Einhaltung der Rechengenauigkeit zu berechnen, besteht in der Logarithmierung. Bei der Berechnung der benötigten Größen während des Trainings oder der Dekodierung benötigt man dann jedoch häufig den Logarithmus einer Summe von Werten, die nur logarithmiert vorliegen. Aufgrund der Zahlenbereichsbeschränkung des Computers ist ein direktes Berechnen (Delogarithmieren - Addieren - Logarithmieren) unmöglich, so dass eine Möglichkeit gefunden werden muss, $\log(a+b)$ als Funktion von $\log a$ und $\log b$ darzustellen. Dies leistet die Kingsbury-Rayner-Formel [Kin71]:

$$\begin{aligned}\log(a+b) &= \log\left[a\left(1+\frac{b}{a}\right)\right] = \\ &= \log a + \log\left(1+\frac{b}{a}\right) = \\ &= \log a + \log\left(1+e^{\log b - \log a}\right)\end{aligned} \tag{5.81}$$

5.2.6 Automatische Segmentierung mit dem Viterbi-Algorithmus

Aufgrund der Verwandtschaft in Bezug auf die Struktur von HCRF und HMM ist es auch bei HCRF möglich, den Viterbi-Algorithmus zur automatischen Segmentierung von Merkmalsströmen zu verwenden (siehe Kapitel 5.1.4). Analog zu HMM muss für jede zu unterscheidende Klasse ein separates Modell erstellt und trainiert werden. Durch Verkettung aller Modelle zu einem großen Super-Modell kann mit dem Viterbi-Algorithmus die optimale Abfolge von Klassen, gefunden werden. Im Unterschied zu HMM sind die erhaltenen Produktionswahrscheinlichkeiten von HCRF eher als ein Maß für die Fähigkeit eines Modells, eine Sequenz zu erzeugen zu interpretieren, als als echte Wahrscheinlichkeit. Insbesondere können aufgrund der fehlenden Normierung Werte kleiner null und größer eins auftreten. Da für eine Zuordnung einer optimalen Klassenfolge allerdings nur relative Vergleiche benötigt werden, ist dieses Vorgehen gerechtfertigt.

5.3 Experimente und Ergebnisse

Im Folgenden werden die Experimente und die Ergebnisse beschrieben, die mit den dynamischen Klassifikatoren HMM und HCRF durchgeführt worden sind. Dabei wird unterschieden, auf welchen Daten die jeweiligen Modelle trainiert wurden, und welches Klassifikationsschema verwendet wurde.

5.3.1 HMM mit Transkriptionen

Eine Möglichkeit der Erkennung von Gruppenaktionen besteht in der Auswertung der Wörter, die die Meeting-Teilnehmer gesprochen haben. Eine automatische Spracherkennung versucht, die in einer Besprechung gesprochenen Äußerungen möglichst genau in einen schriftlicher Form zu konvertieren. Dazu müssen für gewöhnlich Modelle trainiert werden, die eine exakte Transkription der gesprochenen Sprache benötigen. Diese Transkriptionen werden manuell erzeugt. Da die Erstellung der Transkriptionen personell sehr

aufwendig ist, beschränkte man sich auf die Erstellung derselben von ausgewählten Besprechungen mit insgesamt 99 Instanzen von Gruppenaktionen. Man kann sie auch als idealisierte Erkennungsresultate eines Spracherkenners auffassen. Basierend auf den Transkriptionen können in Anlehnung an Verfahren der Topic-Segmentierung und -Klassifikation (vgl. [Sch97b],[Ima97], [Wal99]) Statistiken über die Häufigkeiten der vorkommenden Wörter erstellt werden. Mit einer geeigneten Auswertung und einem entsprechenden statistischen Modell, können dann Bereiche identifiziert werden, die der gleichen Gruppenaktion angehören. Im Allgemeinen wird für die Klassifikation ein 1-State-HMM verwendet. Wird ein so gestaltetes HMM mit den Statistiken erstellt, ergeben sich die nachfolgenden Ergebnisse. Werden alle möglichen zehn Klassen unterschieden (siehe Abschnitt 2.1.3), so liegt die Erkennungsrate unbekannter Segmente bei 44 %. Die Konfusionsmatrix ist in Tabelle 5.1 gezeigt.

	Discussion	Monologue1	Monologue2	Monologue3	Monologue4	Note-taking	Presentation	Whiteboard	Consensus	Disagreement
Discussion	**18**	0	0	0	0	0	0	0	6	1
Monologue1	1	**1**	0	0	0	0	3	0	0	0
Monologue2	0	3	**0**	1	0	0	1	3	0	0
Monologue3	1	3	1	**0**	3	0	1	1	0	0
Monologue4	0	4	0	1	**0**	0	2	0	0	0
Note-taking	0	0	0	0	0	**2**	0	0	0	0
Presentation	2	4	2	1	2	0	**2**	0	0	0
Whiteboard	3	0	1	1	0	0	2	**3**	1	0
Consensus	0	0	0	0	0	0	0	0	**10**	0
Disagreement	0	0	0	0	0	0	0	0	0	**8**

Tabelle 5.1: *Konfusionsmatrix der Erkennung von Segmenten von Besprechungen mit einem 1-State-HMM. Merkmale: manuell erzeugte Transkriptionen, Lexikon L8 + Consensus + Disagreement*

Darin ist ersichtlich, dass eine Unterscheidung der Monologe auf diese Art und Weise nicht möglich ist. Dies ist dadurch zu erklären, dass in den Transkriptionen keine Hinweise auf den Sprecher enthalten sind. Fasst man also die Monologe zusammen, und verringert die Anzahl der Klassen damit auf sieben, verbessert sich die Erkennungsrate auf 61 %.
Beachtenswert ist bei diesem Verfahren, dass die Unterscheidung von *Consensus* und *Disagreement* zuverlässig funktioniert. Dies ist bei anderen in dieser Arbeit vorgestellten Methoden nicht der Fall. Deshalb könnte man mit Hilfe einer Fusion von mehreren Algorithmen eventuell diese Unterscheidung zusätzlich treffen. Allerdings bleibt das Problem der Verwechslung mit der Aktion *Discussion* nach wie vor bestehen. Fasst man die Aktionen *Consensus* und *Disagreement* als *Discussion* auf, so verbessert sich das Erkennungsergebnis nochmals auf insgesamt 68 %.

5.3.2 HMM mit MFCC

Die Verwendung von Mel-Frequenz-Cepstral-Koeffizienten (MFCC) hat in der Spracherkennung große Verbreitung gefunden. Sie sind zu einem Standard geworden, um gesprochene Äußerungen in Textform zu bringen. Meist werden sie dann mit HMM als Klassifikator verarbeitet. In dieser Arbeit werden MFCC zur Klassifikation von Gruppenaktionen verwendet. Die Idee dabei ist, dass die Modelle nicht nur eine – wenn auch primitive – Art von Spracherkennung betreiben, sondern darüber hinaus eine semantisch höhere Bedeutung der Merkmale unterscheiden. Man kann diesen Ansatz also in gewissem Maße als Erweite-

	Discussion	Monologue1	Monologue2	Monologue3	Monologue4	Note-taking	Presentation	Whiteboard
Discussion	**49**	1	1	0	0	0	0	0
Monologue1	3	**6**	0	0	0	0	0	0
Monologue2	0	0	**4**	1	0	0	0	5
Monologue3	2	0	0	**10**	0	0	0	0
Monologue4	2	0	0	0	**6**	0	0	0
Note-taking	2	0	0	0	0	**1**	0	0
Presentation	3	4	0	0	0	0	**4**	3
Whiteboard	0	0	0	0	0	0	0	**16**

Tabelle 5.2: *Konfusionsmatrix der Erkennung von Segmenten von Besprechungen mit einem HMM mit* 6 *Zuständen, Merkmale: MFCC, Lexikon L8*

rung der Erkennung basierend auf den manuell erstellten Transkriptionen sehen. Werden wieder zehn Klassen unterschieden, und wird ein HMM mit sechs Zuständen verwendet, so liegt die Erkennungsrate bei 68 %. Allerdings ist eine zuverlässige Trennung der Klassen *Discussion*, *Consensus* und *Disagreement* nicht möglich. Aus diesem Grund werden die Abschnitte der beiden letztgenannten Klassen als *Discussion* gewertet. Damit ergibt sich die eine Konfusionsmatrix, wie sie in Tabelle 5.2 gezeigt ist. Die Erkennungsrate liegt dann bei 71 %.

Um bei der Erzeugung der MFCC die Quasi-Stationarität zu erhalten, muss der Datenstrom gefenstert werden (siehe Abschnitt 2.3.1). Je nach Größe des Fensters ergibt sich eine längere oder kürzere Merkmalssequenz. Untersucht wurden in dieser Arbeit Fenster mit einer Länge von 10ms. In den Ergebnissen in Tabelle 5.3 ist deutlich zu erkennen, dass eine größere Anzahl von Zuständen nicht mit einer höheren Erkennungsrate korrespondiert. So ergibt sich die beste Leistung mit einem HMM mit sechs Zuständen. Werden mehr Zustände verwendet, verringert sich die Rate deutlich. Dies ist ein deutlicher Hinweis darauf, dass die Anzahl der zur Verfügung stehenden Trainingsdaten nicht ausreichend ist, um größere Modelle zu trainieren.

Die bisherigen Ergebnisse stellen nur die reine Erkennung von einzelnen Segmenten dar. Für eine automatische Analyse einer Besprechung ist auch eine Segmentierung notwen-

Fensterlänge	**Anzahl der Zustände**	**Erkennungsrate** [%]
10 ms	6	68,0
10 ms	32	54,1
10 ms	128	47,5

Tabelle 5.3: *Erkennungsergebnisse bei einer Fensterung des Datenstroms von* 10 ms *mit unterschiedlich großen HMM-Modellen, Merkmale: MFCC, Lexikon L8*

dig. Hier wird versucht, eine Untergliederung mit Hilfe des BIC durchzuführen (vgl. dazu Abschnitt 3.3.1, zu den Fehlermaßen siehe Kapitel 2.4). Damit kann die Klassifikationsleistung von HMM unabhängig von der Segmentierung bewertet werden. In Tabelle 5.4

M1	**M2**			**M3**			
E_f [%]	E_{sb} [%]	E_{sc} [%]	E [%]	INS [%]	DEL [%]	ACC [s]	E_c [%]
38,4	83,3	62,7	73,0	8,4	44,9	19,6	53,6

Tabelle 5.4: *Segmentierung mit BIC und Klassifikation mit HMM, Merkmale: MFCC, Lexikon L8*

ist zu erkennen, dass die Löschungsrate recht hoch ist. Ebenso weicht die Genauigkeit der gefundenen Grenzen mit durchschnittlich 19.6 Sekunden unakzeptabel von den echten Grenzen ab. Die Erkennungsrate mit 46.4 % ist zudem nicht besonders gut. Die Akkuratheit beträgt 49.2 %.

5.3.3 HMM mit Sprecheraktivität

Eine reduzierte Variante von Merkmalen besteht darin, nur die Aktivität eines Sprechers zu verwenden, nicht den Inhalt, oder andere informationstragende Einheiten. Dadurch wird zwar viel an Information verworfen, allerdings kann man auf diese Weise einfach und schnell grundlegende Tests durchführen. Das Profil der Sprecheraktivität besteht nur aus den Einträgen 1, falls die jeweilige Person aktiv ist, oder 0 falls sie ruhig ist. Daher spricht man auch von einem *Binariy Speech Profile* (BSP). Die Ergebnisse der Versuche mit BSP und HMM sind in Tabelle 5.5 und 5.6 gezeigt[4].

In Tabelle 5.6 werden wieder die Klassen *Consensus* und *Disagreement* als *Discussion* gewertet. Bei allen Modellen wird nur eine Gaussverteilung zur Modellierung der Parameter eingesetzt. Versuche mit mehreren Mixturen ergeben zum Teil deutliche Verschlechterungen in der Erkennungsleistung, was sicherlich darauf zurückzuführen ist, dass nicht genügend Trainingsmaterial vorhanden ist, um komplexere Modelle trainieren zu können.

[4]Die teilweise hohe Anzahl an Zuständen wurde gewählt, um der Gefahr der Übersegmentierung zu begegnen. Dies gelingt im vorliegenden Fall nur eingeschränkt, da nicht genügend Trainingsmaterial für das Training solch großer Modelle vorhanden ist.

Zustände	Akkuratheit [%]
60	47,5
70	50,8
75	54,1

Tabelle 5.5: *Akkuratheit der Segmentierung mit HMM mit einer Mixtur, Merkmale: Binary Speech Profile, Lexikon L8 + Consensus + Disagreement*

Zustände	Akkuratheit [%]
2	66,1
3	72,5
6	74,3
8	68,8
75	63,3

Tabelle 5.6: *Akkuratheit der Segmentierung mit HMM mit einer Mixtur, Merkmale: Binary Speech Profile, Lexikon L8*

5.3.4 HMM mit semantischen Merkmalen

Zustände	Mixturen	Akk. [%]
2	1	79,8
2	2	85,3
2	3	91,7
3	1	88,1
3	2	91,7
3	3	89,9
4	1	87,2
4	2	89,9
4	3	94,5

Zustände	Mixturen	Akk. [%]
5	1	87,2
5	2	90,8
5	3	87,2
6	1	89,0
6	2	89,0
6	3	87,2
7	1	89,9
7	2	88,1
7	3	95,4

Tabelle 5.7: *Akkuratheit der Segmentierung unterschiedlicher HMM-Konfigurationen, Merkmale: SEM-A, Lexikon L8*

Ebenso wie bei den statischen Klassifikatoren (siehe Kapitel 3.4.2) wurden auch die HMM mit den semantischen Merkmalen (siehe 2.3.3) getestet. Ausgewählte Ergebnisse sind in Tabelle 5.7 gezeigt. Wie zu erwarten war, ist die Akkuratheit, die mit nur 2 Zuständen erreicht werden kann, bereits zufriedenstellend hoch. Ein fast perfektes Ergebnis kann mit 7 Zuständen und 3 Gaußverteilungen erreicht werden. Allerdings bleibt anzumerken, dass diese Resultate nur beschränkt aussagekräftig sind, da die Merkmale teilweise manuell unterstützt erstellt wurden. Deshalb ist eine Untersuchung der Leistungsfähigkeit von HMM auf komplett automatisch erstellten Daten, wie zum Beispiel den Low-Level-Merkmalen

unumgänglich.

5.3.5 HMM mit Low-Level Merkmalen

Die vorangegangenen Experimente haben gezeigt, dass eine Gliederung einer Besprechung in Gruppenaktionen mit HMM gut möglich ist, wenn geeignete Merkmale vorliegen. Waren diese in den oben beschriebenen Versuchen entweder manuell unterstützt erstellt oder stark vereinfacht, so werden hier multimodale Merkmale verwendet, die sowohl aus dem Audiokanal als auch aus dem Videokanal automatisch extrahiert wurden (siehe Abschnitt 2.3.1). Eine Übersicht über die verwendeten Merkmale gibt Tabelle 2.5 . Zunächst werden wie vorhin auch die bekannten acht Klassen unterschieden. Ausgewählte Ergebnisse sind in Tabelle 5.8 gezeigt. Auch hier wurden die Modelle nur mit einer Gaußverteilung modelliert, da die Größe der Trainingsdaten sehr beschränkt ist. Das HMM mit nur einem Zustand entspricht faktisch einem Gauß-Mixtur-Modell (vgl. Kapitel 3.1.2). Das beste Ergebnis wird in diesem Fall mit einem HMM mit 3 Zuständen erreicht. Grundsätzlich ist anzumerken, dass die Anzahl der Zustände eines HMM, anders als zum Beispiel bei der Phonemerkennung mit Anlaut und Auslaut etc., keine Entsprechung im Aufbau der Merkmalssequenzen hat. Daher gibt es auch keine bevorzugte Anzahl an Zuständen.

Zustände	**Akkuratheit** [%]
1	84,2
2	83,5
3	**92,1**
4	89,9
5	83,5
6	84,9
7	91,4
8	86,3

Tabelle 5.8: *Akkuratheit der Segmentierung unterschiedlicher HMM-Konfigurationen mit einer Gaußverteilung, Merkmale LOW, Lexikon L8*

Zusätzlich zu den Tests mit acht Klassen wurden Experimente mit dem erweiterten Lexikon *L14* (siehe Kapitel 2.1.3) durchgeführt. Die Anzahl der zu unterscheidenden Klassen beträgt hierbei 14. Die Ergebnisse sind in Tabelle 5.9 gezeigt. Das beste Ergebnis erreichen hier HMM mit vier Zuständen und zwei Gaußverteilungen je Zustand. Eine Verwendung von mehr Mixturen bringt keine weitere Verbesserung der Erkennungsleistung, da mit den ohnehin wenigen Daten noch zusätzliche Parameter trainiert werden müssten.

5.3.6 Experimente mit HCRF

HCRF können, wie in Kapitel 5.2.3 beschrieben, als eine Erweiterung von HMM aufgefasst werden. In diesem Abschnitt werden HCRF mit HMM mit dem gleichen Aufbau

Zustände	Mixturen	Akk. [%]
1	1	74,8
2	1	67,6
3	1	67,6
4	1	69,1
5	1	69,1
6	1	71,9
7	1	64,8
8	1	67,6

Zustände	Mixturen	Akk. [%]
1	2	56,8
2	2	69,8
3	2	68,4
4	2	**75,5**
5	2	61,9
6	2	51,8
7	2	56,1
8	2	59,0

Tabelle 5.9: *Akkuratheit der Segmentierung unterschiedlicher HMM-Konfigurationen, Merkmale LOW, Lexikon L14*

verglichen. Da das gradientenbasierte Training der HCRF leicht in einem lokalen Minimum enden kann, werden die HCRF zunächst mit den Parametern von HMM mit der gleichen Struktur initialisiert. Anschließend kann mit dem vorgestellten Gradiententraining das Modell weiter an die Trainingsdaten angepasst werden. Auf Basis der Low-Level Merkmale werden extensive Tests mit einer Vielzahl von Konfigurationen durchgeführt (siehe auch [Rei07]).

	Mittelwert [%]	Standardabweichung [%]
semantische Merkmale	0,1	0,8
Low-Level Merkmale	3,3	2,2

Tabelle 5.10: *Übersicht der besseren Performanz bezüglich der Akkuratheit von HCRF gegenüber HMM, Merkmale: SEM-A und LOW, Lexikon L8*

Verwendung semantischer Merkmale

Bei der Verwendung semantischer Merkmale kann keine explizite Verbesserung oder Verschlechterung der Erkennungsleistung beobachtet werden. Bei 80 verschiedenen Konfigurationen ist der Mittelwert der Abweichung 0,1 %. Werden die HCRF mit den Parametern von gut trainierten[5] HMM initialisiert, so wird die gleiche Akkuratheit erreicht. Der Einsatz von HCRF lohnt sich daher bei semantischen Merkmalen nicht.

Verwendung von Low-Level Merkmalen

Im Folgenden werden Low-Level Merkmale verwendet. Werden HCRF nur mit den Parametern von HMM initialisiert, und Testdurchläufe gestartet, ohne dass sie noch weiter trainiert werden, so ergibt sich im Durchschnitt eine Verbesserung der Akkuratheit von

[5]Die für die Initialisierung verwendeten HMM entsprechen denen, die zur Segmentierung in Kapitel 5.3.4 eingesetzt werden.

3,3 %. Die Standardabweichung liegt bei 2,2 %. Im besten Fall ergibt sich ein Zuwachs der Akkuratheit von 10,1 %. Wird zur Unterteilung von Meetings das Lexikon L14 zugrunde gelegt (vgl. Tabelle 2.3), so ergibt sich ein ähnliches Bild. Im Mittel kann eine Verbesserung der Akkuratheit von 0,2 % erziehlt werden. Das Maximum liegt bei 2,2 %, die Standardabweichung bei 0,6 %. Schlussfolgernd ist festzuhalten, dass HCRF aufgrund ihrer Fähigkeit Abhängigkeiten zu modellieren für die Segmentierung von Besprechungen geeignet erscheinen.
Werden HCRF nach der Initialisierung weiter trainiert, so hängt das Ergebnis stark von der Art der Merkmale ab. Je genauer die Klassengrenzen festgelegt sind, desto besser funktioniert das Training. Wird ein integriertes Training (ähnlich wie bei HMM in der Sprachverarbeitung) gewählt, so kann es passieren, dass durch ungenau festgelegte Klassengrenzen das Modell sogar verschlechtert wird. Dass das Trainingsverfahren funktioniert, konnte verifiziert werden, indem HCRF Modelle mit schwach trainierten[6] HMM initialisiert wurden. Bei HMM, die die Merkmalssequenzen sehr gut nachbilden können, ist jedoch so gut wie keine Verbesserung erreichbar.

5.3.7 Weitere dynamische Klassifikatoren

Erst in jüngster Zeit sind andere Arbeiten entstanden, die zwar prinzipiell mit dem gleichen Meeting-Material arbeiten, aber nur schwer vergleichbar sind, da meist anders geartete Merkmale verwendet werden, oder leicht abweichende Lexika benutzt werden, sodass eine objektive Vergleichbarkeit nicht gegeben ist. Unter anderem werden Asynchrone Hidden-Markov-Modelle zur robusten Erkennung von Gruppenaktionen bei gestörtem Videobild verwendet [AH07b]. Aber auch neuartige Verfahren wie Dynamische Graphische Modelle werden aufgrund ihrer universellen Einsatzfähigkeit im Bereich der Gruppenaktionserkennung eingesetzt [AH05b], [Die07].

5.4 Kapitelzusammenfassung

Die Verwendung von dynamischen Klassifikationsverfahren mit Viterbi-Dekodierung erzielte im Vergleich zu statischen Verfahren einen entscheidenden Fortschritt im Hinblick auf eine zuverlässige Segmentierung von Besprechungen in Gruppenaktionen. Mit HMM konnten sehr hohe Akkuratheiten von über 92 % erzielt werden. Durch den Einsatz von Hidden Conditional Random Fields (HCRF), einer Erweiterung von Conditional Random Fields um verborgene Zustände, die auch als Verallgemeinerung von HMM angesehen werden können, konnten diese Ergebnisse in einigen Konfigurationen noch übertroffen werden. So lange jedoch keine ausgereiften Methoden zum Training für eine allgemeine Form von HCRF, und damit für eine weitgehend beliebige Modellierung von Abhängigkeiten entwickelt sind, bleibt abzuwägen, ob sich der etwas erhöhte Aufwand zur Bestimmung der Parameter beim Einsatz von HCRF lohnt.

[6] Bei diesen HMM wurde das Training nach wenigen Iterationen abgebrochen.

Kapitel 6

Segmentierung mit hybriden Klassifikationsverfahren

Dieses Kapitel beschreibt erweiterte Verfahren zur Segmentierung von Besprechungen. Im Besonderen wird auf zweistufige und hybride Methoden ebenso eingegangen, wie auf fortgeschrittene Abwandlungen bekannter Klassifikatoren.

6.1 Tied-Posterior Ansatz

Hidden Markov-Modelle mit gaußverteilten Ausgabewahrscheinlichkeiten haben sich in der Spracherkennung und auch in der Anwendung zur Segmentierung von Besprechungen bewährt (vgl. Abschnitt 5.3). Bei den bisher verwendeten Verfahren wird für jede Klasse die Beobachtungswahrscheinlichkeit separat mit Hilfe von Gauß-Mixtur-Modellen modelliert. Der Einfluss von Kontextinformationen bleibt dabei allerdings unberücksichtigt. Um Abhängigkeiten der Modelle modellieren zu können, können die Ausgabewahrscheinlichkeiten mit einem diskriminativen Ansatz, wie zum Beispiel Neuronalen Netzen, geschätzt werden. In der Spracherkennung wird dies mit gutem Erfolg durchgeführt (vgl. [Bou94], [Rot00], [Sta03], [Sta06]). Ein Neuronales Netz schätzt hierbei die posteriori Symbolwahrscheinlichkeiten gemeinsam für alle Hidden-Markov-Modelle. Aus diesem Grund spricht man auch von einem Tied-Posterior Ansatz.
Das Neuronale Netz ermittelt die a-posteriori-Wahrscheinlichkeit $P(\kappa_j|\boldsymbol{x}_t)$ für das Auftreten der Klasse κ_j gegeben einen Eingangsvektor $\boldsymbol{x}_t$. Analog zur Gauß-Modellierung der Beobachtungswahrscheinlichkeiten geschieht hier die Verknüpfung der einzelnen Ausgänge des Neuronalen Netzes mit den HMM-Zuständen über zustandsabhängige Gewichtsfaktoren. Durch Anwendung des Satzes von Bayes ergibt sich folgender Ausdruck für die Ausgabedichten der HMM [Sta06]:

$$b_i(\boldsymbol{x}_t) = p(\boldsymbol{x}_t|s_i) \propto \sum_{k=1}^{K} c_{ik} \frac{P(\kappa_i|\boldsymbol{x}_t)}{P(\kappa_i)} \tag{6.1}$$

Dabei gilt für die Summe der Gewichtsfaktoren $\sum_{k=1}^{K} c_{ik} = 1$. In Abbildung 6.1 ist veranschaulicht, wie ein solches hybrides HMM aussieht. Die Gewichtsfaktoren c_{ij} und Übergangswahrscheinlichkeiten der Modelle werden, analog zu Abschnitt 5.1.3, mit dem

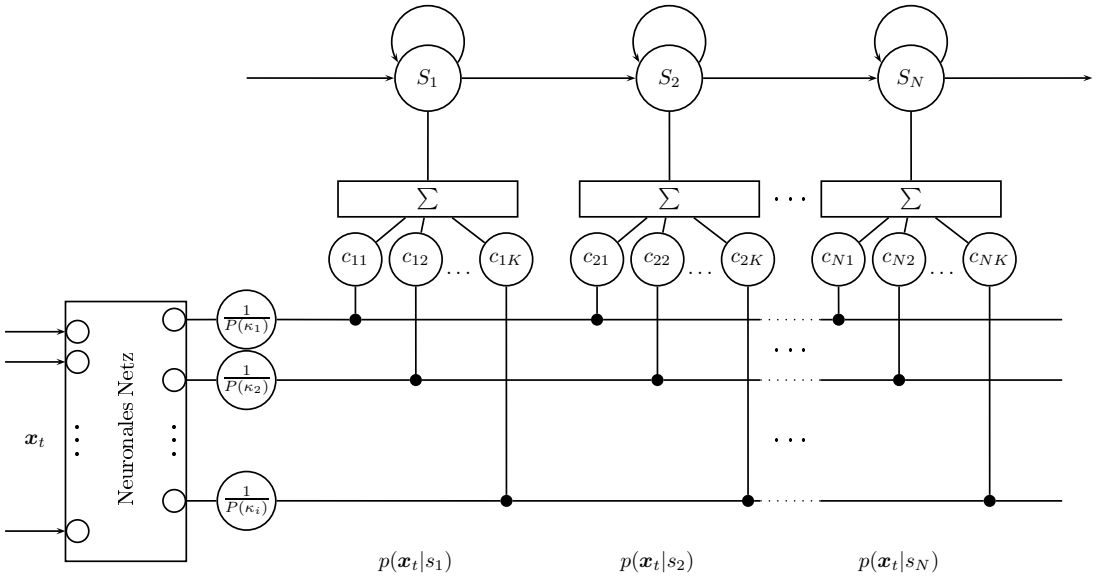


Abbildung 6.1: *Aufbau eines hybriden HMM nach dem Tied-Posterior Ansatz mit einem Neuronalen Netz als Wahrscheinlichkeitsschätzer*

EM-Algorithmus trainiert. Durch diesen Ansatz ist es möglich, die Vorteile eines diskriminativen Klassifikators mit der vollen Flexibilität von HMM zu kombinieren.

6.2 Tandem-Verfahren

Eine alternative Möglichkeit, die diskriminativen Eigenschaften von Neuronalen Netzen auszuschöpfen und dabei gleichzeitig die Vielseitigkeit von HMM zu erhalten, besteht in der Modellierung der aus dem KNN gewonnenen Beobachtungswahrscheinlichkeiten mit Gauß-Mixtur-Modellen. In der Literatur wird dieser Ansatz als *Tandem*-Modell [Her00] bezeichnet. Rigoll et. al [Rig98b] schlagen vor, die diskriminativen Fähigkeiten von Neuronalen Netzen in der Spracherkennung auszunutzen. Man fasst dabei das Neuronale Netz als Teil der Merkmalsextraktion auf. Der diskriminativ trainierte Ausgangsvektor des NN ist dann der neue Merkmalsvektor für das nachfolgende Gauß-HMM-System. Als zugrunde liegendes KNN können die in den Kapiteln 3.1.3 und 4.1 vorgestellten Modelle verwendet werden. Der grundlegende Ablauf dieses Vorgehens ist in Abbildung 6.2 gezeigt.
In dieser Arbeit wird zusätzlich erstmals ein verändertes Schema angewendet (siehe auch [Rei06a]). Anstelle des üblichen MLP wird ein LSTM-RNN in der ersten Stufe eingesetzt. Die Ergebnisse werden sodann mit Hilfe eines symmetrischen gleitenden Mittelwertfilters (Simple-Moving-Average, SMA) [Smi97] geglättet. Dabei wird ein gleitendes Fenster über den erhaltenen Signalverlauf verschoben und über eine ungerade Anzahl B von Werten (der Filterbreite) des Eingangssignals x symmetrisch um jeden Wert x_i gemittelt. Dadurch ergibt sich der gemittelte Wert $\overline{x_i}$:

$$\overline{x_i} = \frac{1}{B} \sum_{t=t-\frac{B-1}{2}}^{t+\frac{B-1}{2}} x_t, \quad \text{mit } t = \frac{B-1}{2}, \ldots, N - \frac{B-1}{2} \tag{6.2}$$

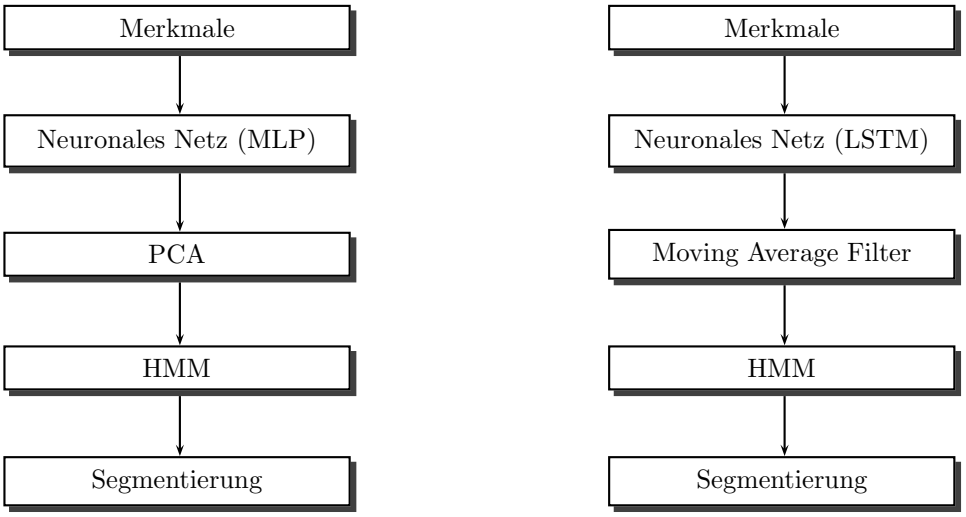


Abbildung 6.2: *Überblick über die einzelnen Schritte beim Tandem-Verfahren (links) und dem abgewandelte Schema mit Filterung (rechts)*

Für die Werte an den Randbereichen, für die nicht die volle Filterbreite zur Verfügung steht, wird mit zunehmender Breite gefiltert (siehe hierzu auch [Sch06]).
Da das LSTM-Netz jeden Frame isoliert klassifiziert, können abrupte Sprünge in den Klassifikationsergebnissen auftreten. Um die Auswirkungen dieser Sprünge zu vermindern wird ein gleitender Mittelwertfilter eingesetzt, der einen gleichmäßigeren Merkmalsverlauf generieren kann. Die auf diese Weise geglätteten Erkennungsergebnisse werden dann einem nachfolgenden HMM zugeführt, das die finale Segmentierung und Erkennung der Gruppenaktionen durchführt.

6.3 Zweischichtige Ansätze

In diesem Abschnitt wird die Segmentierung und Erkennung von Gruppenaktionen in zwei unterschiedliche Schichten aufgeteilt. Jede Schicht erkennt unterschiedliche Ereignisse mit einer unterschiedlichen zeitlichen Auflösung. Auf diese Weise kann die Erkennung der semantischen high-level Gruppenaktionen in eine Erkennung einer Reihe von Sub-Aktionen zerlegt werden. Anders als bei den Verfahren zuvor, werden in den beiden Schichten unterschiedliche Klassen unterschieden. Die untere Schicht beschreibt individuelle Aktionen der Teilnehmer, während die darüberliegende Schicht die Gruppenaktionen modelliert. Man verspricht sich dadurch eine zuverlässigere Klassifikation, eine bessere Ausnutzung des Trainingsmaterials und dadurch eine höhere Robustheit der Segmentierung.
Mehrschichtige Ansätze zur Modellierung komplexer Vorgänge werden bereits in unterschiedlichen Bereichen eingesetzt. Ivanov und Bobick [Iva00] verwenden Sub-HMMs zur Erkennung individueller Handbewegungen und High-Level-HMM, um Sequenzen dieser Bewegungen als semantisch höherwertige Aktion zu erkennen. Für ein automatisches Selektieren der interessantesten Kamera in Meetings verwendet Al-Hames zwei Schichten [AH07a]. Die erste Schicht bestimmt Aktionen der Teilnehmer, während die zweite Schicht die eigentliche Kameraselektion durchführt. Hierarchische HMM (HHMM) [Fin98]

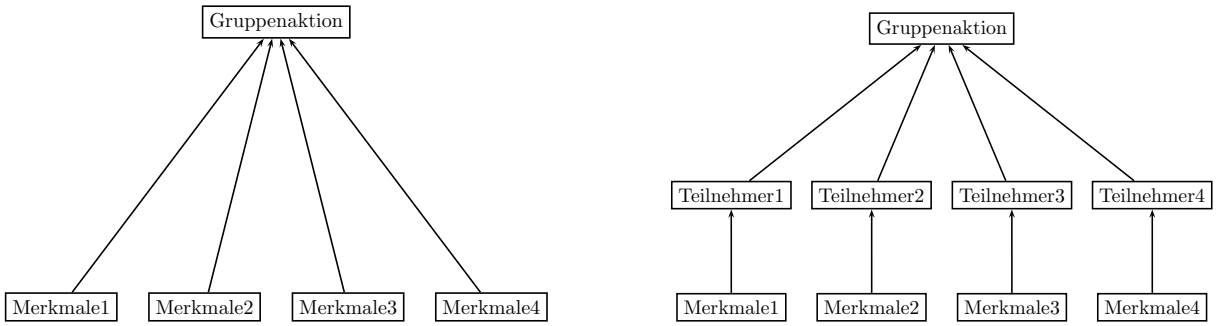


Abbildung 6.3: *Schematischer Vergleich des einstufigen (links) und zweistufigen (rechts) Ansatzes*

werden zum Beispiel verwendet, um DNA-Sequenzen zu beschreiben [Hu00]. HHMM werden auch zur Gewinnung von Information bei der Verarbeitung natürlicher Sprache eingesetzt [Sko03]. Probleme der HHMM bestehen in der großen Anzahl der Parameter, die bestimmt werden müssen, und in der Gefahr des Overfittings, wenn zu wenig Material zum Training zur Verfügung steht.

6.3.1 Vorüberlegungen

Im Bereich der Analyse von Besprechungen liegt es nahe, Gruppenaktionen als eine Kombination von Einzelaktionen bzw. Teilnehmerzuständen der Teilnehmer zu modellieren. Eine Gruppenaktion besteht also aus einer Abfolge verschiedener Einzelaktionen oder Teilnehmerzuständen der an der Besprechung teilnehmenden Personen. In einem Vorabtest wurden fünf Zustände festgelegt, in denen sich Teilnehmer befinden können. Diese sind in Tabelle 6.1 aufgelistet[1].

Zustand	**Beschreibung**
sitting-silent	ein Teilnehmer sitzt auf seinem Platz und spricht nicht
sitting-silent-writing	ein Teilnehmer sitzt auf seinem Platz, spricht nicht und macht Notizen
sitting-talking	ein Teilnehmer sitzt auf seinem Platz und spricht
standing-talking	ein Teilnehmer steht und spricht
standing-talking-writing	ein Teilnehmer steht, spricht und schreibt (z.B. auf das Whiteboard)

Tabelle 6.1: *Beschreibung der Teilnehmerzustände für die Vorüberlegungen*

Um zu testen, ob diese Vorgehensweise generell sinnvoll ist, werden Tests mit den manuell erstellten Annotationen durchgeführt. Aus den definierten Abschnitten wird für jeden Teilnehmer ein Zustand festgelegt. Dieser kann ähnlich wie in Kapitel 3 mit statischen Klassifikatoren modelliert werden. Darauf aufbauend kommt ebenfalls ein statischer

[1]Ein sechster Zustand *standing-silent-writing*, der die Liste der Zustände vervollständigt, tritt in den verwendeten Daten der M4-Datenbank nicht auf.

Klassifikator zum Einsatz. Werden für beide Stufen SVM verwendet, so konnte bei der reinen Erkennung der Gruppenaktionen eine Erkennungsrate von 100 % erzielt werden. Allerdings bleibt anzumerken, dass die Daten manuell aufbereitet sind und deshalb diese Erkennungsrate nicht aussagekräftig ist.

Aktion	**Beschreibung**
speaking	ein Teilnehmer spricht
writing	ein Teilnehmer macht Notizen
idle	ein Teilnehmer spricht nicht und schreibt nicht

Tabelle 6.2: *Beschreibung der individuellen Aktionen*

Für eine echte automatische Erkennung und Segmentierung in einem zweistufigen Ansatz wird die Anzahl der individuellen Aktionen auf drei reduziert. Diese sind in Tabelle 6.2 aufgeführt. Die Anzahl der Instanzen der individuellen Aktionen ist ausreichend, um diese durch HMM in der unteren Ebene zu modellieren (vgl. Tabelle 6.3). Da diese individuellen Aktionen nur lokal für einen Teilnehmer gelten, müssen für die Weiterverarbeitung die Ergebnisse mit relevanten gruppenbezogenen Informationen erweitert werden.

Aktion	**Training**	**Test**
idle	1423	1485
speaking	1057	1022
writing	351	476
gesamt	2831	2983

Tabelle 6.3: *Anzahl der individuellen Aktionen im Trainings- und Testdatensatz*

Für die Bestimmung der individuellen Aktionen werden aus den vorhandenen Merkmalen nur diejenigen verwendet, die personenspezifisch für einen Teilnehmer sind (siehe Tabelle 2.5). Nach der Erkennung der individuellen Aktionen, werden die Ergebnisse mit den Gruppen-Merkmalen angereichert und der zweiten Schicht zur Segmentierung und Klassifikation der Gruppenaktionen weitergegeben.
Zur einfacheren mathematischen Beschreibung seien die individuellen Aktionen zur Menge $\Phi = \{\varphi_1, \ldots, \varphi_K\}$ zusammengefasst. K beschreibt dabei die Anzahl der zu unterscheidenden Klassen, hier also drei.

6.3.2 Layered HMM

Ein Lösungsansatz für ein zweistufiges System basierend auf HMM wird von Oliver et al. [Oli02b] [Oli04] mit den Layered HMM vorgestellt. Diese Art der HMM benötigt weniger Trainingsdaten, da die untere Schicht separat trainiert werden kann. Ebenso bietet dieser Ansatz den Vorteil, dass die HMM in der unteren Schicht unabhängig von der darüberliegenden Schicht verändert werden können, ohne dass am darüberliegenden System

etwas geändert werden muss. Einen Überblick über das System bietet Abbildung 6.4.

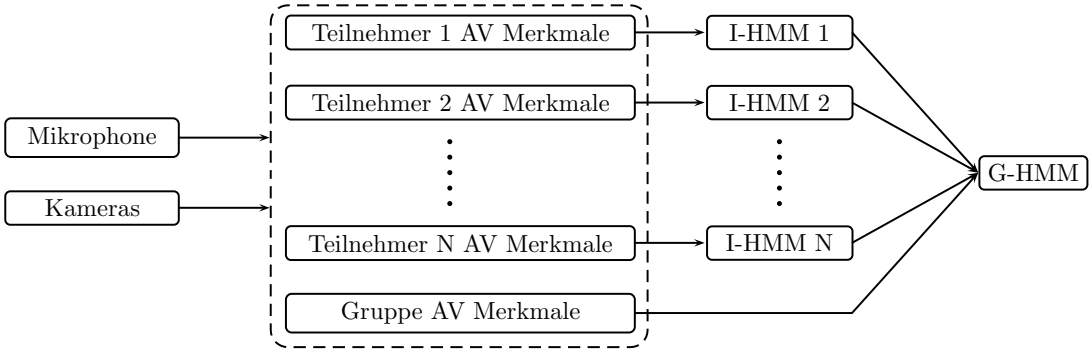


Abbildung 6.4: *Überblick des Layered HMM System (nach [Zha06a])*

Low-Level Merkmale werden benutzt, um die HMM der individuellen Aktionen (I-HMM) zu trainieren. Die Ergebnisse werden mit den Gruppenmerkmalen kombiniert. Dies muss zeitsynchron erfolgen, außerdem ist eine Wahrscheinlichkeit für jede individuelle Aktion zu jedem Zeitpunkt t erwünscht. Um dies zu erreichen, verwendet man den Vorwärtsalgorithmus, der bei der Berechnung der Produktionswahrscheinlichkeiten von HMM eingesetzt wird. Hierbei ist

$$\alpha_i(t) = P(\boldsymbol{o}_1, \ldots, \boldsymbol{o}_t; q_t = s_i) \tag{6.3}$$

die Wahrscheinlichkeit, die Beobachtung vom Anfang bis zum Zeitpunkt t gemacht zu haben und im Zustand s_i zu sein. In den ursprünglichen Layered HMM von Oliver [Oli02a] werden die verschiedenen Ebenen durch eine binäre Entscheidung, welches I-HMM die größte Wahrscheinlichkeit gibt, verbunden. Eine andere Möglichkeit besteht in der Verwendung der logarithmierten Summe der $\alpha_i(t)$ über alle Zustände eines Modells (vgl. [Bar05]). In dieser Arbeit wird eine normierte Größe verwendet, die als Wahrscheinlichkeit interpretiert werden kann:

$$P(\omega_t = \varphi_k) = \frac{1}{\sum_j^{N_s^{tot}} \alpha_t(j)} \sum_i^{N_S^k} \alpha_t(i) \tag{6.4}$$

Hierbei bezeichnet ω_t eine individuelle Aktion, φ_k ist eine Repräsentation einer Aktion, N_S^k beschreibt die Anzahl der Zustände des Modells k und $N_S^{tot} = \sum_k N_S^k$ gibt die Gesamtzahl der Zustände aller Modelle an.
Diese Wahrscheinlichkeit wird für jeden Zeitschritt und für jeden Besprechungsteilnehmer für alle zu unterscheidenden Unterklassen (individuelle Aktionen) berechnet. Im konkreten Fall besteht der so entstandene Vektor für jeden Zeitschritt aus $3 \cot 4 = 12$ Einträgen, da drei individuelle Aktionen und vier Teilnehmer unterschieden werden. Um die Gruppenaktionen zuverlässig daraus bestimmen zu können, wird der Vektor mit gruppenspezifischen Low-Level Merkmalen angereichert (siehe dazu auch Tabelle 2.5). Die daraus erhaltene Merkmalssequenz mit 16-dimensionalen Vektoren bildet das Eingangssignal für die im zweiten Schritt angewendeten G-HMM, welche die Segmentierung der Besprechung in Gruppenaktionen durchführen.

6.3.3 Varianten der Layered HMM

Für die Erkennung der individuellen Aktionen müssen nicht notwendigerweise HMM verwendet werden. Auch andere Klassifikatoren sind hier denkbar, besonders, da jeder Zeitschritt isoliert behandelt werden kann und keine Merkmalssequenzen verarbeitet werden müssen. Alternativen zu den I-HMM werden in diesem Abschnitt vorgestellt (siehe auch [Rei06b]).

MLP-HMM

Ähnlich wie schon beim Tandem Ansatz oder dem Tied-Posterior Ansatz kann ein MLP verwendet werden, um die Wahrscheinlichkeiten für eine individuelle Aktion in einem Zeitschritt zu schätzen. Dies geschieht hier im Unterschied zu den I-HMM ohne Einbeziehung der zuvor beobachteten Merkmalssequenz und ist vollkommen isoliert zu betrachten. In der Ausgabeschicht wird die Softmax-Funktion verwendet, so dass für jeden Zeitschritt t für alle Klassen der individuellen Aktionen eine Auftrittswahrscheinlichkeit angegeben werden kann.

LSTM-HMM

Sehr ähnlich wie bei den MLP können rekurrente Neuronale Netze zur Erkennung der individuellen Aktionen verwendet werden. In dieser Arbeit werden LSTM herangezogen, die schon bei der Erkennung der Gruppenaktionen gute Ergebnisse erbracht haben (vgl. Abschnitt 4.3). Hier werden die einzelnen Ausgänge y_i eines LSTM mit der Summe aller Ausgänge $\sum_{j=1}^{N} y_j$ normiert, um dadurch als Wahrscheinlichkeiten interpretiert werden zu können:

$$\hat{y_i} = \frac{y_i}{\sum_{j=1}^{N} y_j} \tag{6.5}$$

SVM-HMM

Als dritte Alternative zu den I-HMM kommen Support-Vektor-Maschinen zum Einsatz. Auch dieser Klassifikator hat sich bereits in der Erkennung von Gruppenaktionen bewährt (siehe Kapitel 3.4.1). Der Ausgang eines SVM kann nicht direkt als Wahrscheinlichkeit interpretiert werden, da dieser bei einer binären SVM-Klassifikation die Distanz des Testmusters zur Hyperebene beschreibt, und somit prinzipiell einen unbeschränkten Wertebereich besitzt. Um die Ausgänge der SVM dennoch als Wahrscheinlichkeiten interpretieren zu können, werden die bei der Klassifizierung erhaltenen Werte durch eine Softmax-Funktion transformiert, so dass der Wertebereich zwischen null und eins liegt und die Summe der Werte eins ergibt.

6.4 Experimente und Ergebnisse

In diesem Abschnitt sind die Ergebnisse der in den vorigen Abschnitten beschriebenen hybriden Verfahren im Bereich der Segmentierung und Erkennung von Gruppenaktionen aufgeführt. Bei allen Versuchen werden entweder die semantischen Daten oder die Low-Level Merkmale (siehe Kapitel 2.3.3) verwendet. Die Segmentierung erfolgt stets mit Hilfe des Viterbi-Algorithmus.

6.4.1 Ergebnisse mit semantischen Merkmalen

Die Tests mit semantischen Merkmalen wurden durchgeführt, um die Ergebnisse mit denen anderer Verfahren auf diesen Daten vergleichen zu können. Wie aus der Tabelle 6.4 zu entnehmen ist, sind die hier gezeigten Verfahren zum Beispiel denen in Kapitel 3.4.2 deutlich überlegen. Das vorgestellte abgewandelte Verfahren mit LSTM in der Vorverarbeitungsstufe ist dem Standard-Tandem Ansatz überlegen, wenn die hier vorgeschlagene Mittelwertfilterung eingesetzt wird. Die Akkuratheit von 92,7 % ist nur 3.6 % unter dem Ergebnis, wenn die Segmentgrenzen manuell vorgegeben werden und die Gruppenaktionen nur durch das Ergebnis der LSTM mit Mehrheitsentscheid bestimmt werden. Dies stellt das beste Ergebnis dar, das mit den semantischen Merkmalen erreicht werden konnte.

Modell	**Akkuratheit** [%]
Tied-Posterior (L8)	87,2
Tied-Posterior (L14)	85,3
Tandem (MLP-HMM) (L8)	88,1
LSTM-vorsegmentiert (L8)	96,3
LSTM-HMM (L8)	87,2
LSTM-filter-HMM (L8)	92,7

Tabelle 6.4: *Akkuratheit hybrider Verfahren bei der Segmentierung von Gruppenaktionen, Merkmale: SEM-A, Lexikon L8 bzw. L14*

6.4.2 Ergebnisse mit Low-Level Merkmalen

Wie bei den anderen vorgestellten Verfahren in den vorangegangenen Kapiteln, werden auch hier Experimente mit Low-Level Merkmalen durchgeführt.

Tied-Posterior und Tandem

Tabelle 6.5 zeigt die Akkuratheit bei der Segmentierung und Erkennung von Gruppenaktionen, wenn die acht Klassen des Lexikon L8 unterschieden werden. Aufgrund der hohen Abtastrate konnten die Merkmalssequenzen nicht mehr ausreichend mit LSTM modelliert werden, so dass das zweistufige Verfahren, das aus dem Tandem-Verfahren abgeleitet wurde, hier nicht mehr eingesetzt werden kann. Im oberen Teil der Tabelle 6.5 sind die

Modell	**Akkuratheit** [%]
Tied-Posterior	82,7
Tandem (MLP-HMM)	77,7

Tabelle 6.5: *Akkuratheit hybrider Verfahren bei der Segmentierung von Gruppenaktionen, Merkmale: LOW, Lexikon L8*

Ergebnisse der hybriden Verfahren gezeigt. Wie aufgrund der Merkmale zu erwarten ist, sind die Ergebnisse schlechter, als bei der Verwendung der semantischen Merkmale. Es hat sich gezeigt, dass diese Ansätze nicht die Erkennungsleistung erbringen, die ein einfaches HMM leisten kann.

Zweischichtige Verfahren

Die erste Schicht dieser Ansätze modelliert individuelle Aktionen der Besprechungsteilnehmer. Diese Aktionen werden frameweise erkannt, wodurch der Einsatz von statischen Klassifikatoren möglich wird. Für den Einsatz von LSTM-Netzwerken wurden die Merkmale auf eine Abtastrate von 1 Hz konvertiert.

Modell	**Erkennungsrate** [%]
HMM	90,4
MLP	89,8
LSTM	91,2
SVM	90,4

Tabelle 6.6: *Erkennungsraten der individuellen Aktionen (idle, speaking, writing) mit verschiedenen Klassifikatoren, Merkmale: LOW*

Die Erkennungsleistung liegt bei allen vier verwendeten Klassifikatoren in der Größenordnung von 90 % (siehe Tabelle 6.6). Das beste Resultat wird mit LSTM erzielt, wobei dies zum Teil auf die verminderte Abtastrate der Merkmale zurückzuführen ist, da hier in jedem Frame die Informationen eines größeren Zeitraums enthalten sind. Die Ergebnisse von HMM und SVM unterscheiden sich kaum, sodass keinem dieser beiden Erkennern eine Präferenz gegeben werden kann. Konfusionsmatrizen der verschiedenen Ansätze sind in den Tabellen 6.7 bis 6.10 angegeben. Hier ist zu sehen, dass vor allem die Aktionen *idle* und *speaking* bei den HMM und MLP häufiger verwechselt werden. Bei den SVM hingegen ergibt sich eine größere Verwechslung zwischen den Aktionen *idle* und *writing* bzw. *speaking* und *writing*. Dies ist auf eine Überadaption aufgrund der ungleichmäßigen Verteilung der einzelnen Klassen zurückzuführen. Dennoch sind die Erkennungsraten hoch genug, um eine gute Grundlage für die Weiterverarbeitung der Ausgänge für die Erkennung von Gruppenaktionen zu bilden.
Die Weiterverarbeitung der Ergebnisse der Klassifikatoren der individuellen Aktionen erfolgt mit HMM. Tabelle 6.11 zeigt die Akkuratheit der vier unterschiedlichen zweischich-

[%]	**Idle**	**Speaking**	**Writing**
Idle	92,59	0,55	6,86
Speaking	22,55	76,68	0,77
Writing	2,53	0,01	97,47

Tabelle 6.7: *Konfusionsmatrix der erkannten individuellen Aktionen mit* HMM *(Zeilen: tatsächliche Aktion, Spalten: erkannte Aktion), Merkmale: LOW*

[%]	**Idle**	**Speaking**	**Writing**
Idle	92,71	4,06	3,24
Speaking	15,16	84,39	0,45
Writing	14,81	0,46	84,73

Tabelle 6.8: *Konfusionsmatrix der erkannten individuellen Aktionen mit* MLP *(Zeilen: tatsächliche Aktion, Spalten: erkannte Aktion), Merkmale: LOW*

[%]	**Idle**	**Speaking**	**Writing**
Idle	92,77	3,74	3,49
Speaking	14,10	85,53	0,37
Writing	8,02	0,42	91,56

Tabelle 6.9: *Konfusionsmatrix der erkannten individuellen Aktionen mit* LSTM *(Zeilen: tatsächliche Aktion, Spalten: erkannte Aktion), Merkmale: LOW*

[%]	**Idle**	**Speaking**	**Writing**
Idle	86,78	0,21	13,01
Speaking	0,95	81,11	17,94
Writing	5,06	1,17	93,77

Tabelle 6.10: *Konfusionsmatrix der erkannten individuellen Aktionen mit* SVM *(Zeilen: tatsächliche Aktion, Spalten: erkannte Aktion), Merkmale: LOW*

tigen Ansätze, in der zweiten Spalte für das Lexikon L8, in der dritten Spalte für das Lexikon L14. Wie zu erwarten, fallen die Ergebnisse mit zunehmender Anzahl von Klassen deutlich ab. Das beste Ergebnis mit Lexikon L8 (SVM - HMM) liegt 2,2 % absolut über der besten Akkuratheit bei der Verwendung von einschichtigen HMM. Dies stellt das beste Ergebnis dar, das mit diesen Merkmalen mit dem Lexikon L8 erreicht wurde.

Modell	**8 Klassen** [%]	**14 Klassen** [%]
HMM - HMM	87,8	82,0
MLP - HMM	92,8	84,2
LSTM - HMM	92,1	85,6
SVM - HMM	**94,3**	**88,5**

Tabelle 6.11: *Akkuratheit zweischichtiger Verfahren zur Segmentierung von Gruppenaktionen, Merkmale: LOW, Lexikon L8 bzw. L14*

Wird (wie zum Beispiel mit Lexikon L14) eine größere Anzahl von Klassen unterschieden, so liegen die Ergebnisse unter denen, die bei der Unterscheidung von nur acht Klassen erreicht werden. Jedoch sind die Akkuratheiten immer noch hoch genug, um eine vernünftige Gliederung einer Besprechung zu gewährleisten. Ähnliche Arbeiten auf dem gleichen Datensatz von Zhang [Zha06a], [Zha06b], [Zha04a] erreichen ähnlich hohe Erkennungsraten, konnten aber mit dem zweischichtigen SVM - HMM - System deutlich übertroffen werden.
Zusammenfassend ist festzuhalten, dass hybride Modelle nicht automatisch zu einer höheren Akkuratheit führen. Das haben die Experimente mit Tied-Posterior- und Tandem-Ansatz gezeigt. Durch die Aufspaltung der Gliederung von Besprechungen in zwei unterschiedliche Ebenen konnten dagegen durchweg gleichwertige oder bessere Ergebnisse erzielt werden. Dadurch, dass individuelle Aktionen teilnehmerübergreifend modelliert werden können, wird das Problem, zu wenig Daten zur Verfügung zu haben, relativiert. Ebenso kann auf diese Weise das jetzige System, das auf vier Personen ausgelegt ist, relativ einfach auf einen größeren Teilnehmerkreis erweitert werden. Es muss im Prinzip nur das HMM der oberen Schicht, das die Segmentierung durchführt, neu erstellt werden.
Ein weiterer Schritt zu einer noch genaueren Segmentierung von Besprechungen wäre die Einbeziehung von HCRF anstelle der HMM. Zhang [Zha06a] verwendet in seiner Arbeit CRF, um bessere Segmentierungsergebnisse zu erhalten. Deshalb ist davon auszugehen, dass mit HCRF eine noch genauere Gliederung einer Besprechung erreicht werden kann.

6.5 Kapitelzusammenfassung

Hybride Verfahren kombinieren die Vorteile zweier unterschiedlicher Ansätze, wodurch stabilere und bessere Leistungen erbracht werden sollen. In anderen Anwendungsgebieten erfolgreiche Verfahren, wie der Tied-Posterior-Ansatz und das Tandem-Verfahren, haben sich für die Untergliederung von Besprechungen nicht als geeignet erwiesen. Erst durch Abwandlung der Topologie des Tandem-Verfahrens mit dem in dieser Arbeit eingeführten

Filter und LSTM-Netzwerken anstelle von MLP konnte mit 92,7 % die höchste Akkuratheit erzielt werden, wenn semantische Merkmale verwendet wurden.
Zweischichtige Verfahren, die über einen Zwischenschritt die gewünschte Klassifizierung durchführen, sind in der vorliegenden Anwendung weniger anfällig für den häufigen Fall der zu geringen Trainingsmenge. Als Zwischenschritt wurden individuelle Aktionen der Teilnehmer modelliert. Darauf aufbauend konnte ein HMM trainiert werden, das die Segmentierung durchführte. Mit einem Ansatz, der Support-Vektor-Maschinen mit Hidden-Markov-Modellen kombiniert, konnte mit 94,3 % die höchste Akkuratheit auf diesem Datensatz erreicht werden.

Kapitel 7

Zusammenfassung und Ausblick

In der vorliegenden Arbeit wurden neuartige Verfahren zur automatischen Gliederung von Besprechungen vorgeschlagen.
Die automatische Analyse von Besprechungen wurde motiviert mit der ständig wachsenden Zahl von geschäftlichen Meetings und der daraus folgenden Informationsfülle, die ohne technische Hilfmittel nur schwer in den Griff zu bekommen ist. Während die Untersuchung von Besprechungen schon lange ein Thema psychologischer Forschung darstellt, hat die Informationstechnik diesen Bereich erst in jüngster Zeit entdeckt. Anliegen dieser Arbeit war es, eine grobe Gliederung einer Besprechung zu geben, um darauf aufbauend weitere und tiefergehendere Analysen durchführen zu können.
Zunächst wurde die zugrunde liegende Datenbank, die M4-Datenbank, und die daraus abgeleiteten Merkmale vorgestellt. Hierbei wurde zwischen manuell erzeugten Merkmalen, semantischen Merkmalen und Low-Level Merkmalen unterschieden. Zur Beurteilung der entwickelten Algorithmen wurden verschiedene Bewertungsmaße vorgestellt.
Statische Klassifikationsverfahren stellen eine Möglichkeit dar, Mustervektoren zu klassifizieren. Für die Unterscheidung von Gruppenaktionen haben sich verschiedene Klassifikatoren als geeignet erwiesen: Naive-Bayes-Klassifikator, Neuronale Netze und Support-Vektor-Maschinen. Die Segmentierung der Gruppenaktionen wurde mit zwei verschiedenen neu entwickelten Algorithmen durchgeführt, einem integrierten Verfahren und einem zweistufigen Verfahren. Dabei hat sich herausgestellt, dass das integrierte Verfahren zwar bessere Segmentierungsergebnisse liefert, der zweistufige Ansatz allerdings um ein vielfaches schneller abgearbeitet werden kann.
Weitere Ansätze zur Klassifikation von Mustern stellten Modelle dar, die neurobiologisch motiviert sind, wie Rekurrente Neuronale Netze und Neuronale Felder. Long-Short-Term-Memory Neuronale Netze haben sich als sehr effektiv im Bereich der Gruppenaktionserkennung erwiesen. Die Besonderheit dieses Modells liegt in dem konstanten Fehler-Karussell, das ein Vergessen zeitlich zurückliegender Informationen verhindert. Wurden die räumlichen Abstände der Neuronen von rekurrenten Neuronalen Netzen immer weiter verringert, bis sie quasi kontinuierlich verteilt sind, so erhielt man ein Neuronales Feld. Durch Rückkopplungen entstand ein hochdynamisches Modell. Es konnte gezeigt werden, dass durch Diskretisierung und nach Elimination der Dynamik eine Art Rekurrentes Neuronales Netz entsteht. Für die Anwendung zur Erkennung von Gruppenaktionen wurde ein Paralleles Rekurrentes Neuronales Netz verwendet, das alle zeitlichen Eingänge gleichzeitig verarbeiten konnte. Dieser Ansatz hat sich allerdings in Bezug auf die Gliederung

von Besprechungen als nicht sehr leistungsfähig herausgestellt.

Für die Untergliederung von Merkmalssequenzen werden klassischerweise dynamische Klassifikatoren mit Viterbi-Algorithmus eingesetzt. Auch in dieser Arbeit wurden Hidden-Markov-Modelle mit kontinuierlichen Ausgabewahrscheinlichkeiten für die Segmentierung von Besprechungen verwendet. Eine Erweiterung der HMM stellten Hidden Conditional Random Fields dar, die die Einschränkung der Unabhängigkeit zweier Beobachtungen aufheben konnten. Mit beiden dynamischen Ansätzen konnten sehr gute Ergebnisse erreicht werden.

Aufgrund der Tatsache, dass nur eine sehr geringe Menge von Material zum Trainieren der Parameter komplexer Modelle vorhanden ist, konnten die vorherigen Verfahren unter Umständen nicht optimal trainiert werden. Daher wurde der Prozess der Untergliederung von Besprechungen in zwei aufeinanderfolgende Stufen unterteilt. In der ersten Stufe wurden teilnehmerspezifische Aktivitäten detektiert; die zweite Stufe erkannte unter Verwendung der Ergebnisse der ersten Stufe die Gruppenaktionen. Mit einer Kombination aus Support-Vektor-Maschinen in der unteren Schicht und Hidden-Markov-Modellen in der zweiten Schicht konnten die besten Ergebnisse erzielt werden. Die Ergebnisse anderer Forschungsgruppen auf diesem Gebiet konnten erreicht und übertroffen werden.

Zusammengefasst wurde somit das Ziel dieser Arbeit, eine Untergliederung von Besprechungen in sinnvolle Abschnitte durchzuführen, um darauf aufbauend weitergehende Analysen vorzunehmen, erreicht. Die vorgestellten statischen Klassifikationsverfahren sind in der Lage, mit semantischen Merkmalen eine sinnvolle Segmentierung durchzuführen. Für Low-Level Merkmale jedoch sind diese nur eingeschränkt geeignet. Der innovative Ansatz eines Parallelen Neuronalen Netzes konnte die Erwartungen nicht erfüllen und wurde daher wieder verworfen. Deutlich bessere Ergebnisse, besonders bei der Zuordnung der Segmentgrenzen, konnten mit dynamischen Klassifikationsverfahren erreicht werden; die Spärlichkeit der Trainingsdaten machte es jedoch schwierig, diese Modelle ausreichend genau zu trainieren. Daher wurden die stabilsten und besten Ergebnisse mit einem hybriden zweistufigen Verfahren erreicht, das die Vorteile von SVM und HMM erfolgreich kombiniert.

Abschließend wird hier noch auf mögliche künftige Forschungsaktivitäten hingewiesen, die sich aus dieser Arbeit ergeben. Zur Modellierung einzelner Segmente könnte eine Zeitreihenanalyse vorgenommen werden. Diese Methode hat sich auf anderen Gebieten bereits bewährt [Sch06] und kann für diese Thematik ebenso geeignet sein.

Für eine robustere Segmentierung der Gruppenaktionen lohnt es sich, ein Sprachmodell einzuführen, wie es in der Spracherkennung schon lange der Fall ist.

In dieser Arbeit wurde eine feste Konfiguration eines Meetingraumes und eine feste Anzahl von Teilnehmern behandelt. In weitergehenden Untersuchungen muss die Anzahl der Besprechungsteilnehmer variabel gehalten werden. Personen können innerhalb eines Meetings den Raum verlassen, oder hereinkommen. Auch diese Vorkommnisse können mit einer Erweiterung der Algorithmen behandelt werden. Der feste Aufbau der Kameras und der Mikrophone stellt eine Einschränkung dar, die es in Zukunft zu überwinden gilt.

So sollte es möglich sein, auch mit nur einer Kamera, beispielsweise einer omnidirektionalen Kamera, eine Besprechung aufzuzeichnen. Die Ansteckmikrophone, die die Bewegungsfreiheit der Teilnehmer beeinträchtigen, könnten in Zukunft durch eine verbesserte Auswertung der Signale eines Mikrophonarrays ersetzt werden.
Mit dieser Arbeit wurde mit dazu beigetragen, Besprechungen automatisch zu analysieren und zu verstehen. Zukünftige Arbeiten können die begonnenen Arbeiten fortsetzen und ein leistungsfähiges Gesamtsystem erstellen.

Anhang A

Abkürzungen und Formelzeichen

Abkürzungen

AER	Action Error Rate
BSP	Binary Speech Profile
CRF	Conditional Random Field
DCT	Discrete Cosinus Transform
EM	Expectation Maximization
FER	Frame Error Rate
GMF	Global Motion Features
GMM	Gaussian Mixture Model
HCRF	Hidden Conditional Random Field
HMM	Hidden-Markov-Model
KNN	Künstliches Neuronales Netz
L8	Lexikon mit acht Klassen
L14	Lexikon mit 14 Klassen
LOW	automatisch erzeugte Low-Level Merkmale
MFCC	Mel-Frequency Cepstral Coefficient
MLP	Multi-Layer Perceptron
MRF	Markov Random Field
PRNN	Paralleles Rekurrentes Neuronales Netz
RBF	Radial Basis Funktion
RNN	Rekurrentes Neuronales Netz
SEM-M	manuell erzeugte semantische Merkmale (Annotationen)
SEM-M+S	SEM-M mit automatischer Sprechersegmentierung
SEM-A	automatisch erzeugte semantische Merkmale
SIFT	Simple Inverse Filter Tracking
SMA	Simple-Moving-Average, Gleitender Mittelwertfilter
SRP	Steered Response Power
SVM	Support-Vektor-Maschine

Griechische Formelzeichen

α_i	Lagrange-Multiplikator
$\alpha_t(i)$	Vorwärtswahrscheinlichkeit
$\beta_t(i)$	Rückwärtswahrscheinlichkeit
Δ	Differenz
ϵ	Lernfehler
η	Lernrate
ϑ	allgemeine Parameter
κ	Klassenindex
κ_e	Index der erkannten Klasse
$\boldsymbol{\mu}$	Mittelwertvektor
$\boldsymbol{\sigma}(t)$	Standardabweichung
Σ	Kovarianzmatrix
Ω	Menge der Klassen
Ω_κ	Musterklasse mit $\Omega_\kappa \in \Omega$ und $\kappa = 1, \ldots, K$

Lateinische Formelzeichen

Acc	Akkuratheit, Genauigkeit
c_i	Cepstralkoeffizient mit Index i
E	Signalenergie
E	Fehlerrate
f	Frequenz
F0	Grundfrequenz
K	Anzahl der zu unterscheidenden Klassen K=$\lvert\Omega\rvert$
n	Laufvariable
N	Dimension eines Vektors oder Feldes
N	Anzahl der Cepstralkoeffizienten
N_M	Anzahl der Gauß-Mixturen eines HMM
N_S	Anzahl der verborgenen Zustände eines HMM oder HCRF
$\mathcal{N}$	Normalverteilung
$\boldsymbol{O}$	Beobachtungssequenz eines dynamischen Modells
$\boldsymbol{o}$	Einzelne Beobachtung eines dynamischen Modells
P(.)	allgemeine Wahrscheinlichkeit
$\mathbb{R}^n$	Reeller Raum der Dimension n
T_I, T_R	Schwellwert
w_i	Gewichtungsfaktor
$\boldsymbol{W}$	Gewichtsmatrix
x	Variable, Koordinate
$\boldsymbol{x}$	Merkmalsvektor
$\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$	Sequenz von Merkmalsvektoren
y	Variable, Koordinate

y_i	Ausgang eines Neuronalen Netzes

Funktionen

$F(s)$	Aktivierungsfunktion
$G(\boldsymbol{x})$	Propagierungsfunktion
i(t)	Intensität einer Bewegung
$I_a(x,y,t)$,$I_b(x,y,t)$	Intensität eines Grauwerts am Ort (x,y) zum Zeitpunkt t
$I_d(x,y,t)$	Intensität des Differenzbildes am Ort (x,y) zum Zeitpunkt t
$K(\boldsymbol{x}_1,\boldsymbol{x}_2)$	Kernelfunktion
$\boldsymbol{m'}(t)$	Massenschwerpunkt
$\boldsymbol{m}(t)$	Zentrum einer Bewegung
Mel(f)	Mel-Skala in Abhängigkeit der Frequenz f
$s(n)$	Zeitsignal
$w(n)$	Allgemeine Fensterfunktion
$w_{Ham}(n)$	Hamming-Funktion
$(.)^T$	Transponierte eines Vektors
$(.)^{-1}$	Matrixinvertierung

Literaturverzeichnis

[AH05a] Marc Al-Hames. "Report on Implementation and Evaluation Results of Audio, Video, and Multimodal Algorithms". AMI Deliverable 4.2, Dezember 2005.

[AH05b] Marc Al-Hames und Gerhard Rigoll. "A Multi-Modal Mixed-State Dynamic Bayesian Network for Robust Meeting Event Recognition from Disturbed Data". In *Proceedings of the 6th International Conference on Multimedia and Expo (ICME)*. Amsterdam, Niederlande, Juli 2005.

[AH06] Marc Al-Hames, Alfred Dielmann, Daniel Gatica-Perez, Stephan Reiter, Steve Renals, Gerhard Rigoll und Dong Zhang. "Multimodal Integration for Meeting Group Action Segmentation and Recognition". In S. Renals und S. Bengio (Herausgeber), *MLMI 2005, 2nd Joint Workshop on Multimodal Interaction and Related Machine Learning Algorithms*. Springer Verlag, 2006.

[AH07a] Marc Al-Hames, Benedikt Hörnler, Ronald Müller, Joachim Schenk und Gerhard Rigoll. "Automatic Multi-Modal Meeting Camera Selection for Video-Conferences and Meeting Browsing". In *Proceedings of the 8th International Conference on Multimedia and Expo (ICME)*. Beijing, China, Juli 2007.

[AH07b] Marc Al-Hames, Claus Lenz, Stephan Reiter, Joachim Schenk und Gerhard Rigoll. "Robust Multi-Modal Group Action Recognition in Meetigs from Disturbed Videos with the Asynchronous Hidden Markov Model". In *Proceedings of the International Conference on Image Processing (ICIP)*. San Antonio, Texas, USA, September 2007.

[Ahr99] Ingo Ahrns und Heiko Neumann. "Space-Variant Dynamic Neural Fields for Visual Attention". In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, Band 2, Seiten 313–318. Fort Collins, CO, USA, Juni 1999. doi:10.1109/CVPR.1999.784650.

[Ala96] Cesar Martin del Alamo, Francisco Javier Caminero-Gil, Celinda de la Torre-Munilla und Luis Alfonso Hernandez-Gomez. "Discriminative training of GMM for speaker identification". In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Seiten 89–92. IEEE Computer Society, Washington, DC, USA, 1996. ISBN 0-7803-3192-3. doi: http://dx.doi.org/10.1109/ICASSP.1996.540297.

[All99] Carolyn F. Allex, Jude W. Shavlik und Frederick R. Blattner. "Neural network input representations that produce accurate consensus sequences from

DNA fragment assemblies". *Bioinformatics*, Band 15, Nr. 9, Seiten 723–728, 1999.

[Ama77] Shun-Ichi Amari. "Dynamics of Pattern Formation in Lateral-Inhibition Type Neural Fields". *Biological Cybernetics*, Band 27, Nr. 2, Seiten 77–87, 1977.

[Bah86] Lalit R. Bahl, Peter F. Brown, Peter V. de Souza und Robert L. Mercer. "Maximum Mutual Information Estimation of Hidden Markov Model Parameters for Speech Recognition". In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Band 11, Seiten 49–52. Tokyo, Japan, April 1986.

[Bak01] Bram Bakker. "Reinforcement Learning with LSTM in Non-Markovian Tasks with Long-Term Dependencies". Technischer Bericht, Informatics Institute, University of Amsterdam, 2001.

[Bal51] Robert Freed Bales. *Interaction process analysis: A method for the study of small groups.* Addison-Wesley, 1951, 203 Seiten.

[Bar05] Mark Barnard. *Multimedia event modelling and recognition.* Dissertation, Ecole Polytechnique Federale de Lausanne, September 2005.

[Bel59] Richard Ernest Bellman. *Dynamic programming.* University Press, Princeton, NJ, 2. Auflage, 1959, 339 Seiten.

[Ber96] Adam L. Berger, Vincent J. Della Pietra und Stephen A. Della Pietra. "A maximum entropy approach to natural language processing". *Computational Linguistics*, Band 22, Nr. 1, Seiten 39–71, 1996. ISSN 0891-2017.

[Ber99] Thomas Bergener, Carsten Bruckhoff, Percy Dahm, Herbert Janßen, Frank Joublin, Rainer Menzner, Axel Steinhage und Werner von Seelen. "Complex behavior by means of dynamical systems for an anthropomorphic robot". *Neural Networks*, Band 12, Nr. 7-8, Seiten 1087–1099, Oktober 1999. ISSN 0893-6080. doi:http://dx.doi.org/10.1016/S0893-6080(99)00045-3.

[Bil98] Jeff A. Bilmes. "A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models". Technischer Bericht TR-97-021, International Computer Science Institute (ICSI), Berkeley, Kalifornien, USA, April 1998.

[Bou94] Hervé Bourlard und Nelson Morgan. *Connectionist Speech Recognition: A Hybrid Approach.* Kluwer Academic Publishers, 1994, 312 Seiten.

[Bou06] Herve Bourlard und Steve Renals. "AMI overview and prospects for future research". AMI white paper, Januar 2006.

[Bra97] Michael S. Brandstein und Harvey F. Silverman. "A Robust Method for Speech Signal Time-Delay Estimation in Reverberant Rooms". In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Band 1, Seiten 375–378. IEEE Computer Society, Washington, DC, USA, 1997. ISBN 0-8186-7919-0. doi:10.1109/ICASSP.1997.599651.

[Bur98] Christopher J. C. Burges. "A Tutorial on Support Vector Machines for Pattern Recognition". *Data Mining and Knowledge Discovery*, Band 2, Nr. 2, Seiten 121–167, 1998. ISSN 1384-5810. doi:http://dx.doi.org/10.1023/A:1009715923555.

[Cha93] Hung-Jen Chang und Joydeep Ghosh. "Pattern association and retrieval in a continuous neural system". *Biological Cybernetics*, Band 69, Nr. 1, Seiten 77–86, 1993.

[Cli90] Peter Clifford. "Markov random fields in statistics". In G. R. Grimmett und D. J. A. Welsh (Herausgeber), *Disorder in Physical Systems. A Volume in Honour of John M. Hammersley*, Seiten 19–32. Clarendon Press, Oxford, 1990.

[Coo05] Stephen Coombes. "Waves, bumps, and patterns in neural field theories". *Biological Cybernetics*, Band 93, Nr. 2, Seiten 91–108, August 2005.

[Cor95] Corinna Cortes und Vladimir Vapnik. "Support-Vector Networks". *Machine Learning*, Band 20, Nr. 3, Seiten 273–297, 1995. ISSN 0885-6125. doi:http://dx.doi.org/10.1023/A:1022627411411.

[Cov65] Thomas M. Cover. "Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition". *IEEE Transactions on Electronic Computers*, Band EC-14, Seiten 326–334, 1965.

[Cut02] Ross Cutler, Yong Rui, Anoop Gupta, JJ Cadiz, Ivan Tashev, Li wei He, Alex Colburn, Zhengyou Zhang, Zicheng Liu und Steve Silverberg. "Distributed Meetings: A Meeting Capture and Broadcasting System Broadcasting System". In *Proceedings of ACM Multimedia Conference*. Juan Les Pins, Frankreich, Dezember 2002.

[Del95] Stephen Della Pietra, Vincent Della Pietra und John Lafferty. "Inducing features of random fields". Technischer Bericht CMU-CS-95-144, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, Mai 1995.

[Dem77] Arthur P. Dempster, Nan M. Laird und Donald B. Rubin. "Maximum Likelihood from Incomplete Data via the EM Algorithm". *Journal of the Royal Statistical Society B*, Band 39, Nr. 1, Seiten 1–38, 1977.

[DiB00] Joseph Hector DiBiase. *A High-Accuracy, Low-Latency Technique for Talker Localization in Reverberant Environments Using Microphone Arrays*. Dissertation, Division of Engineering at Brown University, Providence, Rhode Island, Mai 2000.

[DiB01] Joseph Hector DiBiase, Harvey Fox Silverman und Michael Shapiro Brandstein. "Robust Localization in Reverberant Rooms". In M. Brandstein und D. Ward (Herausgeber), *Microphone Arrays*, Kapitel 8, Seiten 157–180. Springer, 2001.

[Die04] Alfred Dielmann und Steve Renals. "Dynamic Bayesian Networks for Meeting Structuring". In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Band 5, Seiten 629–632. Montreal, Kanada, Mai 2004. doi:10.1109/ICASSP.2004.1327189.

[Die07] Alfred Dielmann und Steve Renals. "Automatic Meeting Segmentation using Dynamic Bayesian Networks". *IEEE Transactions on Multimedia*, Band 9, Nr. 1, Seiten 25–36, Januar 2007. AMI-125.

[Dro05] Jasha Droppo, Milind Mahajan, Asela Gunawardana und Alex Acero. "How to train a discriminative front end with stochastic gradient descent and maximum mutual information". In *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, Seiten 41–46. San Juan, Puerto Rico, November 2005. doi:10.1109/ASRU.2005.1566501.

[Eck02] Douglas Eck. "A First Look at Music Composition using LSTM Recurrent Neural Networks". Technical Report IDSIA-07-02, IDSIA, 2002.

[Ede01] Hannes Edelbrunner, Uwe Handmann, Christian Igel, Iris Leefken und Werner von Seelen. "Application and Optimization of Neural Field Dynamics for Driver Assistance". In *Proceedings of the International Conference on Intelligent Transportation Systems (ITSC'01)*, Seiten 309–314. IEEE Press, Oakland, Kalifornien, USA, August 2001.

[Eic98] Stefan Eickeler, Stefan Müller und Gerhard Rigoll. "Person-Independent Continuous Online Recognition of Gestures". In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Santa Barbara, USA, Juni 1998. Demo session.

[Eic00] Stefan Eickeler und Gerhard Rigoll. "A Novel Error Measure for the Evaluation of Video Indexing Systems". In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Istambul, Türkei, Juni 2000.

[Elm90] Jeffrey L. Elman. "Finding Structure in Time". *Cognitive Science*, Band 14, Seiten 179–211, 1990. doi:http://dx.doi.org/10.1016/0364-0213(90)90002-E.

[Erm98] Bard Ermentrout. "Neural networks as spatio-temporal pattern-forming systems". *Reports on Progress in Physics*, Band 61, Seiten 353–430, April 1998.

[Fay00] Nicolas Fay, Simon Garrod und Jean Carletta. "Group Discussion as Interactive Dialogue or as Serial Monologue: The Influence of Group Size". *Psychological Science*, Band 11, Nr. 6, Seiten 487–492, 2000.

[Fin98] Shai Fine, Yoram Singer und Naftali Tishby. "The Hierarchical Hidden Markov Model: Analysis and Applications". *Machine Learning*, Band 32, Nr. 1, Seiten 41–62, 1998. ISSN 0885-6125. doi:http://dx.doi.org/10.1023/A:1007469218079.

[Gar96] Ullas Gargi und Rangachar Kasturi. "An Evaluation of Color Histogram Based Methods in Video Indexing". In *International Workshop on Image Databases and Multi-Media Search*, Seiten 75–82. Amsterdam, Niederlande, August 1996.

[Ger01a] Felix A. Gers. *Long Short-Term Memory in Recurrent Neural Networks*. Dissertation, Department of Computer Science, Swiss Federal Institute of Technology, Lausanne, EPFL, Schweiz, 2001.

[Ger01b] Felix A. Gers und Jürgen Schmidhuber. "LSTM recurrent networks learn simple context free and context sensitive languages". *IEEE Transactions on Neural Networks*, Band 12, Nr. 6, Seiten 1333–1340, 2001.

[Gie00] Martin A. Giese. "Neural field model for the recognition of biological motion patterns". In *Proceedings of the Second International ICSC Symposium on Neural Computation (NC 2000)*. Berlin, Deutschland, Mai 2000.

[Gra05] Alex Graves und Jürgen Schmidhuber. "Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures". *Neural Networks*, Band 18, Nr. 5–6, Seiten 602–610, 2005. doi:http://dx.doi.org/10.1016/j.neunet.2005.06.042.

[Gun05] Asela Gunawardana, Milind Mahajan, Alex Acero und John C. Platt. "Hidden Conditional Random Fields for Phone Classification". In *Proceedings of the International Conference on Speech Communication and Technology (INTERSPEECH)*. Lissabon, Portugal, September 2005.

[Gut05] Bernd Gutmann. *Relational Conditional Random Fields*. Diplomarbeit, Albert-Ludwigs-Universität Freiburg, 2005.

[Ham95] Arun Hampapur, Ramesh Jain und Terry E Weymouth. "Production Model Based Digital Video Segmentation". *Multimedia Tools and Aplications*, Band 1, Nr. 1, Seiten 9–46, März 1995.

[Hay94] Simon S. Haykin. *Neural Networks - A Comprehensive Foundation*. Macmillan Publishing Company, 1994, 696 Seiten.

[Her00] Hynek Hermansky, Daniel P.W. Ellis und Sangita Sharma. "Tandem connectionist feature extraction for conventional HMM systems". In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Istambul, Türkei, Juni 2000.

[Hil03] Dustin Hillard, Mari Ostendorf und Elizabeth Shriberg. "Detection Of Agreement Vs. Disagreement In Meetings: Training With Unlabeled Data". In *Proceedings of the Conference on Human Language Technologies: The Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL)*. Edmonton, Kanada, Mai 2003.

[Hoc91] Josef Hochreiter. *Untersuchungen zu dynamischen Neuronalen Netzen*. Diplomarbeit, Institut für Informatik, Technische Universität München, 1991.

[Hoc97] Sepp Hochreiter und Jürgen Schmidhuber. "Long Short-Term Memory". *Neural Computation*, Band 9, Nr. 8, Seiten 1735–1780, 1997.

[Hou05] Charlse H. House. "Visual Lexicons: The Quest for Data - Driven Decision Making". Invited Presentation at MLMI 2005, 2nd Joint Workshop on Multimodal Interaction and Related Machine Learning Algorithms, 2005.

[Hu00] Mike Hu, Chris Ingram, Monica Sirski, Chris Pal, Sajani Swamy und Cheryl Patten. "A Hierarchical HMM Implementation for Vertebrate Gene Splice Site Prediction". Technischer Bericht, Deptartment of Computer Science, University of Waterloo, 2000.

[Ige00] Christian Igel und Michael Hüsken. "Improving the Rprop Learning Algorithm". In H. Bothe und R. Rojas (Herausgeber), *Proceedings of the Second International ICSC Symposium on Neural Computation (NC 2000)*, Seiten 115–121. ICSC Academic Press, Berlin, Deutschland, Mai 2000.

[Ige01] Christian Igel, Wolfram Erlhagen und Dirk Jancke. "Optimization of Dynamic Neural Fields". *Neurocomputing*, Band 36, Nr. 1-4, Seiten 225–233, 2001.

[Ima97] Toru Imai, Richard Schwartz, Francis Kubala und Long Nguyen. "Improved Topic Discrimination of Broadcast News Using a Model of Multiple Simultaneous Topics". In *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '97)*, Band 2, Seite 727. IEEE Computer Society, Washington, DC, USA, 1997. ISBN 0-8186-7919-0.

[Iva00] Yuri A. Ivanov und Aaron F. Bobick. "Recognition of Visual Activities and Interactions by Stochastic Parsing". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Band 22, Nr. 8, Seiten 852–872, 2000. ISSN 0162-8828. doi:http://dx.doi.org/10.1109/34.868686.

[Jai96] Anil K. Jain, Jianchang Mao und K. M. Mohiuddin. "Artificial Neural Networks: A Tutorial". *IEEE Computer*, Band 29, Nr. 3, Seiten 31–44, 1996.

[Jeb99] Tony Jebara und Alex Pentland. "Action Reaction Learning: Automatic Visual Analysis and Synthesis of Interactive Behaviour". In *Proceedings of the First International Conference on Computer Vision Systems (ICVS)*, Seiten 273–292. Springer-Verlag, London, Großbritannien, 1999. ISBN 3-540-65459-3.

[Jen96] Finn V. Jensen. *An introduction to Bayesian networks*. UCL Press, 1996.

[Jir01] Viktor K. Jirsa, Kelly J. Jantzen, Armin Fuchs und J. A. Scott Kelso. "Neural Field Dynamics on the Folded Three-Dimensional Cortical Sheet and Its Forward EEG and MEG". In *Proceedings of the 17th International Conference on Information Processing in Medical Imaging (IPMI'01)*, Seiten 286–299. Springer-Verlag, London, Großbritannien, 2001. ISBN 3-540-42245-5.

[Joa98] Thorsten Joachims. "Text categorization with support vector machines: learning with many relevant features". In Claire Nédellec und Céline Rouveirol (Herausgeber), *Proceedings of the European Conference on Machine Learning (ECML)*, Seiten 137–142. Springer, Chemnitz, 1998.

[Joh95] George H. John und Pat Langley. "Estimating Continuous Distributions in Bayesian Classifiers". In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, Seiten 338–345. Morgan Kaufmann, San Mateo, 1995.

[Joh98] Neil Johnson, Aphrodite Galata und David Hogg. "The Acquisition and Use of Interaction Behavior Models". In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, Seite 866. IEEE Computer Society, Santa Barbara, Kalifornien, USA, 1998. ISBN 0-8186-8497-6.

[Jon02] Michael J. Jones und James M. Rehg. "Statistical Color Models with Application to Skin Detection". *International Journal of Computer Vision*, Band 46, Nr. 1, Seiten 81–96, 2002. doi:10.1023/A:1013200319198.

[Jor86] Michael I. Jordan. "Attractor dynamics and parallelism in a connectionist sequential machine". In *Proceedings of the Eighth Conference of the Cognitive Science Society*, Seiten 531–546. Englewood Cliffs, NJ, USA : Erlbaum, 1986.

[Kin71] Nick G. Kingsbury und Peter J. W. Rayner. "Digital Filtering using Logarithmic Arithmetic". *Electronics Letters*, Band 7, Seiten 56–58, 1971.

[Kip01] Michael Kipp. "ANVIL A Generic Annotation Tool for Multimodal Dialogue". In *Proceedings of EUROSPEECH*. Aalborg, Dänemark, September 2001.

[Kre92] Bart Krekelberg und Joost Nico Kok. "A lateral inhibition neural network that emulates a winner-takes-all algorithm". Technischer Bericht RUU-CS-92-46, Utrecht University, Department of Computer Science, 1992.

[Kub99] Francis Kubala, Sean Colbath, Daben Liu und John Makhoul. "Rough'n'Ready: a meeting recorder and browser". *ACM Computing Surveys*, Band 31, Nr. 2es, Seite 7, 1999. ISSN 0360-0300. doi:http://doi.acm.org/10.1145/323216.323354.

[Kun01] Ludmila I. Kuncheva, James C. Bezdek und Robert P. Duin. "Decision templates for multiple classifier fusion: an experimental comparison". *Pattern Recognition*, Band 34, Nr. 2, Seiten 299–314, Februar 2001. doi:10.1016/S0031-3203(99)00223-X.

[Laf01] John Lafferty, Andrew McCallum und Fernando Pereira. "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data". In *Proceedings of the International Conference on Machine Learning (ICML)*, Seiten 282–289. Morgan Kaufmann, San Francisco, Kalifornien, Massachusetts, USA, Juni 2001.

[Lat03] Guillaume Lathoud, Iain A. McCowan und Darren C. Moore. "Segmenting Multiple Concurrent Speakers Using Microphone Arrays". In *Proceedings of EUROSPEECH*. Genf, Schweiz, September 2003. IDIAP-RR 03-21.

[Lat04] Guillaume Lathoud, Iain A. McCowan und Jean-Marc Odobez. "Unsupervised Location-Based Segmentation of Multi-Party Speech". In *Proceedings of the 2004 ICASSP-NIST Meeting Recognition Workshop*. Montreal, Kanada, Mai 2004. IDIAP-RR 04-14.

[Law97] Steve Lawrence, C. Lee Giles, A. C. Tsoi und A. D. Back. "Face Recognition: A Convolutional Neural Network Approach". *IEEE Transactions on Neural Networks*, Band 8, Nr. 1, Seiten 98–113, 1997.

[Lee01] Iris Leefken, Axel Steinhage und Werner von Seelen. "Generating Complex Driving Behavior by Means of Neural Fields". In *Autonome Mobile Systeme 2001, 17. Fachgespräch*, Seiten 71–79. Springer-Verlag, London, Großbritannien, 2001. ISBN 3-540-42552-7.

[Lee03] Yoonkyung Lee, Yi Lin und Grace Wahba. "Multicategory Support Vector Machines, Theory, and Application to the Classification of Microarray Data and Satellite Radiance Data". Technischer Bericht, Department of Statistics, University of Wisconsin, 1210 West Dayton St., Madison, WI 53706, Mai 2003.

[Lev83] Steve E. Levinson, Lawrence R. Rabiner und Mohan M. Sondhi. "An introduction to the application of the theory of probabilistic functions of a Markov process to automatic speech recognition". *The Bell System Technical Journal*, Band 62, Nr. 4, Seiten 1035–1074, 1983.

[Liu03] Xiao-Peng Liu, Hong-Jie Xing und Xi-Zhao Wang. "A Multistage Support Vector Machine". In *Proceedings of the International Conference on Machine Learning and Cybernetics*. Xi-an, China, November 2003.

[Mac67] James B. MacQueen. "Some Methods for classification and Analysis of Multivariate Observations". In *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability*, Band 1, Seiten 281–297. University of California Press, Berkeley, 1967.

[Mah06] Milind Mahajan, Asela Gunawardana und Alex Acero. "Training Algorithms for Hidden Conditional Random Fields". In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Toulouse, Frankreich, Mai 2006.

[Mar72] John D. Markel. "The SIFT algorithm for fundamental frequency estimation". *IEEE Transactions on Audio and Electroacoustics*, Band 20, Nr. 5, Seiten 367–377, Dezember 1972.

[McC43] Warren S. McCulloch und Walter Pitts. "A Logical Calculus of the Ideas Immanent in Nervous Activity". *Bulletin of Mathematical Biophysics*, Band 5, Seiten 115–133, 1943.

[McC00] Andrew McCallum, Dayne Freitag und Fernando Pereira. "Maximum Entropy Markov Models for Information Extraction and Segmentation". In *Proceedings of the 17th International Conference on Machine Learning (ICML)*, Seiten 591–598. Morgan Kaufmann, San Francisco, Kalifornien, San Francisco, Kalifornien, USA, 2000.

[McC03] Iain McCowan, Samy Bengio, Daniel Gatica-Perez, Guillaume Lathoud, Florent Monay, Darren Moore, Pierre Wellner und Herve Bourlard. "Modeling Human Interaction in Meetings". In *Proceedings of International Conference on Acoustics, Speech and Signal Processing*. Hong Kong, April 2003. IDIAP-RR 02-59.

[McC05] Iain McCowan, Daniel Gatica-Perez, Samy Bengio, Guillaume Lathoud, Mark Barnard und Dong Zhang. "Automatic Analysis of Multimodal Group Actions in Meetings". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Band 27, Nr. 3, Seiten 305–317, März 2005.

[McG82] Joseph E. McGrath und David A. Kravitz. "Group Research". *Annual Reviews of Psychology*, Band 33, Seiten 195–230, 1982.

[McG84] Joseph E. McGrath. *Groups: Interaction and Performance*. Prentice-Hall, 1984. ISBN 0133657000, 287 Seiten.

[Men00] Rainer Menzner, Axel Steinhage und Wolfram Erlhagen. "Generating Interactive Robot Behavior: A Mathematical Approach". In *From Animals to Animats 6: Proceedings of the Sixth International Conference On Simulation of Adaptive Behavior*, Seiten 135–144. The MIT Press/Bradford Books, Honolulu, Hawaii, 2000.

[MM03] Stephane Marchand-Maillet. "Meeting Modelling for Enhanced Browsing". Technischer Bericht TR0301, University of Geneva, 2003.

[Moo02] Darren Moore. "The IDIAP Smart Meeting Room". IDIAP-COM 07, IDIAP, 2002.

[Mor01] Nelson Morgan, Don Baron, Jane Edwards, Dan Ellis, David Gelbart, Adam Janin, Thilo Pfau, Elizabeth Shriberg und Andreas Stolcke. "The Meeting Project at ICSI". In *Proceedings of the Human Language Technology Conference*. San Diego, Kalifornien, USA, März 2001.

[Nie03] Heinrich Niemann. *Klassifikation von Mustern*. Internetpublikation, 2003. http://www5.informatik.uni-erlangen.de/Personen/niemann/klassifikation-von-mustern/m00links.html.

[Oli00] Nuria M. Oliver, Barbara Rosario und Alex P. Pentland. "A Bayesian Computer Vision System for Modeling Human Interactions". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Band 22, Nr. 8, Seiten 831–843, August 2000. doi:10.1109/34.868684.

[Oli02a] Nuria Oliver, Eric Horvitz und Ashutosh Garg. "Hierarchical Representations for Learning and Inferring Office Activity from Multiple Sensory Channels". Technischer Bericht, Microsoft Research, 2002.

[Oli02b] Nuria Oliver, Eric Horvitz und Ashutosh Garg. "Layered Representations for Human Activity Recognition". In *Proceedings of the International Conference on Multimodal Interfaces (ICMI)*, Seite 3. IEEE Computer Society, Washington, DC, USA, 2002. ISBN 0-7695-1834-6.

[Oli04] Nuria Oliver, Ashutosh Garg und Eric Horvitz. "Layered representations for learning and inferring office activity from multiple sensory channels". *Computer Vision and Image Understanding*, Band 96, Nr. 2, Seiten 163–180, 2004. ISSN 1077-3142. doi:http://dx.doi.org/10.1016/j.cviu.2004.02.004.

[Osu97] Edgar Osuna, Robert Freund und Federico Girosi. "Training Support Vector Machines: an Application to Face Detection". In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, Seite 130. IEEE Computer Society, Washington, DC, USA, 1997.

[Per06] Christine Perey. "The Future of Business Meetings: Applications of AMI Technologies". AMI white paper, 2006.

[Pet05] Leif E. Peterson, Mustafa Ozen, Halime Erdem, Andrew Amini, Lori Gomez, Colleen C. Nelson und Michael Ittmann. "Artificial Neural Network Analysis of DNA Microarray-based Prostate Cancer Recurrence". In *Proceedings of the IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB)*, Seiten 1–8. San Diego, Kalifornien, USA, November 2005.

[Qua05] Ariadna Quattoni, Michael Collins und Trevor Darrell. "Conditional Random Fields for Object Recognition". In Lawrence K. Saul, Yair Weiss und Léon Bottou (Herausgeber), *Advances in Neural Information Processing Systems 17*, Seiten 1097–1104. MIT Press, Cambridge, MA, 2005.

[Qua06] Ariadna Quattoni, Sy Bor Wang, Louis-Philippe Morency, Michael Collins und Trevor Darrell. "Hidden-state Conditional Random Fields". Technischer Bericht, Massachusetts Institute of Technology, 2006.

[Rab89] Lawrence R. Rabiner. "A tutorial on hidden Markov models and selected applications in speech recognition". *Proceedings of the IEEE*, Band 77, Nr. 2, Seiten 257–286, Februar 1989.

[Rat97] Adwait Ratnaparkhi. "A Simple Introduction to Maximum Entropy Models for Natural Language Processing". Technischer Bericht, Institute for Research in Cognitive Science, University of Pennsylvania, 1997.

[Rei04a] Stephan Reiter und Gerhard Rigoll. "Multimodal Meeting Event Recognition fusing Three different Types of Recognition Techiques". Joint AMI/PASCAL/IM2/M4 Workshop on Multimodal Interaction and Related Machine Learning Algorithms (MLMI), Martigny, Schweiz, Juni 2004.

[Rei04b] Stephan Reiter und Gerhard Rigoll. "Segmentation and Classification of Meeting Events using Multiple Classifier Fusion and Dynamic Programming". In B. Werner (Herausgeber), *Proceedings of the 17th International Conference on Pattern Recognition (ICPR 2004)*, Band 3, Seiten 434–437. IEEE Computer Society, Cambridge, Großbritannien, August 2004.

[Rei05a] Stephan Reiter und Gerhard Rigoll. "Meeting Event Recognition using a Parallel Recurrent Neural Net Approach". 2nd Joint Workshop on Multimodal Interaction and Related Machine Learning Algorithms, (MLMI), Edinburgh, Großbritannien, Juli 2005.

[Rei05b] Stephan Reiter und Gerhard Rigoll. "Multimodal Meeting Analysis by Segmentation and Classification of Meeting Events based on a Higher Level Semantic Approach". In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Philadelphia, PA, USA, März 2005.

[Rei05c] Stephan Reiter und Gerhard Rigoll. "A Neural-Field-like Approach for Modeling Human Group Actions in Meetings". In *Proceedings of the International Conference on Multimedia and Expo (ICME)*. Amsterdam, Niederlande, Juli 2005.

[Rei06a] Stephan Reiter, Björn Schuller und Gerhard Rigoll. "A combined LSTM-RNN - HMM - Approach for Meeting Event Segmentation and Recognition". In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Toulouse, Frankreich, Mai 2006.

[Rei06b] Stephan Reiter, Björn Schuller und Gerhard Rigoll. "Segmentation and Recognition of Meeting Events using a two-layered HMM and a combined MLP-HMM approach". In *Proceedings of the International Conference on Multimedia and Expo (ICME)*. Toronto, Ontario, Kanada, Juli 2006.

[Rei07] Stephan Reiter, Björn Schuller und Gerhard Rigoll. "Hidden Conditional Random Fields for Meeting Segmentation". In *Proceedings of the International Conference on Multimedia and Expo (ICME)*. Beijing, China, Juli 2007.

[Ren02] Steve Renals. "The m4 project". Project leaflet, 2002.

[Ren05] Steve Renals. "AMI: Augmented Multiparty Interaction". In *Proceedings of the NIST Meeting Transcription Workshop*. Montreal, Kanada, 2005. AMI-10.

[Rie93] Martin Riedmiller und Heinrich Braun. "A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP algorithm". In *Proceedings of the International Conference on Neural Networks*, Seiten 586–591. San Francisco, Kalifornien, USA, 1993.

[Rig94] Gerhard Rigoll. *Neuronale Netze - Eine Einführung für Ingenieure, Informatiker und Naturwissenschaftler*. "Kontakt & Studium". Expert Verlag, 1994, 274 Seiten.

[Rig96a] Gerhard Rigoll, Andreas Kosmala, Jörg Rottland und Christoph Neukirchen. "A comparison between continuous and discrete density hidden Markov models for cursive handwriting recognition". In *Proceedings of the 13th International Conference on Pattern Recognition (ICPR)*, Band 2, Seiten 205–209. Wien, Österreich, August 1996.

[Rig96b] Gerhard Rigoll und Christoph Neukirchen. "A New Approach to Hybrid HMM/ANN Speech Recognition using Mutual Information Neural Networks." In *NIPS*, Seiten 772–778. 1996.

[Rig98a] Gerhard Rigoll, Andreas Kosmala und Stefan Eickeler. "High Performance Real-Time Gesture Recognition Using Hidden Markov Models". *Lecture Notes in Computer Science*, Band 1371, Seiten 69–81, 1998.

[Rig98b] Gerhard Rigoll und Daniel Willett. "A NN/HMM Hybrid for Continuous Speech Recognition with a Discriminant Nonlinear Feature Extraction". In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Seiten 9–12. Seattle, USA, 1998.

[Rig01a] Gerhard Rigoll. "Neuroinformatik 1". Skriptum zur Vorlesung, Gerhard Mercator Universität Duisburg, 2001.

[Rig01b] Gerhard Rigoll. "Neuroinformatik 2". Skriptum zur Vorlesung, Gerhard Mercator Universität Duisburg, 2001.

[Rig05] Gerhard Rigoll. "Pattern recognition". Skriptum zur Vorlesung, Technische Universität München, 2005.

[Rob94] Anthony J. Robinson. "An application of recurrent nets to phone probability estimation". *IEEE Transactions on Neural Networks*, Band 5, Nr. 2, Seiten 298–305, März 1994. doi:10.1109/72.279192.

[Rom01] Nicholas C. Romano und Jay F. Nunamaker. "Meeting Analysis: Findings from Research and Practice". In *Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS)*, Seite 1072. IEEE Computer Society, Washington, DC, USA, 2001. ISBN 0-7695-0981-9.

[Rot00] Jörg Rottland und Gerhard Rigoll. "Tied Posteriors: An Approach for Effective Introduction of Context Dependency in Hybrid NN/HMM LVCSR". In *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. Istambul, Türkei, Juni 2000.

[Rus02] Günther Ruske. "Mustererkennung in der Sprachverarbeitung". Skriptum zur Vorlesung, Technische Universität München, 2002.

[Sch78] Gideon Schwarz. "Estimating the Dimension of a Model". *Annals of Statistics*, Band 6, Nr. 2, Seiten 461–466, März 1978.

[Sch93] Jürgen Schmidhuber. "Netzwerkarchitekturen, Zielfunktionen und Kettenregel". Habilitationsschrift, Technische Universität München, April 1993.

[Sch96] Michael S. Schmidt. "Identifying Speakers With Support Vector Networks". In *Proceedings of Interface '96*. Sydney, Australien, Juli 1996.

[Sch97a] Robert J. Schalkoff. *Artificial neural networks*. McGraw-Hill, 1997, 422 Seiten.

[Sch97b] Richard Schwartz, Toru Imai, Francis Kubala, Long Nguyen und John Makhoul. "A Maximum Likelihood Model for Topic Classification of Broadcast News". In *Proceedings of EUROSPEECH*. Rhodos, Griechenland, September 1997.

[Sch01a] Tanja Schultz, Alex Waibel, Michael Bett, Florian Metze, Yue Pan, Klaus Ries, Thomas Schaaf, Hagen Soltau, Martin Westphal, Hua Yu und Klaus Zechner. "The ISL Meeting Room System". In *Proceedings of the Workshop on Hands-Free Speech Communication (HSC)*. Kyoto, Japan, April 2001.

[Sch01b] Bernhard Schölkopf. "Tutorial: SVM and Kernel methods". In *Proceedings of Neural Information Processing Systems (NIPS)*. Vancouver, Kanada, 2001.

[Sch04a] Carsten Schauer und Horst-Michael Groß. "Design and Optimization of Amari Neural Fields for Early Auditory-Visual Integration". In *In Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, Seiten 2523–2528. Budapest, Ungarn, 2004.

[Sch04b] Björn Schuller, Gerhard Rigoll und Manfred Lang. "Discrimination of Speech and Monophonic Singing in Continuous Audio Streams Applying Multi-Layer

Support Vector Machines". In *Proceedings of the International Conference on Multimedia and Expo (ICME)*. Taipei, Taiwan, Juni 2004.

[Sch06] Björn Schuller. *Automatische Emotionserkennung aus sprachlicher und manueller Interaktion*. Dissertation, Lehrstuhl für Mensch-Maschine-Kommunikation, Technische Universität München, Juni 2006.

[Sha48] Claude E. Shannon. "A mathematical theory of communication". *Bell System Technical Journal*, Band 27, Seiten 379–423 and 623–656, Juli and Oktober 1948.

[Sko03] Marios Skounakis, Mark Craven und Soumya Ray. "Hierarchical Hidden Markov Models for Information Extraction". In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*. Acapulco, Mexiko, 2003.

[Smi97] Steven W. Smith. *The Scientist and Engineer's Guide to Digital Signal Processing*. California Technical Publishing, San Diego, Kalifornien, USA, 1997, 626 Seiten.

[ST95] Ernst Günter Schukat-Talamazzini. *Automatische Spracherkennung*. Vieweg Verlag, 1995.

[Sta95] Thad Starner und Alex Pentland. "Visual Recognition of American Sign Language Using Hidden Markov Models". In M. Bichsel (Herausgeber), *Proceedings of the International Workshop Automated Face and Gesture Recognition*, Seiten 189–194. IEEE Press, Piscataway, N.J., Juni 1995.

[Sta03] Jan Stadermann und Gerhard Rigoll. "Comparing NN paradigms in hybrid NN/HMM speech recognition using tied posteriors". In *Workshop on Automatic Speech Recognition and Understanding (ASRU)*. St. Thomas, U. S. Virgin Islands, Dezember 2003. doi:10.1109/ASRU.2003.1318409.

[Sta06] Jan Robert Stadermann. *Automatische Spracherkennung mit hybriden akustischen Modellen*. Dissertation, Lehrstuhl für Mensch-Maschine-Kommunikation, Technische Universität München, 2006.

[Ste97] Axel Steinhage und Gregor Schöner. "Self-calibration based on invariant view recognition: Dynamic approach to navigation". *Robotics and Autonomous Systems*, Band 20, Seiten 133–156, 1997.

[Ste00a] Axel Steinhage. "The Dynamic Approach To Anthropomorphic Robotics". In *Proceedings of the Fourth Portuguese Conference on Automatic Control (CONTROLO)*. Guimarães, Portugal, Oktober 2000.

[Ste00b] Axel Steinhage. "Dynamic neural fields for robot control". In *Proceedings of the International Workshop On Dynamical Neural Networks and Applications (DYNN)*. Bielefeld, Deutschland, November 2000.

[Teb95] Joseph Michael Tebelskis. *Speech recognition using neural networks*. Dissertation, Carnegie Mellon University, Pittsburgh, PA, USA, 1995.

[Tri99] Alain Tritschler und Ramesh Gopinath. "Improved Speaker Segmentation and Segments Clustering using the Bayesian Information Criterion". In *In Proceedings of EUROSPEECH*, Seiten 679–682. Budapest, Ungarn, September 1999.

[Tuc05] Simon Tucker und Steve Whittaker. "Accessing Multimodal Meeting Data: Systems, Problems and Possibilities". In *Lecture Notes In Computer Science*, Band 3361, Seiten 1–11. Samy Bengio and Herve Bourlard, 2005.

[Vap98] Vladimir N. Vapnik. *Statistical learning theory*. Adaptive and learning systems for signal processing, communications and control. Wiley, New York, 1998, 736 Seiten.

[Vap00] Vladimir N. Vapnik. *The nature of statistical learning theory*. Statistics for engineering and information science. Springer, New York, 2nd edition Auflage, 2000, 314 Seiten.

[Vit67] Andrew J. Viterbi. "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm". *IEEE Transactions on Information Theory*, Band 13, Nr. 2, Seiten 260–269, April 1967.

[Vog05] Tim P. Vogels, Kanaka Rajan und L.F. Abbott. "Neural Network Dynamics". *Annual Review of Neuroscience*, Band 28, Seiten 357–376, Juli 2005. doi:10.1146/annurev.neuro.28.061604.135637.

[Wai01] Alex Waibel, Michael Bett, Florian Metze, Klaus Ries, Thomas Schaaf, Tanja Schultz, Hagen Soltau, Hua Yu und Klaus Zechner. "Advances in Automatic Meeting Record Creation and Access". In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing 2001*. Seattle, USA, Mai 2001.

[Wai03] Alex Waibel, Tanja Schultz, Michael Bett, Matthias Denecke, Robert Malkin, Ivica Rogina, Rainer Stiefelhagen und Jie Yang. "Smart: The Smart Meeting Room Task at ISL". In *In Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Band 4, Seiten 752–755. Hong Kong, April 2003.

[Wal99] Frederick Walls, Hubert Jin, Sreenivasa Sista und Richard Schwartz. "Topic Detection in Broadcast News". In *Proceedings of the DARPA Broadcast News Workshop*. Herndon, VA, USA, Februar 1999.

[Wal02] Hanna Wallach. *Efficient Training of Conditional Random Fields*. Diplomarbeit, Division of Informatics, University of Edinburgh, 2002.

[Wal04a] Hanna M. Wallach. "Conditional Random Fields: An Introduction". Technischer Bericht MS-CIS-04-21, Department of Computer and Information Science, University of Pennsylvania, 2004.

[Wal04b] Frank Wallhoff, Martin Zobl und Gerhard Rigoll. "Action Segmentation And Recognition in Meeting Room Scenarios". In *International Proceedings on International Conference on Image Processing (ICIP)*, Seiten 2223–2226. Singapur, Oktober 2004.

[Wal04c] Frank Wallhoff, Martin Zobl, Gerhard Rigoll und Igor Potucek. "Face Tracking in Meeting Room Scenarios Using Omnidirectional Views". In B. Werner (Herausgeber), *Proceedings of the 17th International Conference on Pattern Recognition (ICPR 2004)*, Band 4, Seiten 933–936. IEEE Computer Society, Cambridge, Großbritannien, August 2004.

[Wal06] Frank Wallhoff. *Entwicklung und Evaluierung neuartiger Verfahren zur automatischen Gesichtsdetektion, Identifikation und Emotionserkennung*. Dissertation, Lehrstuhl für Mensch-Maschine-Kommunikation, Technische Universität München, 2006.

[Wan06] Sy Bor Wang, Ariadna Quattoni, Louis-Philippe Morency, David Demirdjian und Trevor Darrell. "Hidden Conditional Random Fields for Gesture Recognition". In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. New York, NY, USA, Juni 2006.

[War94] Narada Warakagoda. *A Hybrid ANN-HMM ASR system with NN based adaptive preprocessing*. Diplomarbeit, Norwegian University of Science and Technology, Trondheim, 1994.

[Wel05] Pierre Wellner, Mike Flynn und Maël Guillemot. "Browsing Recorded Meetings with Ferret". In *Machine Learning for Multimodal Interaction*, Band 3361/2005 von *Lecture Notes in Computer Science*, Seiten 12–21. Springer, 2005.

[Wes98] Jason Weston und Chris Watkins. "Multi-class Support Vector Machines". Technischer Bericht, Department of Computer Science, Royal Holloway, University of London, 1998.

[Wil73] Hugh R. Wilson und Jack D. Cowan. "A Mathematical Theory of the Functional Dynamics of Cortical and Thalamic Nervous Tissue". *Kybernetik*, Band 13, Nr. 2, Seiten 55–80, 1973.

[Wil90] Ronald J. Williams und David Zipser. "Gradient-Based Learning Algorithms for Recurrent Connectionist Networks". In Y. Chauvin und D. E. Rumelhart (Herausgeber), *Backpropagation: Theory, Architectures, and Applications*. Erlbaum, Hillsdale, NJ, USA, 1990.

[Wu99] Ying Wu und Thomas S. Huang. "Vision-Based Gesture Recognition: A Review". In *Proceedings of the International Gesture Workshop*, Seiten 103–115. Springer, Gif-sur-Yvette, Frankreich, März 1999.

[Xu92] Lei Xu, Adam Krzyzak und Ching Y. Suen. "Methods of Combining Multiple Classifiers and Their Applications to Handwriting Recognition". *IEEE Transactions on Systems, Man and Cybernetics*, Band 22, Nr. 3, Seiten 418–435, Juni 1992.

[Yak70] Sidney J. Yakowitz. "Unsupervised learning and the identification of finite mixtures". *IEEE Transactions on Information Theory*, Band 16, Nr. 3, Seiten 330–338, Mai 1970.

[Zha04a] Dong Zhang, Daniel Gatica-Perez, Samy Bengio, Iain McCowan und Guillaume Lathoud. "Modeling Individual and Group Actions in Meetings: a Two-Layer HMM Framework". In *Second IEEE Workshop on Event Mining: Detection and Recognition of Events in Video, In Association with CVPR*. Washington, DC, USA, Juni 2004. IDIAP-RR 04-09.

[Zha04b] Dong Zhang, Daniel Gatica-Perez, Samy Bengio, Iain McCowan und Guillaume Lathoud. "Modeling Individual and Group Actions in Meetings With Layered HMMs". IDIAP-RR 33, IDIAP, Martigny, Schweiz, 2004. Submitted for publication.

[Zha06a] Dong Zhang. *Probabilistic Graphical Models for Human Interaction Analysis*. Dissertation, Ecole Polytechnique Federale de Lausanne, 2006.

[Zha06b] Dong Zhang, Daniel Gatica-Perez, Samy Bengio, Iain McCowan und Guillaume Lathoud. "Modeling Individual and Group Actions in Meetings With Layered HMMs". *IEEE Transactions on Multimedia*, Band 8, Nr. 3, Seiten 509–520, 2006.

[Zho00] Bowen Zhou und John H.L. Hansen. "Unsupervised Audio Stream Segmentation and Clustering via the Bayesian Information Criterion". In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, Band 3, Seiten 714–717. Beijing, China, Oktober 2000.

[Zob03] Martin Zobl, Frank Wallhoff und Gerhard Rigoll. "Action Recognition in Meeting Scenarios using Global Motion Features". In J. Ferryman (Herausgeber), *IEEE International Workshops on Performance Evaluation of Tracking and Surveillance (PETS-ICVS)*, Seiten 32–36. University of Reading, Großbritannien, 2003.