Lehrstuhl für Mensch-Maschine-Kommunikation
Technische Universität München

Automatische Spracherkennung mit hybriden akustischen Modellen

Dipl.-Ing. Jan Robert Stadermann

Vollständiger Abdruck der von der Fakultät für Elektrotechnik und Informationstechnik
der Technischen Universität München zur Erlangung des akademischen Grades eines Doktor-Ingenieurs
genehmigten Dissertation.

Vorsitzender:
Univ.-Prof. Dr.-Ing. Jörg Eberspächer
Prüfer der Dissertation:
1. Univ.-Prof. Dr.-Ing. habil. Gerhard Rigoll
2. Univ.-Prof. Dr.-Ing. Hermann Ney, Rheinisch-Westfälische Technische Hochschule Aachen

Die Dissertation wurde am 27. Oktober 2005 bei der Technischen Universität München eingereicht und durch die Fakultät für Elektrotechnik und Informationstechnik
am 16. November 2005 angenommen.

Vorwort

Die vorliegende Arbeit ist während einer gut fünfjährigen Tätigkeit als wissenschaftlicher Mitarbeiter am Lehrstuhl für Mensch-Maschine-Kommunikation der Fakultät für Elektro- und Informationstechnik der Technischen Universität München und im (ehemaligen) Fachgebiet für Technische Informatik des Institutes für Informationstechnik der Universität Duisburg-Essen entstanden. In dieser Zeit habe ich mich mit verschiedenen Aspekten der hybriden akustischen Modellierung für die automatische Spracherkennung beschäftigen können.

Mein besonderer Dank gilt dem Betreuer dieser Arbeit, Prof. Dr.-Ing. habil. Gerhard Rigoll, der mir einen Einblick in dieses interessante Forschungsthema sowohl in Duisburg als auch in München ermöglicht hat und dessen Anregungen einen fruchtbaren Rahmen für diese Arbeit geboten haben. Prof. Dr.-Ing. Hermann Ney von der RWTH Aachen (Lehrstuhl für Informatik VI) danke ich für die Übernahme des Koreferates und dem damit verbundenen Aufwand. Ferner danke ich den ehemaligen Kollegen aus Duisburg und den Mitarbeitern des Lehrstuhls für Mensch-Maschine-Kommunikation für ihre stets vorhandene Unterstützung und Diskussionsbereitschaft, insbesondere Dipl.-Ing. Björn Schuller und Dipl.-Ing. André Störmer für das Korrekturlesen dieser Arbeit. Abschließend möchte ich mich bei den von mir betreuten Studenten für ihre konstruktive Mitarbeit bedanken, speziell bei den Diplomanden Elmar Sommer und Miguel Morgado. Meiner Frau Melanie sowie meinen Eltern und Freunden danke ich für ihr Verständnis, in der Zeit der Anfertigung dieser Arbeit häufig nur geteilte Aufmerksamkeit erfahren zu haben.

Jan Stadermann

Inhaltsverzeichnis

1. Einleitung

Die Kommunikation zwischen Mensch und Maschine (MMK[1]) ist Gegenstand aktueller Forschung und gewinnt in der Gesellschaft zunehmend an Bedeutung. Während sich in der Vergangenheit der Mensch weitgehend an die Maschine anpassen mußte, ist es die heutige Aufgabe der Forschung, die Bedienung von Maschinen intuitiv und einfach zu gestalten. Dabei sollen bevorzugt die natürlichen Kommunikationskanäle des Menschen eingesetzt werden, ein Schwerpunkt liegt aufgrund der Menge an übertragbaren Informationen auf dem optischen und dem akustischen Kanal. Insbesondere das Nutzen von Sprache als natürliche Kommunikationsform des Menschen für die MMK erscheint als naheliegendes technisches Entwicklungsziel [Lang, 1994; Lang u. Stahl, 1994].
Nachdem kurz die theoretischen Hintergründe der automatischen Spracherkennung diskutiert werden, steht in dieser Arbeit die Entwicklung eines neuen akustischen Modells für einen solchen Erkenner im Zentrum. Nach einer kurzen, allgemeinen Einführung im Abschnitt 1.1 folgt ein Überblick über das akustische Modell (Abschnitt 1.2), sowie eine Vorstellung der Gliederung dieser Arbeit (Abschnitt 1.3).

1.1. Automatische Spracherkennung

Historisch hat die Spracherkennung mit der sprecherabhängigen Erkennung isolierter Einzelworte begonnen, solche Systeme basieren auf dem Mustervergleich zwischen dem Testmuster und abgespeicherten Referenzmustern. Durch Weiterentwicklung und durch die Anwendung neuer Methoden ist aktuell die Erkennung kompletter, spontan gesprochener, sprecherunabhängiger Äußerungen mit großem Wortschatz über einen Telefonkanal möglich [Evermann u. a., 2005]. Insbesondere die Einführung der statistischen Modellierung mit Hidden-Markov-Modellen (HMM) und die Möglichkeit der schnellen Verarbeitung großer Datenmengen durch moderne Rechnertechnik hat zu dieser Leistungssteigerung geführt.
Neben der Erkennung des Inhalts einer gesprochenen Äußerung stehen auch Eigenschaften des Sprechers im Fokus einer automatischen Verarbeitung. Dazu gehören die Sprecheridentifikation und -verifikation, sowie die Erkennung der bei der Äußerung ausgedrückten Emotionen. Insbesondere bei der Emotionserkennung kann sowohl die akustische Information, als auch der erkannte Text ausgewertet werden [Schuller u. a., 2005].
Eingesetzt wird die Spracherkennung neben den klassischen Diktieraufgaben vor allem in automatischen Dialogsystemen zur Bedienung von Telefonportalen oder in Infotainmentsystemen im Automobil. Hierbei ist neben der syntaktischen Erkennung der Worte

[1] MMK - Mensch-Maschine-Kommunikation

auch die semantische Erkennung des Inhalts (NLU[2]) notwendig. Möglich ist hierbei sowohl ein zweites, getrenntes System, als auch ein integrierter Ansatz [Thomae u. a., 2003]. Zur besseren Reaktion eines solchen Dialogsystems auf Nutzeranfragen kann sowohl die Identität des Sprechers, als auch sein Emotionszustand ausgewertet werden. Schließlich kann auf einer noch abstrakteren Ebene zum Beispiel die Zusammenfassung einer Besprechung automatisch erstellt werden, hierbei werden neben der akustischen Information auch Bildsignale mehrerer Videokameras ausgewertet [Reiter u. a., 2005]. Gemeinsam ist den beschriebenen Anwendungen, daß die Methoden zur Erkennung sich nur in Einzelheiten von den Methoden der akustischen Modellierung, wie sie in dieser Arbeit beschrieben werden, unterscheiden. Obwohl die im Folgenden beschriebenen Ansätze in dieser Arbeit nur für die akustische Modellierung eingesetzt werden, sind sie auch für die weiteren, vorgestellten Anwendungen ohne große Änderungen nutzbar.

1.2. Hybride Modellierung

Die Bezeichnung *hybride Modellierung* umfaßt eine Vielzahl von Kombinationsmöglichkeiten von HMM und anderen Klassifikatoren, in dieser Arbeit liegt der Schwerpunkt auf der Berechnung von Symbolauftrittswahrscheinlichkeiten mit diskriminativen Klassifikatoren, kombiniert mit HMM zur Modellierung der zeitlichen Abfolge. Generelle Nachteile der Modellierung mit HMM sind beim meist verwendeten Baum-Welch-Trainingsverfahren [Baum, 1972] eine unabhängige Optimierung der einzelnen Modelle während der Trainingsphase, sowie eine mangelnde Berücksichtigung des zeitlichen Kontextes. Ein neuronaler Klassifikator hat die genannten Einschränkungen nicht, er trainiert diskriminativ, jeder Ausgang wird im Verhältnis zu allen anderen optimiert und die Berücksichtigung von Kontext ist relativ einfach zu realisieren. Obwohl es einige Ansätze gibt [Hild u. Waibel, 1993], sind diese Klassifikatoren allein jedoch nur ungenügend in der Lage, Sequenzen unterschiedlicher Länge zu klassifizieren, so daß die Kombination mit HMM als *hybrides* Modell eine natürliche Erweiterung darstellt. Ein erfolgreicher Ansatz zur Klassifikation mit HMM und neuronalen Netzen [Bourlard u. Morgan, 1994] wird im Rahmen dieser Arbeit durch Einführung von verbundenen Auftrittswahrscheinlichkeiten erweitert. Vorteile dieser Erweiterung sind eine einfach zu realisierende kontextabhängige Modellierung, sowie die Unabhängigkeit von Klassifikatorausgängen und HMM-Topologie. Hierdurch kann die hybride akustische Modellierung auch im Umfeld der verteilten Spracherkennung flexibel eingesetzt werden. Die Flexibilität hinsichtlich der Klassifikatorauslegung wird durch unterschiedliche Klassifikationsalgorithmen und unterschiedliche Vorverarbeitungen demonstriert. Außerdem bieten sich durch die Erweiterung neue Möglichkeiten der Adaption dieser hybriden Modelle, von denen einige in dieser Arbeit am Beispiel der Sprecheradaption vorgestellt werden.

[2]NLU - *engl.*: Natural Language Understanding

1.3. Gliederung

Den Aufbau des Spracherkenners, wie er in der vorliegenden Arbeit verwendet wird, zeigt Bild 1.3.1. In den folgenden Kapiteln werden die abgebildeten Blöcke vorgestellt

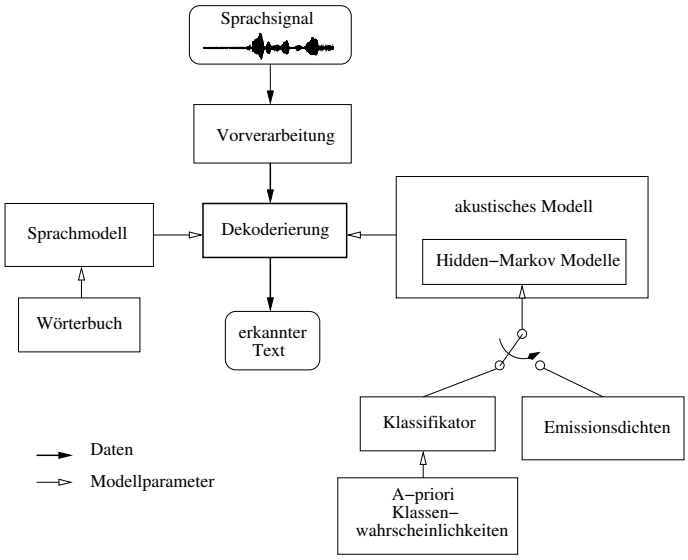


Abbildung 1.3.1.: Blockdiagramm eines automatischen Spracherkenners

und detailliert erläutert: Kapitel 2 behandelt die Vorverarbeitung des Sprachsignals, die notwendig ist, um die informationstragenden Anteile des Signals zu extrahieren. Kapitel 3 behandelt statische Klassifikatoren als Teil der hybriden Modelle, die die extrahierten Informationen in Symbolklassen einteilen. Aus der großen Menge an möglichen Klassifikatoren werden neuronale Netze unterschiedlicher Topologien und Support-Vektor-Maschinen mit zugehörigen Algorithmen zum Training im Detail vorgestellt. Abgerundet wird das Kapitel durch einen Vergleich dieser Verfahren bei der Klassifikation akustischer Einheiten. Hidden-Markov-Modelle mit Emissionsdichten sind Gegenstand von Kapitel 4. Neben den Grundlagen und dem Trainingsalgorithmus nach Baum-Welch werden auch die Sprachmodellierung und die Dekodierung kurz angerissen, sowie Ergebnisse präsentiert. In Kapitel 5 wird der Klassifikator mit den HMM zum hybriden Modell verbunden. Es werden verschiedene Kombinationsmöglichkeiten diskutiert und die erzielten Ergebnisse mit den Resultaten aus Kapitel 4 verglichen. Die beiden darauffolgenden Kapitel zeigen Erweiterungsmöglichkeiten zu den hybriden Modellen: Kapitel 6 zeigt Möglichkeiten zur Adaption der Modellparameter und Kapitel 7 beschreibt eine Anwendungsmöglichkeit der hybriden akustische Modelle für die verteilte Spracherkennung.
Zur Auswertung der untersuchten Modelle werden die Wallstreet-Journal Datenbasis (WSJ), sowie die AURORA2 Datenbasis herangezogen. Beide Datenbasen sind im Anhang B im Detail beschrieben. Abgeschlossen wird die Arbeit durch eine Zusammenfassung und einen Ausblick in Kapitel 8.

2. Vorverarbeitung des Sprachsignals

Der Vorverarbeitung kommt die sehr wichtige Aufgabe der Extraktion der linguistischen Information aus dem akustischen Signal im Spracherkennungssystem zu. Im Folgenden sei angenommen, daß das Sprachsignal in abgetasteter und quantisierter Form im Rechner vorliegt. Typische Abtastfrequenzen sind 8kHz für Telefonsprache und 16kHz für Desktop-Anwendungen. Beim analogen Telefonkanal ist 4kHz technisch bedingt die obere Grenzfrequenz, eine Abtastfrequenz von 8kHz erfüllt das Abtasttheorem und erfaßt also alle verfügbaren Informationen. Erlaubt der Kanal eine größere Bandbreite, hat sich

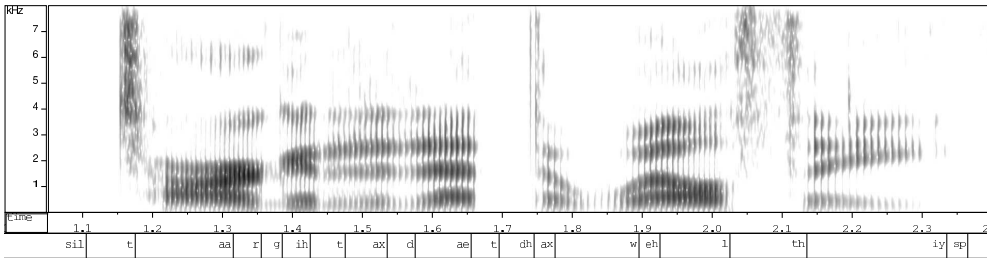
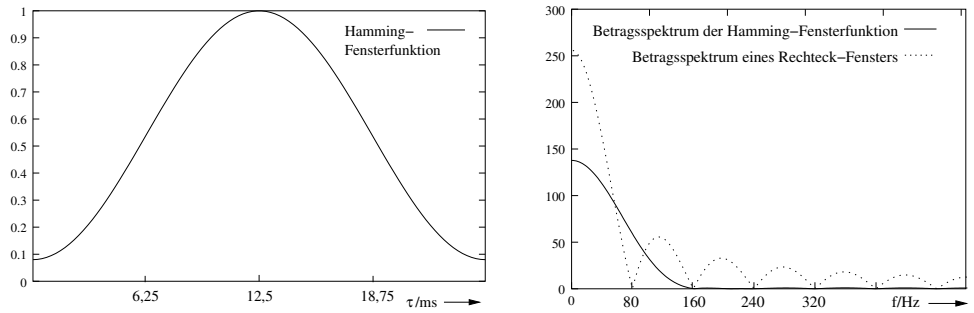


Abbildung 2.0.1.: Spektrogramm der Wörter *targeted at the wealthy* mit eingetragener Phonemsegmentierung, Abtastfrequenz: 16 kHz

eine Abtastfrequenz von 16 kHz als guter Kompromiß herausgestellt um einerseits alle relevanten Informationen aus dem Sprachsignal extrahieren zu können und andererseits die Datenmenge nicht unnötig zu vergrößern. Quantisiert wird das Sprachsignal üblicherweise mit einer Auflösung von 16 bit. Abbildung 2.0.1 zeigt das Spektrogramm eines mit 16 kHz und 16 bit abgetasteten Sprachsignals mit zugehöriger Phonem-Segmentierung (s. Kapitel 4).

2.1. Allgemeine Vorverarbeitung

Sprachsignale sind aus ihrer Natur heraus nicht-stationäre Signale, dies stellt insbesondere für die Modellierung mit Markov-Modellen (s. Kapitel 4) ein Problem dar, da Markov-Prozesse als stationär vorausgesetzt werden. Um trotzdem die Markov-Modellierung nutzen zu können, wird das Sprachsignal nur während eines kurzen Zeitfensters betrachtet [Ruske, 1988]. Die Dauer des Fensters T_F ist dabei so bemessen, das eine Quasi-Stationarität innerhalb des Ausschnitts erfüllt ist (T_F sollte im Bereich zwischen 10ms und 30ms liegen). Start- und Endpunkt des Zeitfensters werden dann so lange weitergeschoben, bis das Signal komplett abgearbeitet ist. Ein guter Kompromiß zwischen



(a) Zeitsignal (b) Betragsspektrum

Abbildung 2.1.1.: Hamming-Fensterfunktion ($T_F = 25ms$) und Betragsspektrum eines Rechteckfensters zum Vergleich

Datenmenge und Zeitauflösung wird erzielt, wenn sich die einzelnen Fenster um mindestens 50% überlappen. Die Fensterung mit einem Rechteckfenster (dies entspricht dem einfachen Ausschneiden der Signalwerte) erzwingt im Spektralbereich die Faltung des Signalspektrums mit einer si-Funktion, was eine nicht vernachlässigbare Verzerrung zur Folge hat. Um dem zu begegnen, verwendet man Fensterfunktionen, die ein günstigeres Spektrum mit gedämpften Nebenmaxima besitzen. Aus der Auswahl der in Frage kommenden Funktionen [Deller u. a., 1993] wird in dieser Arbeit ausschließlich die Hamming-Fensterfunktion $w(\tau) = 0,54 + 0,46\cos(\frac{2\pi\tau}{T_F})$ verwendet. In Abbildung 2.1.1 ist das (kontinuierliche) Zeitsignal des Hammingfensters und das entsprechende Betragsspektrum dargestellt. Zum Vergleich ist zusätzlich das Betragsspektrum eines Rechteckfensters gleicher Länge angegeben, der deutlich größere Nebenmaxima im Frequenzverlauf aufweist. Einer Verfälschung des Zeitsignals durch die Fensterfunktion an den Rändern der Fenster wird bereits durch die Überlappung der Fenster begegnet. Abbildung 2.1.2 zeigt die vollständige Kette einer allgemeinen Vorverarbeitung für Sprachsignale. Wie dort zu sehen, findet vor der Fensterung noch eine Pre-Emphase statt. Die Pre-Emphase wird im allgemeinen als Hochpaß 1. Ordnung nach Gleichung (2.1.1) realisiert und dient zum Hervorheben höherer Frequenzen, die zur Lautunterscheidung insbesondere von stimmlosen Lauten bedeutungstragend sind. Für das Spektrum im z-Bereich $S(z)$ bzw. für das Zeitsignal $s(\tau)$ ergibt sich unter Verwendung des Eingangssignals $s'(\tau) \circ\!\!-\!\!\bullet\ S'(z)$

$$S(z) = S'(z)(1 - \alpha z^{-1}) \circ\!\!-\!\!\bullet\ s(\tau) = s'(\tau) - \alpha s'(\tau - 1) \quad \text{mit } \alpha = 0.97 \tag{2.1.1}$$

($s'(\tau)$ ist das ungefilterte Zeitsignal) Im Rahmen aller Experimente dieser Arbeit wird die Fensterlänge zu $T_F = 25ms$ gewählt. Die Verschiebung zwischen zwei Fenstern beträgt $T_V = 10ms$.

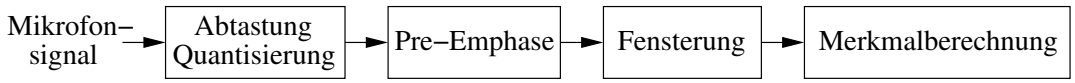


Abbildung 2.1.2.: Arbeitsschritte zur Signalvorverarbeitung

2.2. Merkmalberechnung

Die Algorithmen der folgenden Unterabschnitte sind Realisierungen für die Merkmalberechnung aus Abb. 2.1.2. Das Ergebnis der Berechnung ist ein Merkmalsvektor für jedes Fenster des Sprachsignals.

2.2.1. Mel-Cepstrum

Die *Mel*-Skala transformiert, ebenso wie die *Bark*-Skala, die Frequenz in eine der akustischen Wahrnehmung angepaßte Skala. Die Umrechnung von der Frequenzachse zur *Mel*-Skala zeigt Gl. (2.2.1), zur Umrechnung in die Bark-Skala s. Gl. (2.2.5).

$$\frac{f}{\text{Mel}} = 2595\text{Mel} \log_{10}\left(1 + \frac{f}{700\text{Hz}}\right) \tag{2.2.1}$$

Auf Basis der Mel- oder Bark-Skala kann nun eine Filterbank bestimmt werden, deren Mittenfrequenzen linear auf diesen Skalen angeordnet sind und deren Bandbreiten den Frequenzgruppen-Bandbreiten an dieser Stelle entsprechen [Zwicker u. Terhardt, 1980], um eine gehörgerechte Filterung des Signals zu ermöglichen. Im Frequenzbereich ergibt sich eine nichtlineare Verzerrung der Mittenfrequenzen und Bandbreiten, wie Abbildung 2.2.1 zeigt. Das Ziel der anschließenden Berechnung des Cepstrums, einer Form der ho-

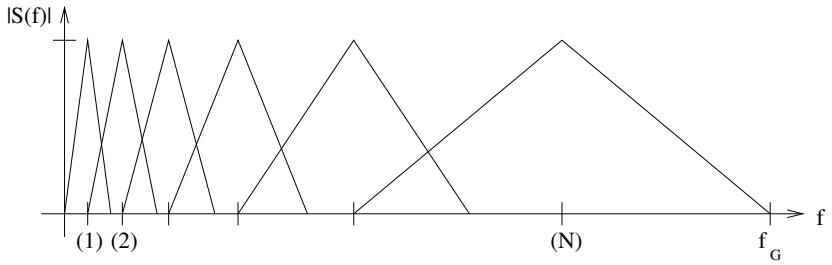


Abbildung 2.2.1.: Mittenfrequenzen und Bandbreiten der Mel-Filterbank

momorphen Analyse, ist die Trennung von Anregung und Vokaltrakt, die im Zeitbereich durch eine Faltungsoperation verknüpft sind [Eppinger u. Herter, 1993]. Zur Berechnung des *Mel-Frequenz-Cepstrums*(MFCC[1]) sind also folgende Schritte notwendig:

- Berechnung des Kurzzeit-Leistungsspektrums eines Fensters

[1]MFCC - *engl.*: mel-frequency cepstral coefficients

- Filterung des Spektrums mit einer Mel-Filterbank nach Abb. 2.2.1
- Logarithmieren des Mel-skalierten Leistungsspektrums
- Inverse diskrete Cosinus Transformation des logarithmierten Spektrums als Äquivalent zur Rücktransformation $c_n = \sqrt{\frac{2}{M}} \sum_{k=1}^{M} \log(m_k) \cos\left(\frac{\pi n}{M}(k-0.5)\right)$, wobei c_n den n-ten Cepstrumkoeffizienten, m_k das Ergebnis des k-ten Mel-Filters und M die Gesamtzahl an Mel-Filtern bezeichnet.

Abbildung 2.2.2 zeigt die Cepstralkoeffizienten c_1, c_2 und den logarithmierten Energieverlauf eines Satzausschnittes. Weitgehend sprecherunabhängige Komponenten des

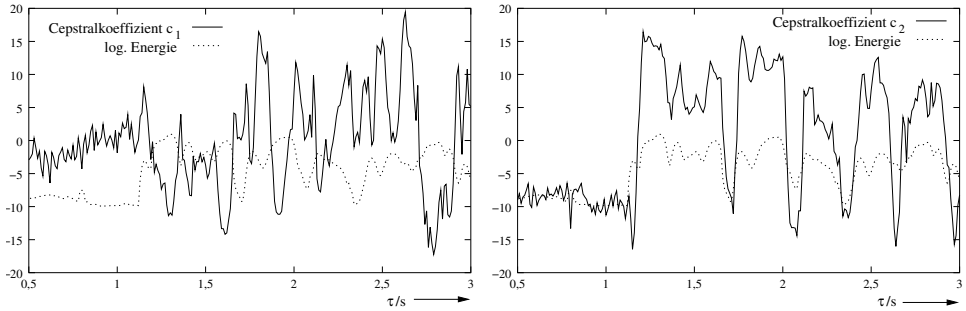


Abbildung 2.2.2.: Cepstralkoeffizienten c_1 und c_2, sowie logarithmierte Kurzzeitenergie des Satzausschnittes *targeted at the wealthy*

Signals, die durch Formung des Vokaltraktes entstanden sind, finden sich in den niederwertigen Cepstralkoeffizienten wieder, während sich die sprecherspezifischen Anteile, wie die Grundfrequenz in den höherwertigen Koeffizienten enthalten sind. In der Sprachverarbeitung hat sich die Berechnung des Mel-Cepstrums und die anschließende Verwendung der Cepstralkoeffizienten c_1 bis c_{12} als Quasi-Standard etabliert [Picone, 1993]. Abbildung 2.2.3 zeigt das Schema der Merkmalberechnung mit üblichen Dimensionen der Datenvektoren. Die MFCC-Merkmale sind in der Praxis sehr anfällig für Störungen

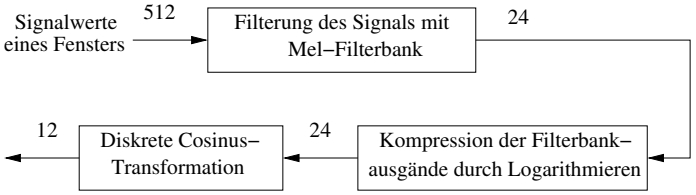


Abbildung 2.2.3.: MFCC-Berechnung, die Zahlenwerte geben jeweils die Dimension des Datenvektors an

des Kanals und für Hintergrundgeräusche. Die Robustheit gegenüber Kanaländerungen kann durch Subtraktion der cepstralen zeitlichen Mittelwerte vom Merkmalsvektor erreicht werden [Atal, 1974]. Die Annahme hierbei ist, daß das Eingangssignal sich aus

einer Faltung des Sprachsignals mit der Kanalimpulsantwort zusammensetzt. Im Frequenzbereich ergibt sich dann eine Multiplikation aus Spektrum des Sprachsignals und Spektrum der Kanalimpulsantwort. Durch Logarithmieren und Rücktransformieren entsteht also

$$c_X(k) = c_S(k) + c_U(k) \tag{2.2.2}$$

wobei $c_X(k)$ das Cepstrum des verzerrten Sprachsignals bezeichnet, $c_S(k)$ ist das Cepstrum des Sprachsignals und $c_U(k)$ das Cepstrum des Kanals. Wird nun eine zeitliche Mittelung jedes einzelnen Koeffizienten über alle Fenster des Signals durchgeführt und ferner angenommen, daß die Cepstralkoeffizienten $c_U(k)$ des Kanals zeitunabhängig sind [Westphal, 1997], erhält man

$$\bar{c_X}(k) = \bar{c_S}(k) + c_U(k) \tag{2.2.3}$$

Durch Subtraktion des cepstralen Mittelwertes $\bar{c_X}(k)$ von $c_X(k)$ aus Gl. (2.2.2) ergeben sich neue Merkmale $\tilde{c_X}(k)$:

$$\tilde{c_X}(k) = c_X(k) - \bar{c_X}(k) = c_S(k) + c_U(k) - \bar{c_S}(k) - c_U(k) = c_S(k) - \bar{c_S}(k) \tag{2.2.4}$$

Unter der Annahme eines zeitlich unveränderlichen Kanals kann dessen Einfluß also durch Subtraktion des Mittelwertes herausgerechnet werden. Nachteil dieses Verfahrens ist, daß für eine gute Schätzung des Mittelwertes das gesamte zu untersuchende Signal vorliegen muß. In der Praxis behilft man sich entweder durch einen gleitenden Mittelwert, der ständig aktualisiert wird, oder eine komplette Äußerung wird abgewartet, bevor die Erkennung beginnt.

2.2.2. Perceptual Linear Prediction

Perceptual Linear Prediction (PLP) nach [Hermansky, 1990] berechnet Merkmale, die dem Mel-Cepstrum sehr ähnlich sind. Hauptunterschiede sind zum einen eine etwas andere Filterbank zum anderen wird statt des Logarithmierens die kubische Wurzel zur Kompression des Spektrums verwendet. Die Motivation zur Berechnung von PLP Koeffizienten ist die genauere Nachbildung von Eigenschaften des Gehörs. Das Schema der Merkmalberechnung ist Abbildung 2.2.4 zu entnehmen. Die Einteilung der Frequenz-

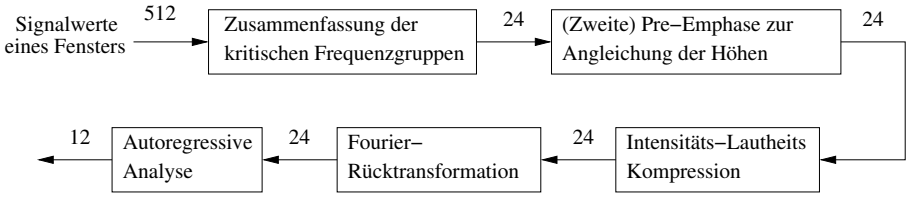


Abbildung 2.2.4.: PLP-Berechnung, die Zahlenwerte geben jeweils die Dimension des Datenvektors an

gruppen erfolgt entlang der *Bark*-Skala in äquidistanten Abschnitten. Die Transformation in die Bark-Skala zeigt Gl. (2.2.5)

$$\frac{\Omega(\omega)}{\text{Bark}} = 6\,\text{Bark}\ \log\left(\frac{\omega/\frac{1}{s}}{1200\pi} + \sqrt{\frac{\omega/\frac{1}{s}}{1200\pi} + 1}\right) \tag{2.2.5}$$

wobei $\omega = 2\pi f$ die Kreisfrequenz und Ω den transformierten Wert in *Bark* bezeichnen. Die anschließende Faltung mit einer Frequenzgruppen-Maskierungskurve (siehe Abb. 2.2.5) ist ähnlich der Filterung mit einem Dreiecksfilter der Mel-Filterbank (Abschnitt 2.2.1), berücksichtigt aber genauer spektrale Verdeckungseffekte des menschlichen Gehörs. Zusammen mit der anschließenden Höhenanhebung und der Intensitäts-

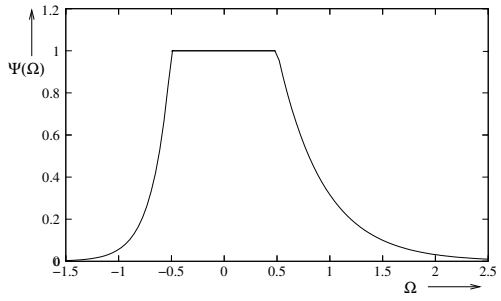


Abbildung 2.2.5.: Maskierungskurve für die Frequenzgruppenanalyse (PLP)

Lautheits-Kompression (hier realisiert durch die kubische Wurzel) können diese Operationen in eine PLP-Filterbank umgerechnet werden, die ähnlich der Mel-Filterbank in einer Rechenoperation auf das Signal angewandt wird. Nach der anschließenden Rücktransformation erhält man die Autokorrelationsfunktion, von der dann N autoregressive Koeffizienten mittels Levinson-Durbin Rekursion [Alexander, 1986] gewonnen werden. Obwohl die PLP-Merkmale eigenständig benutzt werden können, werden im Rahmen dieser Arbeit die Verarbeitungsschritte von PLP Merkmalen nur in Kombination mit einem RASTA-Filter (siehe nächster Abschnitt) eingesetzt.

2.2.3. RASTA-PLP

Die Berechnung von RASTA Merkmalen [Hermansky u. Morgan, 1994] stellt eine Erweiterung der PLP-Merkmale (Abschnitt 2.2.2) dar und beruht auf relativen Spektraländerungen. Grundidee ist dabei, alle Änderungen im Spektrum, deren Größenordnungen sich außerhalb der Änderungen durch die Sprache selbst bewegen, durch ein Filter abzuschwächen. Insbesondere Änderungen am Kanal (zwischen Trainings- und Testbedingungen) lassen sich hierdurch unterdrücken. Aber auch additive Geräusche können durch eine Erweiterung (*J-RASTA*) kompensiert werden. Abbildung 2.2.6 zeigt die Verarbeitungsschritte, wobei die schattierten Module bereits aus der Berechnung der PLP-Merkmale bekannt sind. Die Kompressionskennlinie (3. Block der Verarbeitungskette)

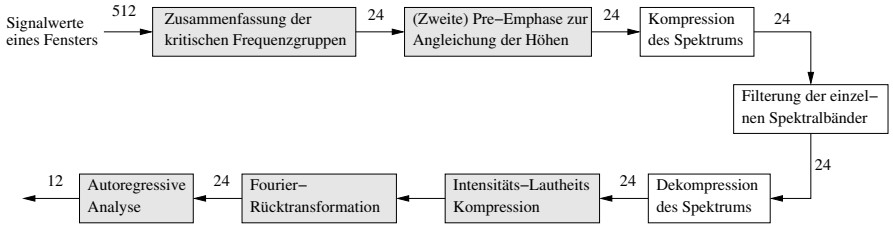


Abbildung 2.2.6.: Berechnung von RASTA-Merkmalen, die Zahlenwerte geben jeweils die Dimension des Datenvektors an

ist dabei durch

$$y = \log(1 + \mathfrak{J}x) \tag{2.2.6}$$

gegeben, wobei y das komprimierte Merkmal, x das Ergebnis der PLP-Filterbank und $\mathfrak{J}$ ein Parameter des Kompressors darstellen. Die Kompression hat auch hier eine "Entfaltung" des Kanals zum Ziel, in [Hirsch u. a., 1991] wird zusätzlich festgestellt, daß eine Filterung im Spektralbereich verstärkte Robustheit gegenüber additiven Störungen erzeugt. Daher weist der Kompressor nach Gl. (2.2.6) durch den Parameter $\mathfrak{J}$ für tiefe Frequenzen eher lineares Verhalten auf (y ist weiterhin im Spektralbereich) und für hohe Frequenzen eher logarithmisches Verhalten (y wird komprimiert). Das verwendete RASTA-Filter (4. Verarbeitungsblock aus Bild 2.2.6) ist ein Bandpaß mit folgender Charakteristik:

$$H(z) = 0,1z^4 \, \frac{2 + z^{-1} - z^{-3} - 2z^{-4}}{1 + 0,94z^{-1}} \tag{2.2.7}$$

2.3. Nachverarbeitung der Merkmale

Zusätzliche Verbesserungen der Merkmalsvektoren können durch Hinzunahme der Kurzzeitenergie eines Fensters gewonnen werden [Furui, 1986], zur besseren Handhabung des Wertebereichs wird die Energie e vor der Weiterverarbeitung logarithmiert:

$$e = \log \sum_{\tau=1}^{T_F} s(\tau + t)^2 \tag{2.3.1}$$

Weiterhin ist es nach [Furui, 1986] zweckmäßig, Regressionskoeffizienten aus den Merkmalsvektoren zu berechnen und an diese anzufügen. In der verwendeten Implementierung beschreiben die Regressionskoeffizienten die Änderung der Merkmale über der Zeit:

$$f_{\Delta n} = \frac{\sum_{p=1}^{P} p \cdot f_i(\tau + p) - f_i(\tau - p)}{2 \sum_{p=1}^{P} p^2} \tag{2.3.2}$$

Die Merkmale des Regressions-Vektors $\vec{f}_\Delta = (f_{\Delta 1}, \ldots, f_{\Delta N})^T$ werden aufgrund ihres Zusammenhangs mit der ersten diskreten Ableitung als *Delta*-Werte bezeichnet. Wendet man Gl. (2.3.2) auf den Vektor $\vec{f}_\Delta$ an, so ergibt sich die zweite diskrete Zeitableitung der

Merkmale, Komponenten des Vektors $\vec{f}_{\Delta\Delta}$ werden daher Beschleunigungs- oder *Delta-Delta*-Merkmale genannt. Alle Merkmalsvektoren dieser Arbeit werden aufgrund der beschriebenen Vorteile [Furui, 1986; Rottland, 2000] um dynamische Merkmale ergänzt. Bei Verwendung der MFCC und der Kurzzeitenergie ergibt sich also mit $\vec{f}_{MFCC+e}(t) = (c_1, \ldots, c_{12}, e)^T$ der Merkmalsvektor für ein Fenster zu

$$\vec{f}(t) = \left(\vec{f}_{MFCC+e}, \vec{f}_{\Delta}, \vec{f}_{\Delta\Delta}\right)^T \tag{2.3.3}$$

3. Statische Klassifikatoren

In diesem Kapitel werden zwei leistungsfähige Algorithmen der Mustererkennung, Neuronale Netze und Support-Vektor-Maschinen, vorgestellt und zur Klassifikation von akustischen Daten eingesetzt. Obwohl die Merkmalsvektoren, die aus den Daten errechnet werden, eine zeitliche Reihenfolge haben, findet die eigentliche Klassifikation statisch statt, da alle Vektoren die gleiche Länge aufweisen und aus Signalstücken gleicher Länge entstanden sind.

3.1. Neuronale Netzwerke

Neuronale Netze (NN) ist im Bereich der Mustererkennung ein Übergriff für eine Klasse biologisch motivierter Netzwerke. Das biologische Neuron ist Teil des Gehirns, es ist dort der Grundbaustein der Informationsverarbeitung. Das menschlichen Gehirn besteht aus ungefähr 10^{10} bis 10^{11} Neuronen, die untereinander mit je 10^3 bis 10^4 Verbindungen verknüpft sind. Biologisch ausgeführt sind diese Verbindungen als Synapsen, die mittels elektrischer Potenziale und Potenzialänderungen untereinander kommunizieren. Der ganze Ablauf dieser Kommunikation läuft nicht-linear durch Aufsummation von über die Synapsen einlaufenden Potenzialimpulsen im Zellkern ab. Werden bestimmte Schwellen durch diese Summation überschritten, so beginnen auch die empfangenden Neuronen ihrerseits Impulse auszusenden. Die Lernfähigkeit der biologischen Neuronen ergibt sich durch Veränderung der Verstärkungsfaktoren für einlaufende und ausgehende Impulse [Rigoll, 1994c].
Mathematisch abstrahiert berechnet die kleinste Einheit des mathematischen Netzwerkes, das Neuron, eine nichtlineare Abbildung $\vec{x} \rightarrow y = f(\vec{x})$. Durch die Kombination der Neuronen in bestimmten Strukturen entsteht eine mehrdimensionale Abbildung $\vec{x} \rightarrow \vec{y}$. Die folgenden Unterkapitel stellen zwei spezielle Netztopologien vor, die besonders zur Klassifikation mehrerer Klassen bei einer großen Menge an Daten geeignet sind.

3.1.1. Das Multi-Layer-Perzeptron

Das *Multi-Layer-Perzeptron* (MLP) ist ein mathematischer Algorithmus zur Klassifikation nichtlinearer Probleme. Grundelement des MLP ist das Perzeptron, welches die gewichtete Summe seiner Eingänge berechnet und das Ergebnis mittels einer nichtlinearen Schwellenfunktion $F(.)$ binär klassifiziert. Wird der Eingangsvektor des Perzeptrons

mit $\vec{x} = (x_1, \ldots, x_l, \ldots, x_{L-1})^T$ bezeichnet, so ergibt sich mit den Netzparametern w_l für die gewichtete Summe $\xi = \sum_{l=1}^{L-1} w_l x_l + w_0$ und an seinem Ausgang

$$y = F(\xi) = F\left(\sum_{l=1}^{L-1} w_l x_l + w_0\right) \tag{3.1.1}$$

Der Wert w_0 wird als *bias* bezeichnet und regelt die Verschiebung der Klassifikations-Hyperebene aus dem Nullpunkt.
Ist die Funktion $F(.)$ zum Beispiel als Vorzeichenfunktion sgn(.) implementiert, so legt ein Perzeptron also eine Hyperebene in den Eingangsraum, wobei der jeweilige Halbraum die Klasse bestimmt. Ein Trainingsverfahren, sowie weitere Einzelheiten können [Duda u. Hart, 1973] entnommen werden.
Verwendet man nun mehrere Perzeptrons parallel, so kann jedem Perzeptron eine Klasse

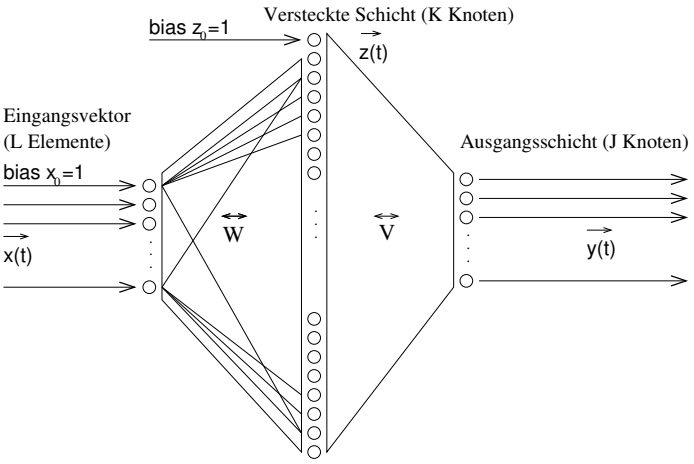


Abbildung 3.1.1.: MLP mit einer versteckten Schicht

zugeordnet werden und - bei Verwendung einer Nichtlinearität F mit kontinuierlichem Wertebereich - eine Klassifikation per Maximumentscheid durchgeführt werden.
Bei Einführung mehrerer Schichten aus jeweils mehreren Perzeptrons, wobei eine nachfolgende Schicht die Ausgangswerte der vorangegangenen Schicht als Eingang verwendet, gelangt man schließlich zum MLP, wie es in Bild 3.1.1 mit zwei Schichten gezeigt ist. Der Ausgangsvektor ergibt sich dann ausgehend von Gl. (3.1.1) zu

$$\vec{y} = \begin{pmatrix} y_1 \\ \vdots \\ y_J \end{pmatrix} = \vec{F}_o\left(\mathbf{V}^T \vec{z}\right) = \vec{F}_o\left(\mathbf{V}^T\, \vec{F}_h\left(\mathbf{W}^T \vec{x}\right)\right) \tag{3.1.2}$$

Die Matrix $\mathbf{V}$ mit dem Ergebnisvektor $\vec{z}$ wird als *versteckte* Schicht bezeichnet, da die Resultate dieser Schicht am Ausgang des Netzes nicht sichtbar sind. Daraus folgend bezeichnet man die Matrix $\mathbf{W}$ mit dem Ergebnisvektor $\vec{y}$ als Ausgangsschicht.

Zur Vereinfachung der Notation werden alle *bias*-Werte zusammengefaßt und in die Matrizen $\mathbf{V}$ und $\mathbf{W}$ integriert. Die Eingangsvektoren $\vec{x}$ bzw. $\vec{z}$ der Matrizen werden dann einfach um eine zusätzliche Komponente mit dem Wert 1 erweitert. $\vec{F}_o(.)$ und $\vec{F}_h(.)$ deuten hierbei an, daß für die einzelnen Schichten unterschiedliche Nichtlinearitäten verwendet werden können.
Für das Training eines solchen Netzes ist es notwendig, daß die Funktionen $F_o(.)$ und $F_h(.)$ differenzierbar sind. Eine gebräuchliche Wahl unter diesen Bedingungen ist die *Sigmoid*-Funktion, hier dargestellt für die versteckte Schicht ($\zeta_k = \sum_{l=0}^{L-1} w_{lk} x_l$ bezeichnet den Ausgang eines versteckten Neurons vor der Anwendung der Nichtlinearität):

$$F_h(\zeta_k) = \frac{1}{1 + \exp(-\zeta_k)} \tag{3.1.3}$$

In [Blum u. Li, 1991] wird gezeigt, daß ein solches Netz mit zwei Schichten und der Berechnungsvorschrift nach Gl. (3.1.2) in der Lage ist, beliebige funktionale Zusammenhänge zu approximieren, sofern die versteckte Schicht ausreichend viele Knoten enthält. Aus diesem Grund sind die MLPs in dieser Arbeit auf die Struktur aus Bild 3.1.1 festgelegt.
Ferner ist es nach [Bourlard u. Morgan, 1994] möglich, mit einem solchen Netz und einem geeigneten Trainingskriterium *a posteriori*-Klassenauftrittswahrscheinlichkeiten $Pr(\rho_j|\vec{x})$ für die Klasse ρ_j gegeben den Netzeingang $\vec{x}$ zu schätzen. Die Schätzwerte dieser Wahrscheinlichkeiten sollten den Stochastizitätsbedingungen $0 \leq Pr(\rho_j|\vec{x}) \leq 1$ und $\sum_{j=1}^{J} Pr(\rho_j|\vec{x}) = 1$ genügen. Um am NN-Ausgang diese Anforderungen zu erfüllen, ist die Sigmoid-Funktion wegen der letztgenannten Anforderung nicht geeignet - hier bietet sich die *Softmax*-Funktion F_o an, die eine Normierung des Ausgangsvektors bewirkt (ξ_j ist das Ergebnis eines Ausgangsneurons vor der Anwendung der Nichtlinearität):

$$F_o(\xi_j) = \frac{\exp(\xi_j)}{\sum_{n=1}^{J} \exp(\xi_n)} \tag{3.1.4}$$

Trainingsverfahren

In diesem Abschnitt wird ein weitverbreitetes Trainingsverfahren für MLPs vorgestellt, mit dem Netz Klassenauftrittswahrscheinlichkeiten geschätzt werden können: die *allgemeine Delta-Regel*, auch bekannt als *back propagation* Algorithmus. Grundlagen und Erweiterungen dieses Verfahrens sind zum Beispiel in [Schalkoff, 1994] ausführlich erläutert.
Die allgemeine Delta-Regel beruht auf der Minimierung einer Optimierungsfunktion $E(.)$ durch Gradientenabstieg, die einzelnen Schritte können wie folgt zusammengefaßt werden:

1. Initialisierung des Netzes
2. Anlegen der Trainingsdaten $\vec{x}(t)$ an das Netz und berechnen der Ausgangsvektoren $\vec{z}(t)$ und $\vec{y}(t)$
3. Berechnen des Gradienten der Optimierungsfunktion bezgl. der Ausgangsschicht $\frac{\partial E}{\partial v_{kj}}$ unter Benutzung des Netzausgangs $\vec{y}$ und der Zielvektoren $\vec{y}'$

4. Zurückverfolgen[1] der Ausgangsgradienten durch das Netz zur Berechnung der Gradienten $\frac{\partial E}{\partial w_{lk}}$ bezgl. der versteckten Schicht
5. Inkrementieren von t, bis ein Block von T Trainingsdaten verarbeitet worden ist
6. Neuberechnung aller Gewichte in Richtung des *negativen Gradienten*, $t = T + 1$ und zurück zu 2, bis alle Trainingsdaten verarbeitet sind

Das gesamte Training wiederholt sich in mehreren Iterationen. Die Größe des Blocks T muß empirisch bestimmt werden, es sind Werte zwischen 1 (Neuberechnung der Gewichte nach jedem Trainingsbeispiel) und der Gesamtanzahl an Trainingsbeispielen (nur eine Neuberechnung in jeder Iteration) möglich [Schalkoff, 1994].
Beim Training eines MLPs nach dieser Methode kann nur ein *lokales Minimum* der Optimierungsfunktion gefunden werden. Einer guten Initialisierung des Verfahrens kommt also eine wichtige Bedeutung zu. In der Praxis ergibt sich zusätzlich noch das Problem des Übertrainierens, das in Abb. 3.1.2 illustriert ist. Insbesondere bei zu wenigen Trainingsdaten im Verhältnis zur Größe des Netzes ist dieser Effekt unvermeidlich. Wünschenswert wäre die in Abb. 3.1.2 durchgezogen dargestellte Hyperebene, die eine

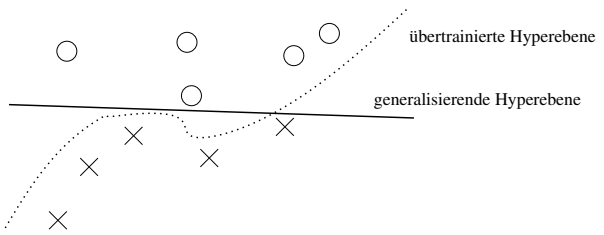


Abbildung 3.1.2.: Beispiel für eine übertrainierte und eine generalisierende Hyperebene

einfache und allgemeine Trennfläche zwischen den zwei Klassen repräsentiert. Besitzt das Netz zu viele Parameter im Verhältnis zur Anzahl der Trainingsdaten, so bildet sich nach mehreren Trainingsiterationen die in Abb. 3.1.2 gepunktet dargestellte Trennfläche heraus, die nicht die erwünschte Verallgemeinerung der Trainingsdaten repräsentiert.
Zur Vermeidung dieses Effekts bietet sich das Kreuzvalidierungsverfahren[2] [Bourlard u. Morgan, 1994] an. Hierbei wird zunächst ein Teil der Trainingsdaten (typisch sind 10%) nicht zum Training verwendet, sondern als *Evaluationsdaten* beiseite gelegt. Nach jeder Iteration findet ein Test nur unter Benutzung der Evaluationsdaten statt. In Abb. 3.1.3 ist der Verlauf der Fehlerrate auf den Trainings- und Evaluationsdaten schematisch wiedergegeben. Mit fortschreitender Iteration ist zunächst eine Verbesserung der Fehlerrate bei diesem Test zu erwarten. Beginnt jedoch ein Übertrainieren des Netzes, so wird das Resultat auf den Evaluationsdaten wieder schlechter werden, da diese ja nicht im Training verwendet werden. Nach [Bourlard u. Morgan, 1994] ist also genau an diesem Punkt das Training des Netzes zu beenden, was eine deutlich verbesserte Generalisierung zur Folge hat. Ein gebräuchliches Kriterium für den Test der Evaluationsdaten ist

[1] *engl.*: back propagation
[2] *engl.*: cross validation

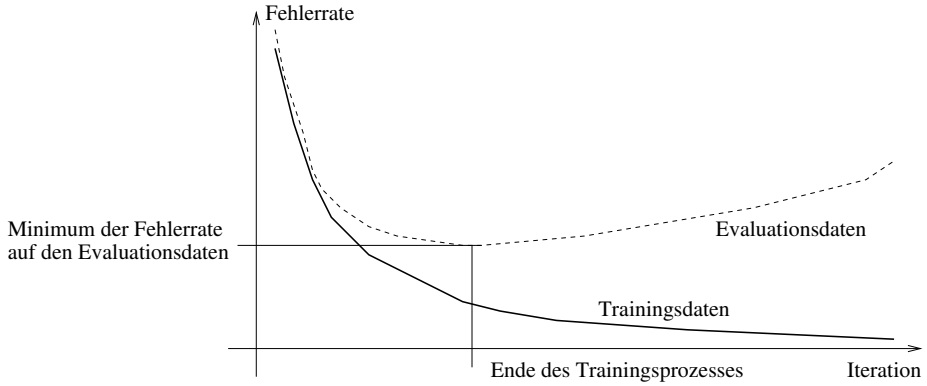
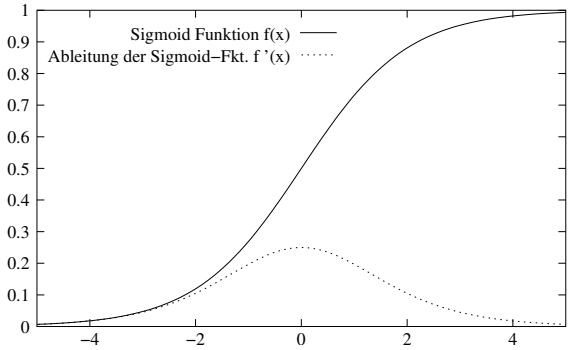


Abbildung 3.1.3.: Schematische Darstellung der Fehlerrate über der Anzahl der Iterationen auf den Trainings- und Evaluationsdaten

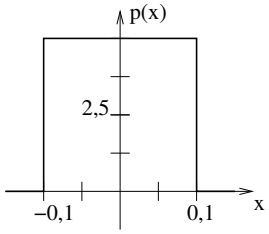
die Klassifikationsfehlerrate der einzelnen Eingangsvektoren. Da bei Sprachsignalen der Eingangsvektor aus einem aus den Audiodaten ausgeschnittenen Fenster besteht, wird diese Fehlerrate auch als *Fenster-Fehlerrate* (FFR) bezeichnet. Eine Diskussion und Ergebnisse zur Fenster-Fehlerrate verschiedener Netze finden sich im Abschnitt 3.3.1.

Initialisierung des Netzes

Wie oben beschrieben, erlaubt das Training des NNs mittels Gradientenabstieg nur ein lokales Minimum der Optimierungsfunktion zu finden. Aus diesem Grund ist die Wahl eines geeigneten Startwertes bei der Minimumsuche von entscheidender Bedeutung. Dem gegenüber steht jedoch im Allgemeinen der Mangel an Information zu einer geeigneten Initialisierung. Insbesondere bei der Phonemklassifikation akustischer Daten liegt kaum Wissen vor, um die Netzstartwerte geeignet zu wählen. Ein häufig verwendeter Kompromiß ist die Initialisierung der Netzgewichte mit Zufallswerten [Schalkoff, 1994; Rigoll, 1994c]. Diese sind dann so zu wählen, daß der Trainingsalgorithmus den dynamischen Bereich der Netzausgänge möglichst gut ausnutzen kann. Abbildung 3.4(a) zeigt den Verlauf der Sigmoid-Funktion und ihrer Ableitung (welche in der Gradientenberechnung benötigt wird). Um nun größere Änderungen an den Gewichten vornehmen zu können, muß also der Startwert der Sigmoid-Funktion möglichst in der Nähe des Nullpunktes liegen. Zu erreichen ist dies durch eine mittelwertfreie Verteilung mit kleiner Varianz. Im Folgenden wird hierzu die Gleichverteilung aus Abb. 3.4(b) angenommen. Lediglich die Bias-Gewichte zwischen versteckten Neuronen und Ausgängen können mit aussagekräftigeren Werten besetzt werden: Die Bias-Gewichte v_{0j} bestimmen die Netzausgänge unter der Annahme, daß die gewichtete Summe der übrigen Vektorkomponenten von $\vec{z}$ sich bei einem initialisierten Netz zu (annähernd) Null ergibt. Beim untrainierten Netz fordert man nun, daß die Netzausgänge die *a priori*-Wahrscheinlichkeiten $Pr(\rho_j)$ der



(a) Sigmoid-Funktion und ihre Ableitung



(b) Gleichverteilung zur Initialisierung der NN-Gewichte

Abbildung 3.1.4.: Nichtlinearität und Initialisierungsdichte des NN

zugeordneten Klassen schätzen. Damit ergibt sich mit Gl. (3.1.3) für die Bias-Gewichte der Ausgangsschicht

$$v_{0j} = \log\left(\frac{Pr(\rho_j)}{1 - Pr(\rho_j)}\right) \tag{3.1.5}$$

Gradientenberechnung

Die übliche Wahl des *quadratischen Fehlers* als Optimierungsfunktion [Schalkoff, 1994] ist wegen der *Softmax*-Funktion in der Ausgangsschicht und der Verwendung als Symbolklassifikator für die Spracherkennung nicht optimal. Stattdessen bietet sich eine Maximierung der Kreuzentropie zwischen Transkription und Netzausgängen an, die nach [Joost u. Schiffmann, 1998; Zhou u. Austin, 1998] für die hier zu lösende Aufgabe der Phonemklassifikation besser geeignet ist. Hierfür wird zunächst angenommen, daß das NN auf den Trainingsdaten (mit bekannter Klassenzugehörigkeit) die Wahrscheinlichkeit

$$y_j = Pr(\rho = \rho_j | \vec{x}(t) \in \text{Klasse j}) \tag{3.1.6}$$

approximiert. Wenn weiterhin davon ausgegangen wird, daß die Trainingsbeispiele untereinander statistisch unabhängig sind, so folgt

$$\prod_{t=1}^{T} Pr(\rho = \rho_j | \vec{x}(t) \in \text{Klasse j}) = \prod_{t=1}^{T} \prod_{m=1}^{J} (y_m(t))^{y'_m(t)} \ , \tag{3.1.7}$$

wobei $y'_m(t)$ ein Element des Zielvektors darstellt (δ_{mj} bezeichnet das Kroneckersymbol):

$$y'_m(t) = \delta_{mj} = \begin{cases} 1 & \text{für } m = j \\ 0 & \text{sonst} \end{cases} \tag{3.1.8}$$

Die Kreuzentropie als Optimierungsfunktion E ergibt sich durch Logarithmieren von Gl. (3.1.7):

$$E = \sum_{t=1}^{T} \sum_{m=1}^{J} y'_m(t) \log y_m(t) \tag{3.1.9}$$

Benutzt man also die Kreuzentropie für alle T Trainingsdaten und verwendet für die Nichtlinearität $F_o(.)$ die *Softmax*-Funktion (Gl. (3.1.4)), so ergibt sich für den Gradienten der Matrix der Ausgangsgewichte $\mathbf{V}$:

$$\frac{\partial E}{\partial v_{kj}} = \sum_{t=1}^{T} \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial \xi_j} \frac{\partial \xi_j}{\partial v_{kj}} = \sum_{t=1}^{T} (y_j'(t) - y_j(t)) z_k = \sum_{t=1}^{T} d_j z_k \,, \tag{3.1.10}$$

mit der Ableitung der Softmax-Funktion $\frac{\partial y_a}{\partial \xi_a} = y_j \delta_{aj} - y_a y_j$ und dem Hilfsvektor $\vec{\xi} = \mathbf{V}^T \vec{z}$. Die Abkürzung $d_j = \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial \xi_j}$ wird in der Literatur [Schalkoff, 1994] oft als *delta*-Wert bezeichnet. Insbesondere beim Umgang mit rückgekoppelten Netzen (s. 3.1.2) erreicht diese Abkürzung eine deutliche Vereinfachung der Beschreibung. Bei der Berechnung des Gradienten der "inneren" Schicht muß der Einfluß aller Ausgangsgradienten berücksichtigt werden. Bei Verwendung der *Sigmoid*-Funktion (Gl. (3.1.3)) als Nichtlinearität $F_h(.)$ und mit der Definition $\vec{\zeta} = \mathbf{W}^T \vec{x}$, sowie der Ableitung der Sigmoid-Funktion $\frac{\partial z_k}{\partial \zeta_k} = z_k(1 - z_k)$ läßt sich der Gradient der Gewichtsmatrix $\mathbf{W}$ unter Benutzung der oben eingeführten Hilfsgrößen schreiben als

$$\frac{\partial E}{\partial w_{ik}} = \sum_{t=1}^{T} \left(\sum_{j=1}^{J} \frac{\partial E}{\partial y_j} \frac{\partial y_j}{\partial \xi_j} \frac{\partial \xi_j}{\partial z_k} \right) \frac{\partial z_k}{\partial \zeta_k} \frac{\partial \zeta_k}{\partial w_{ik}} = \sum_{t=1}^{T} \left(\sum_{j=1}^{J} d_j v_{kj} \right) \; z_k(1 - z_k)\, x_i \tag{3.1.11}$$

An der Gleichung (3.1.11) erkennt man die Fortpflanzung der berechneten Gradienten rückwärts durch das Netz (*back propagation*). Zur Neuberechnung der Gewichte der Eingangsschicht muß also zunächst der komplette Gradient der Ausgangsschicht bekannt sein.

Neuberechnung der Gewichte

Die Neuberechnung der Gewichte erfolgt in Richtung des negativen Gradienten, um die Optimierungsfunktion E zu minimieren. Am Beispiel der Gewichtsmatrix $\mathbf{V}$ ergibt sich:

$$v_{kj}^{(n+1)} = v_{kj}^{(n)} - \Delta v_{kj}^{(n)} \text{ mit } \Delta v_{kj}^{(n)} = \beta \frac{\partial E^{(n)}}{\partial v_{kj}} \tag{3.1.12}$$

Die Neuberechnung der Gewichtsmatrix $\mathbf{W}$ geschieht analog. Die Nachteile dieser Strategie sind

1. Der Algorithmus bleibt in einem lokalen Minimum stecken, der Ort des Minimums hängt von der Initialisierung des Netzes und der "Lernrate" β ab.

2. Der zu wählende Parameter β beeinflußt das Konvergenzverhalten des Algorithmus.

3. Das Konvergenztempo hängt direkt von der Größe des Gradienten ab. Da der Gradient in der Umgebung eines Extremums kleiner wird, nimmt das Konvergenztempo in dieser Umgebung ab.

Insbesondere der Einfluß von β auf das Konvergenzverhalten stellt eine große Schwierigkeit dar: Bei zu kleiner Wahl von β benötigt der Algorithmus unnötig lange, um zum Minimum zu gelangen. Eine zu große Wahl führt im schlechtesten Fall zu Oszillationen, ohne sich dem Minimum zu nähern. Eine geeignete Wahl von β hängt also vom nicht bekannten Verlauf der Optimierungsfunktion ab und kann nur durch Vorexperimente empirisch bestimmt werden.
Ein besseres Konvergenzverhalten des Trainingsalgorithmus kann durch verschiedene Variationen von Gl. (3.1.12) erreicht werden:

- Momentum
  Die Neuberechnung der Gewichte nach Gl. (3.1.12) kann zu sehr unregelmäßigen Veränderungen der Gewichte führen, falls $\Delta v_{kj}^{(n)}$ große Werte annimmt. Das Übertragungsverhalten dieser Gleichung kann als I-Glied aufgefaßt werden [Rigoll, 1994c]. Durch Einführen von Verzögerungselementen höherer Ordnung kann das Übertragungsverhalten gedämpft werden. Im Originalbereich ergibt sich der sogenannte *Momentum*-Term als Zusatz, der eine regelmäßigere Änderung der Gewichte erzeugt:
  $$v_{kj}^{(n+1)} = v_{kj}^{(n)} + \theta \left(v_{kj}^{(n)} - v_{kj}^{(n-1)} \right) - \Delta v_{kj}^{(n)} \tag{3.1.13}$$

- *Resilient Propagation* (RPROP) [Riedmiller u. Braun, 1993; Igel u. Hüsken, 2000]
  Die wesentliche Neuerung des RPROP Algorithmus ist die Unabhängigkeit der Größe Δv_{kj} aus Gl. (3.1.12) vom Betrag des lokalen Gradienten an dieser Stelle. Das Vorzeichen der Gewichtsänderung bestimmt sich über das Vorzeichen des Gradienten. Der Betrag der Änderung Δv_{kj} ist zunächst frei wählbar und ändert sich je nach Vorzeichenwechsel des Gradienten:
  $$\Delta v_{kj}^{(n)} = -\operatorname{sgn}\left(\frac{\partial E^{(n)}}{\partial v_{kj}} \right) \Delta_{jk}^{(n)} \tag{3.1.14}$$
  mit
  $$\Delta_{kj}^{(n)} = \begin{cases} \beta^{+} \cdot \Delta_{kj}^{(n-1)}, & \text{falls } \frac{\partial E^{(n-1)}}{\partial v_{kj}} \frac{\partial E^{(n)}}{\partial v_{kj}} > 0 \\ \beta^{-} \cdot \Delta_{kj}^{(n-1)}, & \text{falls } \frac{\partial E^{(n-1)}}{\partial v_{kj}} \frac{\partial E^{(n)}}{\partial v_{kj}} < 0 \\ \Delta_{kj}^{(n-1)}, & \text{sonst} \end{cases} \tag{3.1.15}$$
  Ändert sich das Vorzeichen des Gradienten, so wurde ein Extremum übersprungen und die Gewichtsänderung wird um einen festen Faktor verkleinert. Liegt kein Vorzeichenwechsel vor, so kann die Gewichtsänderung weiter vergrößert werden.
  Zusätzlich kann (je nach Implementierung) ein Sprung über ein Extremum zurückgenommen werden. Ein großer Vorteil bei der praktischen Umsetzung ist, daß die Parameter des RPROP-Algorithmus im Gegensatz zu Gl. (3.1.12) deutlich unabhängiger vom Klassifikationsproblem und vom Aussehen der Optimierungsfunktion gewählt werden können. Die Nachteile Nummer 2 und 3 können damit deutlich abgemildert werden. Bei der blockweisen Neuberechnung der Gewichte, wie sie in dieser Arbeit verwendet wird, ist zu beachten, daß anstelle des Gradienten der

vorigen Iteration nur der Gradient des vorigen Blocks vorliegt. Aus diesem Grund wird für den "alten" Gradienten ein gleitender Mittelwert eingeführt [Robinson, 1994] ($\alpha = 0.5$):

$$\frac{\partial E^{(n-1)}}{\partial v_{kj}} = \alpha \frac{\partial E^{(n-1)}}{\partial v_{kj}} + (1-\alpha)\frac{\partial E^{(n)}}{\partial v_{kj}} \tag{3.1.16}$$

- *Quickprop* (QPROP) [Fahlman, 1988]
  Auch der QPROP-Algorithmus versucht, die Nachteile Nummer 2 und 3 zu reduzieren. Ausgangspunkt ist der Wunsch, die schrittweise Minimierung des Gradienten durch einen direkten Sprung ins Minimum zu ersetzen. Da eine Berechnung höherer Ableitungen der Optimierungsfunktion bei großen Datenmengen nicht praktikabel ist, betrachtet der QPROP-Algorithmus den aktuellen Gradienten $\frac{\partial E^{(n)}}{\partial v_{kj}}$ und den vorherigen Gradienten $\frac{\partial E^{(n-1)}}{\partial v_{kj}}$, um Informationen über die Krümmung der Optimierungsfunktion zu bekommen. Unter der Annahme, daß die Optimierungsfunktion näherungsweise quadratisch ist und ein endliches Minimum besitzt (die Optimierungsfunktion läßt sich als nach oben geöffnete Parabel darstellen), ist ein direkter Sprung ins Minimum durch

$$\Delta v_{jk}^{(n)} = \frac{\frac{\partial E^{(n)}}{\partial v_{kj}}}{\frac{\partial E^{(n-1)}}{\partial v_{kj}} - \frac{\partial E^{(n)}}{\partial v_{kj}}} \Delta v_{jk}^{(n-1)} \tag{3.1.17}$$

  möglich. Zusätzliche Parameter sind auch hier notwendig: Eine maximale Schrittweite $\beta^{\max}\Delta v_{jk}^{(n-1)}$, ersetzt Gl. (3.1.17), falls der Nenner zu klein wird oder sich das Vorzeichen umkehrt, was einen Sprung in die falsche Richtung zur Folge hätte. Ist die Gewichtsänderung $\Delta v_{jk}^{(n-1)} = 0$ (z.B. beim Start des Algorithmus), so wird das neue Gewicht nach Gl. (3.1.12) neu berechnet.

3.1.2. Rückgekoppelte Netze

Zusätzliche Möglichkeiten zur Klassifikation von Mustern mit zeitlicher Ordnung bietet die Einführung von Rückkopplungen im NN. Durch eine einfache Rückkopplung können vergangene Muster und vergangene Entscheidungen des Netzes indirekt zur aktuellen Klassifikation beitragen, der Einfluß der Vergangenheit nimmt exponentiell ab. Möglichkeiten zur Realisierung wären zum Beispiel Rückkopplungen innerhalb der versteckten Schicht eines MLPs oder Rückkopplungen vom Ausgang zurück auf den Eingang.
Der erste Fall führt zur Jordan-Architektur, die nur geringe Modifikationen am Trainingsalgorithmus erfordert [Jordan, 1986]. Größere Flexibilität hat eine Rückkopplung der Ausgangsknoten zurück zum Eingang. Durch diese Maßnahme werden die Schichten des Netzwerks zeitlich aneinandergefügt (s. Bild 3.1.6), so daß eine zusätzliche versteckte Schicht wie beim MLP zur Erzeugung nicht-linearer Klassifikationsflächen nicht mehr notwendig ist. In [Robinson, 1994] ist diese Architektur für die Phonemklassifikation verändert worden, indem die Ausgangsknoten mit Phonemauftrittswahrscheinlichkeiten von den Ausgangsknoten zur Rückkopplung getrennt werden. Daraus folgt dann ein

rückgekoppeltes neuronales Netz[3] (RNN) mit der Netztopologie aus Bild 3.1.5. Der

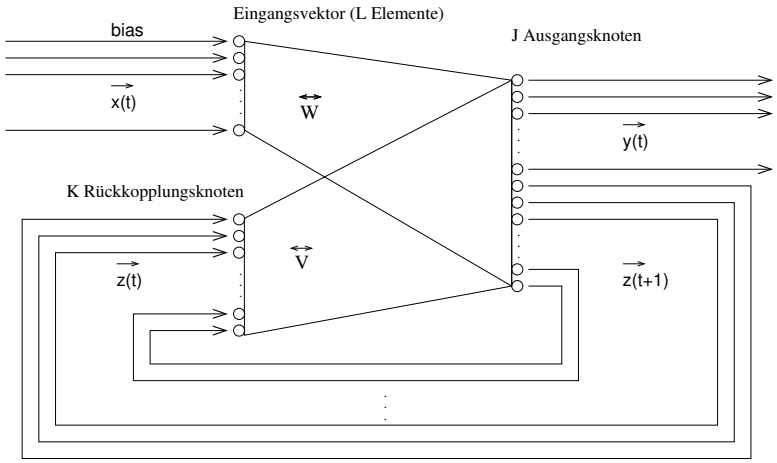


Abbildung 3.1.5.: Rekurrentes neuronales Netzwerk

Ausgangsvektor des Netzes ergibt sich nach der im Bild 3.1.5 gewählten Notation zu

$$\vec{y}(t) = \vec{F}_o\left(\mathbf{W}^T\vec{x}(t)|_{1..J} + \mathbf{V}^T\vec{z}(t)|_{1..J}\right) \tag{3.1.18}$$

$$\vec{z}(t+1) = \vec{F}_h\left(\mathbf{W}^T\vec{x}(t)|_{J+1..J+K} + \mathbf{V}^T\vec{z}(t)|_{J+1..J+K}\right) \tag{3.1.19}$$

wobei die Notation $\mathbf{V}^T\vec{z(t)}|_{1..J}$ die Komponenten 1 bis J des Vektors $\mathbf{V}^T\vec{z}$ bezeichnet. Die Bezeichnungen der Ein- und Ausgänge entsprechen denen aus Abschnitt 3.1.1, hier ist $\vec{z}$ aber der Vektor der Rückkopplungsknoten (anstelle der versteckten Schicht). $F_o(.)$ ist wieder die Nichtlinearität der Ausgangsknoten, $F_h(.)$ die Nichtlinearität der netzinternen (Rückkopplungs-)Knoten. Analog zum MLP werden auch hier die *Sigmoid*-Funktion für $F_h(.)$ und die *Softmax*-Funktion für $F_o(.)$ verwendet.
Für die Anwendung der Phonemklassifikation kann auch hier gezeigt werden, daß ein RNN in der oben dargestellten Formulierung und dem nachfolgend beschriebenen Training in der Lage ist, *a posteriori*-Auftrittswahrscheinlichkeiten (gegeben den Eingangsvektor) zu schätzen. Die mathematische Ableitung dazu findet sich in [Santini u. Bimbo, 1995].

Trainingsverfahren

Zum Training eines RNNs nach Bild 3.1.5 existieren im Wesentlichen zwei Verfahren [Williams u. Zipser, 1990]:

- Real Time Recurrent Learning (RTRL)
- Back Propagation Through Time (BPTT)

[3] *engl.* Recurrent Neural Net

BPTT erfordert gegenüber RTRL einen größeren Bedarf an Trainingsdaten, um die gleiche Qualität des Netzes zu erzielen, besitzt aber einen deutlichen Geschwindigkeitsvorteil durch einfachere Berechnungen. Da beim Training von Spracherkennern üblicherweise relativ große Mengen an Trainingsdaten zur Verfügung stehen (mind. 10^6 Trainingsbeispiele), ist das Training mit dem BPTT-Algorithmus die bevorzugte Alternative. Der prinzipielle Ablauf des Netztrainings unter Verwendung von BPTT kann wie folgt zusammengefaßt werden:

1. Initialisierung des Netzes $t = 1$
2. Hineingeben der Trainingsdaten $\vec{x}(t)$ und des alten Rückkopplungsvektors $\vec{z}(t)$ in das Netz und berechnen des neuen Rückkopplungsvektors $\vec{z}(t+1)$ und des Ausgangsvektoren $\vec{y}(t)$
3. Inkrementieren von t und zurück zu 2, bis ein Block von T_1 Eingangsdaten verarbeitet worden ist
4. Berechnen des Gradienten der Optimierungsfunktion E bezgl. der Ausgangsschicht und der Rückkopplungsschicht unter Benutzung der Netzvektoren $\vec{z}(t)$, $\vec{y}(t)$ und der Zielvektoren $\vec{y}'(t)$ für alle durchlaufenen Zeitschritte, beginnend mit den zuletzt durchlaufenen Daten (*back propagation*)
5. Setzen des Zeitpunktes $t = T_1 + 1$ und Durchlaufen des nächsten Blocks von Trainingsdaten (zurück zu 2) bis eine vorher festgelegte Anzahl an Trainingsdaten $T_2 \geq T_1$ durchlaufen worden ist
6. Neuberechnung aller Gewichte in Richtung des *negativen Gradienten* und mit $t = T_2 + 1$ zurück zu 2 bis alle Trainingsdaten verarbeitet sind

Wie bereits im Abschnitt 3.1.1 beschrieben, wird auch hier der Vorgang in mehreren Iterationen wiederholt. Die Anzahl der Neuberechnungen der Gewichte ist ebenso wie die Größe eines Trainingsdatenblocks zur Gradientenberechnung nur empirisch bestimmbar und hängt von der Netzgröße, der Anzahl der Trainingsdaten und der Art der Aufgabe ab. In [Williams u. Zipser, 1990] findet sich eine Diskussion des Trainingsverfahrens für das Problem der Blockgröße und der Anzahl der Neuberechnungen.
Zur Vermeidung des Übertrainings wird, wie schon in 3.1.1 beschrieben, nach jeder Iteration eine Kreuzvalidierung mit nicht verwendeten Daten durchgeführt und das Training gegebenenfalls beendet.

Initialisierung des Netzes

Die Initialisierung eines RNNs kann weitgehend analog zur Initialisierung eines MLPs (s. Abschnitt 3.1.1) ablaufen. In [Senior, 1994] ist festgestellt worden, daß nach dem Training eines mit Zufallswerten initialisierten RNNs die Gewichte der Rückkopplungsmatrix $\mathbf{V}$ zum gleichen Knoten ($v_{k(J+k)}$) einen positiven Wert aufweisen, während die übrigen Gewichte ($v_{k(J+m)}$, mit $k \neq m$) mit kleinem negativen Mittelwert und größerer Varianz schwanken. Dies läßt die Interpretation zu, daß das RNN seinen Rückkopplungszustand beibehalten möchte und der Eingangsvektor $\vec{x}$ eine Störung dieses Zustandes

repräsentiert. Also kann nach [Senior, 1994] bei der Initialisierung der Matrix $\mathbf{V}$ der Rückkopplungsgewichte durch eine Voreinstellung dieses Zustandes eine Verbesserung des Trainingsverfahrens erreicht werden:
Ohne Einfluß eines Eingangsvektors ($\vec{x} = \vec{0}$) kann ein konstanter Rückkopplungsvektor $\vec{z}(t) = (\frac{1}{2}, ..., \frac{1}{2})^T$ angenommen werden - dies ergibt sich aus der Auswertung der Sigmoid-Funktion an der Stelle Null (s. Abb. 3.4(a)). Zum Start des Netzes werden die Rückkopplungsknoten also auf diesen Wert gesetzt, eine Beibehaltung dieses Zustandes ergibt dann aus Gl. (3.1.19) mit dem Bias w_{0k}

$$z_k(t+1) = \frac{1}{2} = F_h(\sum_{m=1}^{K} w_{0(J+k)} + v_{m(J+k)} z_m(t)) = F_h(\sum_{m=1}^{K} w_{0(J+k)} + v_{m(J+k)} \frac{1}{2}) \quad (3.1.20)$$

Unter Benutzung der oben erwähnten Beobachtungen aus dem Training von RNN läßt sich diese Gleichung zu

$$\frac{1}{2} \approx F_h(w_{0(J+k)} + v_{k(J+k)} \frac{1}{2}) \Leftrightarrow v_{k(J+k)} \frac{1}{2} = -w_{0(J+k)} \quad (3.1.21)$$

vereinfachen. Betrachtet man nun den Rückkopplungsvektor als Addition des stabilen Zustandes plus einer "kleinen"[4] Störung ϵ_k: $z_k(t) = \frac{1}{2} + \epsilon_k$, so folgt mit Gl. (3.1.21)

$$z_k(t+1) = F_h(w_{0(J+k)} + v_{k(J+k)} \frac{1}{2} + v_{k(J+k)} \epsilon_k) \Rightarrow z_k(t+1) = F_h(v_{k(J+k)} \epsilon_k) \quad (3.1.22)$$

In der Umgebung des Nullpunktes kann die Sigmoid-Funktion $F_h(.)$ durch eine Taylorreihe nur mit linearem Term ausgedrückt werden:

$$z_k(t+1) \approx F_h'(0) \epsilon_k v_{k(J+k)} + F_h(0) \quad (3.1.23)$$

Durch Umformung von Gl. (3.1.23) und mit $z_k(t) = \frac{1}{2} + \epsilon_k$ ergibt sich schließlich

$$z_k(t+1) - F_h(0) \approx F_h'(0) \epsilon_k v_{k(J+k)} \Rightarrow \epsilon_k \approx F_h'(0) \epsilon_k v_{k(J+k)} \Rightarrow 1 = F_h'(0) v_{k(J+k)} \quad (3.1.24)$$

Damit sind nun für die Rückkopplungsgewichte zu gleichen Rückkopplungsknoten $v_{k(J+k)} \approx \frac{1}{F_h'(0)} = 4$ und für die Bias-Gewichte $w_{0(J+k)} = -v_{k(J+k)} \frac{1}{2} = -2$ geeignete Startwerte gefunden, die in der Umgebung der zu erwartenden Lösung liegen. Zusätzliche Diskriminanz kann durch die Annahmen für die Bias-Gewichte aus Abschnitt 3.1.1 hineingebracht werden, da die einzelnen Bias-Gewichte dann verschiedene Werte annehmen (vergl. Gl. (3.1.5)). Bei einem typischen Wert für die *a priori*-Klassenwahrscheinlichkeiten $Pr(j)$ von $0,05$ (z.B. bei einem Phonemklassifizierungsproblem) ergibt sich $w_{0j} = -2,94$, was noch im akzeptablen Rahmen mit der oben diskutierten Näherung übereinstimmt.

Berechnung der Gradienten

[4] *klein* im Verhältnis zum Wert des Rückkopplungsknoten

Die wesentliche Idee des BPTT-Verfahrens ist, das Trainingsverfahren für MLPs (Abschnitt 3.1.1) für rückgekoppelte Netze zu portieren. Dies geschieht durch Ausfalten des Netzes über die Zeit (s. Abb. 3.1.6), es entsteht ein virtuelles MLP mit einer Anzahl an Schichten gleich der Anzahl der durchlaufenen Zeitschritte, jedoch mit zusätzlichen Ein- und Ausgängen in jeder Schicht. Bei Verwendung der gleichen Optimierungsfunk-

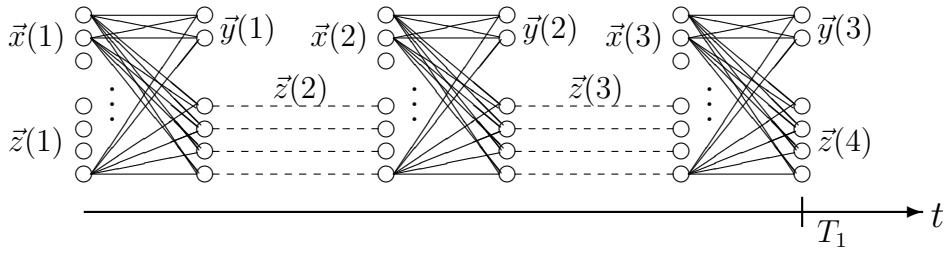


Abbildung 3.1.6.: Ausfalten des rückgekoppelten Netzes in der Zeit (Beispiel mit $T_1 = 3$)

tion E (Kreuzentropie, s. Gl. 3.1.9) ergeben sich für die Gradienten Ausdrücke analog zum MLP. Es ist hier zweckmäßig, zunächst nur die *delta*-Werte für die Ausgangs- und Rückkopplungsknoten zu bestimmen. Am zuletzt durchlaufenen Zeitschritt T_1 ergibt sich

$$d_j(T_1) = \begin{cases} \frac{\partial E}{\partial y_j(T_1)} \frac{\partial y_j(T_1)}{\partial \xi_j(T_1)} = y_j'(T_1) - y_j(T_1) \text{ für einen Ausgangsknoten } (j = 1 \dots J) \\ 0 \text{ für einen Rückkopplungsknoten } (j = J+1 \dots J+K) \end{cases} \tag{3.1.25}$$

Für einen Rückkopplungsknoten existiert im letzten Zeitschritt kein Zielvektor. Der komplette Gradient für den letzten Zeitschritt T_1 eines Blocks ist dann mit $j = 1 \dots J + K$

$$\begin{aligned} \frac{\partial E(T_1)}{\partial w_{lj}} &= d_j(T_1)\, x_l(T_1) \\ \frac{\partial E(T_1)}{\partial v_{kj}} &= d_j(T_1)\, z_k(T_1 - 1) \end{aligned} \tag{3.1.26}$$

Geht man dann einen Zeitschritt zurück, so müssen die Gradienten eine Schicht rückwärts verfolgt werden (*back propagation*). Die folgenden Ausdrücke gelten für alle weiteren Zeitschritte/"Schichten" $t < T_1$:

$$d_j(t) = \begin{cases} y_j'(t) - y_j(t) \text{ für } j = 1 \dots J \\ \left(\sum_{m=1}^{J+K} d_m(t+1) v_{jm} \right) z_{(j-J)}(t)(1 - z_{(j-J)}(t)) \\ \qquad\qquad \text{für } j = J+1 \dots K \end{cases} \tag{3.1.27}$$

Hierbei ist wieder die Ableitung der Sigmoid-Funktion in den Rückkopplungsknoten $\frac{\partial z_k}{\partial \zeta_k} = z_k(1 - z_k)$ eingesetzt worden. Die Gradienten ergeben sich analog zu Gl. (3.1.26), diesmal für Zeitschritt t mit $j = 1 \ldots J + K$:

$$\begin{aligned} \frac{\partial E(t)}{\partial w_{lj}} &= d_j(t)\, x_l(t) \\ \frac{\partial E(t)}{\partial v_{kj}} &= d_j(t)\, z_k(t-1) \end{aligned} \tag{3.1.28}$$

Bis ein übergeordneter Block $T_2 \geq T_1$ durchlaufen worden ist, können die einzelnen Gradienten aufsummiert werden und es kann danach eine Neuberechnung der Gewichte stattfinden.

Neuberechnung der Gewichte

Die Neuberechnung der Gewichte läuft beim Training eines RNNs analog zur Neuberechnung in einem MLP ab. Aus diesem Grund können alle in 3.1.1 dargestellten Algorithmen unverändert auch für RNN verwendet werden, zu beachten ist lediglich, daß der Vektor der versteckten Neuronen hier durch den Vektor der Rückkopplungsneuronen ersetzt wird.

3.1.3. Einbeziehung von Kontextinformation

Die Klassifikationsleistung der im Abschnitt 3.1.1 und 3.1.2 beschriebenen Netze ist für ein hybrides akustisches Modell, wie es im Kapitel 5 vorgestellt wird, nicht ausreichend. Das größte Problem ist dabei die Klassifikation eines Merkmalsvektors, berechnet aus einem nur 25ms langen Fenster (vergl. Kapitel 2) des Audiosignals. Die zu unterscheidenden Klassen (z.B. Phoneme) umfassen im Gegensatz dazu üblicherweise 3-50 Fenster. Um die dem Netz zur Verfügung stehende Information zu vergrößern, bieten sich folgende Maßnahmen an:

Vergrößerung des Eingangsvektors

Bei der Klassifikation einer zeitlich geordneten Sequenz von Merkmalsvektoren ist prinzipiell der Zugriff auf mehr als einen Merkmalsvektor möglich. Abbildung 3.1.7 zeigt eine Möglichkeit, weitere Merkmalsvektoren in die Klassifikation einzubeziehen: Der erweitere Eingangsvektor für das neuronale Netz besteht aus der Aneinanderreihung $n+m+1$ zeitlich aufeinanderfolgender Merkmalsvektoren $\vec{x(t)} = \Big(\vec{f}(t-n), \ldots, \vec{f}(t), \ldots, \vec{f}(t+m)\Big)$. Sofern keine weiteren Informationen vorliegen, die dagegen sprechen, wird zur Vereinfachung $m = n$ gewählt. Die vom Netz zu berechnende Wahrscheinlichkeit erweitert sich dann zu (vergl. [Bourlard u. Morgan, 1994])

$$Pr(\rho_j|\vec{x}) = Pr(\rho_j|\vec{f}(t-n), \ldots, \vec{f}(t+m)) \tag{3.1.29}$$

In der Praxis geschieht der Zugriff auf zukünftige Fenster durch eine Verzögerung um m Fenster. Die Größe von m (und n) ist nur empirisch bestimmbar und hängt von

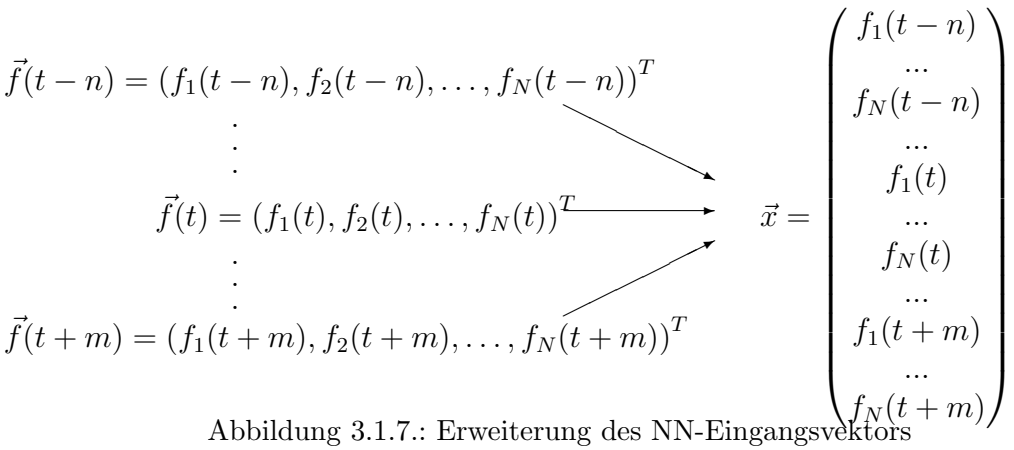


Abbildung 3.1.7.: Erweiterung des NN-Eingangsvektors

dem Klassifikationsproblem und der Netzgröße ab. Theoretisch wäre dieses Verfahren auch für ein rückgekoppeltes Netz möglich, hier wird jedoch der vergangene Kontext automatisch durch die Rückkopplung berücksichtigt, außerdem wird der Vorteil einer kleineren Netzstruktur durch eine vergrößerte Eingangsschicht verringert.

Verzögerung des Ausgangsvektors

Für rückgekoppelte Netze ist neben der durch die Rückkopplungsneuronen vorhandenen Modellierung vergangener Muster auch eine Erweiterung auf zukünftige Daten möglich. Eine Verzögerung des Ausgangsvektors um τ Zeitschritte verändert die Netzfunktion $\vec{y}(t) = F_o(\vec{x}(t))$ (vergl. Gl. (3.1.19)) zu

$$\vec{y}(t-\tau) = F_o(\vec{x}(t)) \tag{3.1.30}$$

Dadurch "sieht" das Netz Eingangsvektoren bis zum zukünftigen Zeitpunkt $t+\tau$, bevor der Ausgangsvektor $\vec{y}(t)$ berechnet wird. In der Praxis werden einfach τ Eingangsvektoren durch das Netz propagiert, bevor der erste Ausgangsvektor ausgegeben wird. Im Gegensatz zur vorher beschriebenen Erweiterung des Eingangsvektors müssen hier keine zusätzlichen Netzparameter hinzugefügt werden, da die Speicherung der Muster bereits in den Rückkopplungsneuronen erfolgt.
Die konkrete Wahl von τ ist nur experimentell möglich und hängt von der Klassifikationsaufgabe ab. Zu große Werte von τ führen durch das zeitliche exponentielle Abklingen des Rückkopplungseinflusses zu einer Vernachlässigung der bereits vergangenen Muster, die aber zeitlich zum berechneten Ausgangsvektor gehören.

3.1.4. Diskussion der Netzparadigmen

Die zu vergleichenden Paradigmen sind in dieser Arbeit die in den Abschnitten 3.1.1 und 3.1.2 vorgestellten Architekturen. Gemeinsam ist beiden Netzen das überwachte Trainingsverfahren, welches die Abweichungen zur Referenz durch das Netz zurückverfolgt. Beide Verfahren finden mittels Gradientenabstieg ein lokales Minimum einer Optimierungsfunktion und die Neuberechnungen der Parameter können in beiden Netztypen mit

den gleichen Algorithmen erfolgen.
Der Hauptunterschied zwischen den Architekturen liegt in der Modellierung von zeitlichem Kontext. Dem MLP können die Eingangsvektoren prinzipiell in beliebiger Reihenfolge präsentiert werden, der Kontext der Merkmalsvektoren wird explizit im Eingangsvektor hinzugefügt. Zusätzlich ist jede Entscheidung des MLPs unabhängig von vorangegangenen oder zukünftigen Entscheidungen. Demgegenüber lernt das RNN auch die zeitliche Abfolge der einzelnen Eingangsvektoren, hier darf die Reihenfolge also nicht ohne Weiteres verändert werden. Kontext wird implizit beim Durchlaufen der Merkmalsvektorsequenz berücksichtigt. Durch die Weiterverwendung des Ergebnisses der Rückkopplungsknoten im nächsten Zeitschritt sind auch die einzelnen Klassifikationen nicht mehr unabhängig. Bei der hier verwendeten Topologie (Bild 3.1.5) klingt jedoch das Gedächtnis exponentiell ab. Trotzdem hat die Tendenz des RNNs, seinen Zustand beizubehalten, einen Einfluß auf die Klassifikationsleistung, insbesondere bei kurzen Phonemen, die dann unter Umständen nicht mehr erkannt werden. Ähnlich wie beim Vergleich zwischen FIR[5]- und IIR[6]-Filtern zeigt sich auch beim Vergleich zwischen MLP und RNN, daß die Rückkopplung bei vergleichbarer Leistung eine Anzahl freier Parameter ersetzt. Das RNN kann also kleiner gehalten werden, was die Geschwindigkeit insbesondere in der Erkennungsphase eines Gesamtsystems deutlich erhöht. Unterschiede in Bezug auf die Klassifikationsleistung der beiden Netzparadigmen werden im Abschnitt 3.3.1 diskutiert.
Zur Beschleunigung der NN-Klassifikation sind Verfahren möglich, die möglichst schon während des Trainings redundante Knoten aus dem Netz entfernen. In [Engelbrecht u. a., 1999] wird hierzu die Betrachtung der Signifikanz eines Parameters, definiert als die Reaktion des Parameters auf Änderungen eines Netzausgangs, betrachtet. Eine Untersuchung dieser Verfahren in [Morgado, 2004] hat insbesondere bei RNN durch Berechnung der Signifikanz eines Rückkopplungsknotens erfolgreiche Ansätze aufgezeigt. Da die Rechenzeit im Rahmen dieser Arbeit nur eine untergeordnete Rolle spielt, wird im Folgenden auf eine weitere Betrachtung dieser Verfahren verzichtet (vergl. dazu die Anforderungen an einen Klassifikator aus Kapitel 7).

3.1.5. Gleichtzeitiges Lernen mehrerer Probleme

Im Zusammenhang mit NN bedeutet das gleichzeitige Lernen mehrerer Probleme das Hinzufügen von Gruppen von Ausgangsneuronen in die Netztopologie, die zusätzliche Klassifikationsaufgaben parallel zur schon vorhandenen Klassifikationsaufgabe durchführen. Die Motivation zur Einführung solcher Zusatzaufgaben ist die Verbesserung der eigentlichen “Hauptaufgabe” durch Lerneffekte bei den Zusatzaufgaben. Eine umfassende Übersicht über die Effekte und das Verhalten von MLP beim Lernen von Zusatzaufgaben bietet [Caruana, 1997]. Danach sind die Gründe für eine Verbesserung der Hauptaufgabe

[5]FIR: *engl.* finite impulse response
[6]IIR: *engl.* infinite impulse response

- Statistische Verstärkung der Trainingsdaten
  Durch die Verwendung der stets begrenzten Menge an Trainingsdaten für mehrere Probleme können die Informationen aus den Daten besser extrahiert werden.

- "Abhören"
  Eine schwierige Klassifikationsaufgabe kann durch die gleichzeitige Klassifikation einer einfachen Zusatzaufgabe beim Lernen der Trennebenen profitieren.

- Verstärkung der Lernfunktion
  Ein besseres Optimum der Lernfunktion könnte gefunden werden, falls die Optimierungsfunktionen zweier Aufgaben an gleicher Stelle ein Extremum aufweisen.

In [Parveen u. Green, 2003] wird dieses Prinzip auf ein NN zur Phonemklassifikation angewandt. Die Zusatzaufgabe besteht hier in der Rekonstruktion des Merkmalsvektors aus einem verrauschten Eingangsvektor. Der Test hat auf der AURORA2 Datenbasis (s. Anhang B.3) stattgefunden. Bemerkenswert hierbei ist, daß die Zusatzaufgabe nicht aus einer weiteren Klassifikation, sondern aus der Rekonstruktion verrauschter Eingangsdaten besteht.
Das Verhalten von RNN unter Verwendung zusätzlicher Klassifikationsaufgaben be-

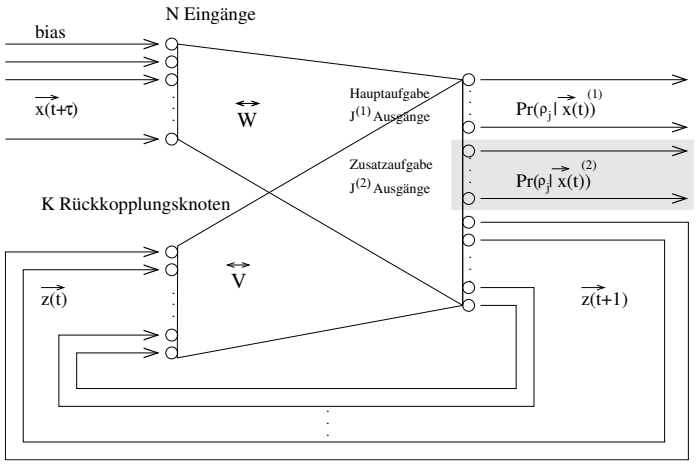


Abbildung 3.1.8.: RNN-Topologie bei Klassifikation zusätzlicher Aufgaben

schreibt [Stadermann u. a., 2005]. Die Netztopologie ist systematisch zur Klassifikation weiterer Aufgaben erweitert worden (s. Abbildung 3.1.8). Für jeden Block an Ausgängen, die jeweils einer Aufgabe zugeordnet sind, gilt

$$\sum_{j=1}^{J^{(m)}} Pr(\rho_j|\vec{x})^{(m)} = 1 \text{ für alle } m \tag{3.1.31}$$

Ergebnisse mit verschiedenen Zusatzaufgaben und deren Beschreibungen werden in den Abschnitten 3.3.1 (nur das RNN) bzw. 5.5.2 (gesamtes hybrides akustisches Modell mit RNN mit Zusatzaufgaben) diskutiert.

3.2. Support-Vektor-Maschinen

Support-Vektor-Maschinen (SVM) stellen ein leistungsstarkes Werkzeug der Mustererkennung dar, deren Theorie im Folgenden in einem kurzen Überblick erläutert wird und die in diesem Kapitel zur Klassifikation von akustischen Daten eingesetzt werden. Unterschiede in der Klassifikationsleistung zu den im Abschnitt 3.1 beschriebenen neuronalen Netzen werden anhand von Experimenten beleuchtet.

3.2.1. Generalisierungsfähigkeit eines Klassifikators

Die Theorie der SVM gründet sich auf die statistische Lerntheorie, die in [Vapnik, 1995] und [Vapnik, 1998] ausführlich beschrieben und mathematisch hergeleitet wird. Hier soll nur ein kurzer Abriß der Ergebnisse präsentiert werden, die zum Verständnis der Eigenschaften der SVM notwendig sind. Eine detaillierte Erläuterung der Theorie ist auch in einem Tutorial zu SVM [Burges, 1998] zu finden.
Angenommen sei eine Menge von Merkmalsvektoren $\vec{x}_i$; $i = 1, \ldots, M$ mit der entsprechenden, bekannten Klassenzugehörigkeit y'_i. Im Falle von akustischen Daten sind die Merkmale zeitlich aufeinanderfolgend (s. Kapitel 2), so daß die Trainingsdaten mit diskretem Zeitindex t numeriert werden können: $(\vec{x}(t), y'(t))\,;\, t = 1, \ldots, T$. Zeitlicher Kontext der Vektoren wird also wie beim MLP (s. Abschnitt 3.1.1) nicht explizit modelliert. Der Klassifikator ist dann eine Funktion $f(.)$ der Eingangsdaten $\vec{x}(t)$ und einem Set von trainierten Parametern α (α kann hier sowohl ein Skalar, als auch eine Matrix sein), die ein Ergebnis liefert, aus dem die Klassenzugehörigkeit der Eingangsdaten ermittelt werden kann. Im Falle eines neuronalen Netzes entspricht das Parameterset α den Gewichtsmatrizen aller Schichten des Netzes.
Das empirische Risiko R_{emp} ist definiert als der gemessene mittlere Fehler, den der Klassifikator auf den Trainingsdaten erzeugt [Burges, 1998]:

$$R_{emp}(\alpha) = \frac{1}{2T} \sum_{i=1}^{T} \left| y'(i) - f(\vec{x}(i), \alpha) \right| \tag{3.2.1}$$

Beschränkt man sich auf binäre Klassifikation, d.h. $y'(i) \in \{1, -1\}$ und $f(\vec{x}, \alpha) \in [1, -1]$, so gilt für ein gewähltes η $(0 \leq \eta \leq 1)$ mit der Wahrscheinlichkeit $1 - \eta$ für das zu erwartende Risiko $R(\alpha)$:

$$R(\alpha) \leq R_{emp}(\alpha) + \sqrt{\frac{h(\log(2T/h) + 1) - \log(\eta/4)}{T}} \tag{3.2.2}$$

Dabei ist h die Vapnik-Chervonenkis (VC) Dimension, sie ist gleich der Maximalzahl von Punkten im Merkmalsraum ($\mathbb{R}^n$), die mit dem jeweiligen Klassifikator auf beliebige Art getrennt werden können.
Das Ziel ist nach [Vapnik, 1995], den Klassifikator zu finden, der die niedrigste obere Schranke (die rechte Seite der Gleichung (3.2.2)) für das zu erwartende Risiko $R(\alpha)$ erzeugt. Eine mögliche Strategie ist also, aus der Menge von Klassifikatoren mit einem bestimmten empirischen Risiko, denjenigen zu wählen, der die geringste VC-Dimension aufweist.

3.2.2. Training von Support-Vektor-Maschinen

Linear separierbare Daten

Unter der Bedingung, daß die Daten $\vec{x}(t)$ mit Klassenzugehörigkeit $y'(t)$ linear separierbar sind, ergibt sich für einen linearen Klassifikator mit den Parametern $\vec{w}, b$:

$$\vec{x}(t) \cdot \vec{w} + b \geq +1 \text{ für } y'(t) = +1 \tag{3.2.3}$$

$$\vec{x}(t) \cdot \vec{w} + b \leq -1 \text{ für } y'(t) = -1 \tag{3.2.4}$$

Die Optimierungsaufgabe besteht nun darin, die Parameter $\vec{w}$ und b so zu bestimmen, daß der Abstand der Hyperebenen $H_1 : \vec{x}(t)\vec{w} + b = 1$ und $H_{-1} : \vec{x}(t)\vec{w} + b = -1$ maximiert wird. In Abb. 3.2.1 sind Mustervektoren und Hyperebenen beispielhaft für

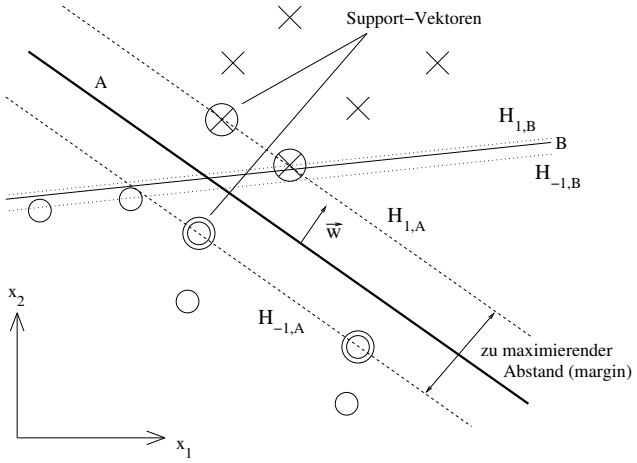


Abbildung 3.2.1.: Hyperebenen und Mustervektoren in einem zweidimensionalen Beispiel mit linearen separierbaren Datenvektoren

ein zweidimensionales Problem dargestellt. Zu erkennen ist, daß der maximale Abstand der Hyperebenen H_1 und H_{-1} nur bei der Trennebene A erreicht ist. Die Trennebene B separiert die Daten ebenfalls linear und ohne Fehler, erzeugt aber nicht den maximalen Abstand der Hyperebenen H_1 und H_{-1}. Weiterhin ist ersichtlich, daß nur einige wenige Datenvektoren $\vec{x}(i)$ die Lage der Hyperebenen H_1 und H_{-1} bestimmen. Das Auffinden dieser *Support*-Vektoren $\vec{s}$ (in Abb. 3.2.1 eingekreist) ist das Ziel des Algorithmus, realisiert durch Minimierung von $||\vec{w}||^2$: Der Abstand zwischen H_1 und H_{-1} kann aus Gl. (3.2.4) zu $d = \frac{2}{||\vec{w}||}$ bestimmt werden. Für die VC-Dimension gilt dann nach [Schölkopf, 2001]: $h \leq R^2 d^2$, wobei R den kleinsten Radius der Sphäre um den Ursprung darstellt, welche alle Trainingsdaten einschließt.
Werden Lagrange-Multiplikatoren α_i eingeführt, so ergibt sich ein konvexes quadratisches Optimierungsproblem. Zu minimieren ist

$$E = \frac{1}{2}||\vec{w}||^2 - \sum_{i=1}^{T} \alpha_i y'(i) \left(\vec{x}(i) \cdot \vec{w} + b\right) + \sum_{i=1}^{T} \alpha_i \tag{3.2.5}$$

bezüglich der Gewichte $\vec{w}$ und des Offsets b, bei gleichzeitiger Maximierung der dualen Variablen α_i [Schölkopf, 2001]. Setzt man die Lösungen der Gradienten $\frac{\partial E}{\partial \vec{w}} = 0$ und $\frac{\partial E}{\partial b} = 0$ ein, so erhält man das duale Problem:

$$E_D = \sum_{i=1}^{T} \alpha_i - \sum_{i,j=1}^{T} \alpha_i \alpha_j y'(i) y'(j) \left(\vec{x}(i) \cdot \vec{x}(j)\right) \tag{3.2.6}$$

Die Lösung führt über die Maximierung des dualen Problems und ergibt $\vec{w} = \sum_{i=1}^{N_S} \alpha_i y'(i) \vec{x}(i)$, wobei N_S die Anzahl der Support-Vektoren darstellt, da nur bei diesen die Bedingung $\alpha_i > 0$ erfüllt ist (für alle anderen Trainingsvektoren gilt $\alpha_i = 0$). Die Größe b kann anschließend aus den Karush-Kuhn-Tucker-Bedingungen [Burges, 1998] ermittelt werden.

Erweiterung auf nicht-separable Probleme

Die obigen Überlegungen sind auf nicht-separable Probleme erweiterbar, indem die Klassifikationsbedingungen (3.2.4) durch Einführung zusätzlicher Bestrafungsvariablen ξ_t gelockert werden:

$$\begin{aligned} \vec{x}(t) \cdot \vec{w} + b &\geq +1 - \xi_t \text{ für } y'(t) = +1 \\ \vec{x}(t) \cdot \vec{w} + b &\leq -1 + \xi_t \text{ für } y'(t) = -1 \\ \xi_t &\geq 0 \end{aligned} \tag{3.2.7}$$

Trainingsfehler (Datenpunkte auf der falschen Seite der Hyperebene) können nun unter Berücksichtigung der zusätzlichen Kosten ξ_t behandelt werden, die Optimierungsfunktion (Gl. 3.2.5) ändert sich zu

$$E = \frac{1}{2}||\vec{w}||^2 + C \sum_{i=1}^{T} \xi_i - \sum_{i=1}^{T} \alpha_i \left[y'(i) \left(\vec{x}(i) \cdot \vec{w} + b\right) - 1 + \xi_i\right] - \sum_{i=1}^{T} \beta_i \xi_i \tag{3.2.8}$$

wobei β_i zusätzliche Lagrange-Multiplikatoren sind, die $\xi_i \geq 0$ sicherstellen. Die Größe C ist ein einstellbarer Parameter, der die Trainingsfehler in der Optimierungsfunktion gewichtet (je größer C, desto stärker werden Trainingsfehler "bestraft"). Die duale Formulierung des Problems Gl. (3.2.6) bleibt unverändert, allerdings gilt nun $0 \leq \alpha_i \leq C$.

Kernel-Funktionen

Die bisherigen Überlegungen ergeben einen linearen Klassifikator mit optimaler Hyperebene. Durch Einführung einer *Kernel*-Funktion läßt sich die bisher beschriebene Theorie auch zur Generierung nicht-linearer Hyperebenen einsetzen: Sowohl bei der Klassifikation (Gl. (3.2.7)) als auch bei der Formulierung der Optimierungsfunktion (Gl. (3.2.8)) tauchen die Trainingsdaten $\vec{x}(t)$ nur im Skalarprodukt $\vec{w} \cdot \vec{x}(t)$ auf. Dadurch ergibt sich die Möglichkeit, die Daten durch eine Transformation $T\{.\}$ in einen beliebigdimensionalen Raum zu transformieren, in dem dann das Skalarprodukt $T\{\vec{x}(t)\} \cdot T\{\vec{w}\}$ berechnet wird. In diesem hochdimensionalen Raum sind dann die Daten linear trennbar, so daß alle bisherigen Überlegungen dieses Kapitels in diesem Raum gültig sind.

Um diese Transformation nicht explizit für jeden Datenpunkt durchzuführen, führt man eine Kernel-Funktion $K\left(\vec{x_i}, \vec{x_j}\right)$ ein mit

$$K\left(\vec{x}(t), \vec{w}\right) = K(\vec{x}(t))K(\vec{w}) \tag{3.2.9}$$

Funktionen, die die Bedingung aus Gl. (3.2.9) erfüllen, werden durch Überprüfung der Mercer-Bedingungen [Burges, 1998] gefunden. Beispiele für Kernelfunktionen sind

- Polynomialer Kernel $K(\vec{x_i}, \vec{x_j}) = (\vec{x_i}\vec{x_j} + 1)^p$
- Sigmoid-Kernel $K(\vec{x_i}, \vec{x_j}) = \tanh\left(\kappa\vec{x_i}\vec{x_j} - \nu\right)$
- Gauß-Kernel $K(\vec{x_i}, \vec{x_j}) = \frac{1}{2\sigma^2}\exp\left(-||\vec{x_i}\vec{x_j}||^2\right)$

Die einzige Änderung an den Trainingsgleichungen ist die Transformation des Skalarproduktes in einen (unbekannten) hochdimensionalen Raum durch Anwendung der Kernelfunktion. Neue, unbekannte Daten können mit einer trainierten SVM durch Berechnung von Gl. (3.2.10) klassifiziert werden, wobei das Vorzeichen $\mathrm{sgn}(y(t))$ die Klasse angibt und $|y(t)|$ den Abstand zur Hyperebene repräsentiert.

$$y(t) = \sum_{i=1}^{N_S} \alpha_i y'(i) K(\vec{s_i}, \vec{x}(t)) \tag{3.2.10}$$

$\vec{s_i}$ bezeichnet diejenigen Trainingsdatenpunkte, die als Support-Vektoren ausgewählt worden sind.

3.2.3. Klassifikation von Mehrklassenproblemen

Mit einer einzelnen SVM lassen sich nur binäre Probleme lösen. Für die akustische Modellierung in einem Spracherkenner sind jedoch überwiegend Mehrklassenprobleme zu lösen (z.B. Phonemklassifikation). Ein möglicher Ansatz zur Klassifikation von Mehrklassenproblemen ist die Verwendung von mehreren SVM. Bei der Kombination mehrerer SVM zur Lösung eines Mehrklassenproblems gibt es zwei verschiedene Grundstrategien [Hengen u. a., 2004]:

- Eine Klassifikation *einer gegen alle*: Jede SVM wird darauf trainiert, eine Klasse gegen alle übrigen zu trennen. Es wird also für jede Klasse eine SVM benötigt. Nachteil dieses Verfahrens ist, daß theoretisch eine mehrdeutiges Ergebnis möglich ist, wenn mehrere SVM "ihre" Klasse erkennen. Unter Verwendung einer Softmax-Funktion kann trotzdem eine eindeutige Klassenauftrittswahrscheinlichkeit für jede Klasse angegeben werden (vergl. Abschnitt 5.1.2). Die *eins gegen alle*-Klassifikation mit SVM kann nach [Hengen u. a., 2004] bei nichtlinearen Problemen kein optimales Klassifikationsergebnis zur Verfügung stellen, allerdings ist nur dieses Verfahren aufgrund des geringeren Zeitbedarfs gegenüber einer *eins gegen eins*-Klassifikation für die akustische Modellierung realisierbar.

- Eine Klassifikation *eins gegen eins*: Hierbei trennt eine SVM jeweils zwei bestimmte Klassen. Benötigt werden hierbei $\frac{J(J-1)}{2}$ SVM, wobei J die Anzahl der Klassen bezeichnet. Zur Berechnung von Klassifikationsergebnissen für alle Klassen müssen alle SVM ausgewertet werden, nach [Hastie u. Tibshirani, 1998] können dann ebenfalls Klassenauftrittswahrscheinlichkeiten bestimmt werden. Obwohl dieses Verfahren den Vorteil einer wesentlich genaueren Klassifikation besitzt, ist eine *eins gegen eins*-Klassifikation für die Spracherkennung in der dargestellten Form zu langsam. Eine Alternative, die bei der reinen Klassifikation mit weniger Klassifikationen auskommt, ist eine baumförmige Anordnung der SVM. Allerdings ist für einen Dekoder nach Kapitel 5 die Auftrittswahrscheinlichkeit für jede Klasse notwendig, so daß auch mit baumförmiger SVM-Struktur kein Zeitvorteil erreicht werden kann.

3.3. Ergebnisse

Beim Entwurf von Phonemklassifikatoren mit neuronalen Netzen oder SVM ist es zweckmäßig, vor der Verwendung des Netzes in den kompletten Spracherkenner ein Maß für die Qualität des Klassifikators zu haben. Eine Möglichkeit zur Berechnung eines solchen Maßes ist die *Fenster-Fehlerrate* (FFR), die nach Abschnitt 3.1.1 der Rate der Klassifikationsfehler der einzelnen Trainingsdaten entspricht. Zur Berechnung der FFR wird eine Transkription auf der Ebene der zu unterscheidenden Klassen (hier: Phoneme oder HMM-Zustände) für jedes Fenster benötigt. Im Allgemeinen wird diese Transkription mit dem Viterbi-Algorithmus aus einer Wort-Transkription erzeugt (s. Abschn. 4.7.1). Da eine solche Transkription bereits zum Training des Klassifikators vorliegen muß, entsteht an dieser Stelle kein zusätzlicher Aufwand. Die FFR kann mit

$$\mathrm{FFR} = \frac{1}{T}\sum_{t=1}^{T} |\mathrm{sgn}\left(\underset{j}{\mathrm{argmax}}\left(y'(j)(t)\right) - \underset{j}{\mathrm{argmax}}\left(y_j(t)\right)\right)| \tag{3.3.1}$$

berechnet werden. Die FFR ist also die Summe der Anzahl aller falsch klassifizierten Fenster (der Index des maximalen Netzausgangs stimmt nicht mit der Transkription überein) geteilt durch die Gesamtzahl der Fenster.

Nach [Shire, 2001] ist die FFR allerdings kein eindeutiges Kriterium für den Nutzen des Netzes im kompletten Spracherkenner, sondern allenfalls ein Hinweis. Beim Vergleich der Tabellen 3.1 und 3.2 mit den Tabellen 5.2 und 5.4 wird bestätigt, daß die FFR keinen eindeutigen Schluß auf die Wortfehlerrate des Spracherkennungssystems zuläßt. Aus diesem Grund ist in [Stadermann u. Rigoll, 2003a] die *Phonem-Fehlerrate* (PFR) als Alternative untersucht worden. Bei der PFR wird der maximale Index $\underset{\rho_j}{\mathrm{argmax}}(\frac{Pr(\rho_j|\vec{x})}{Pr(\rho_j)})$ (Index des maximalen Netzausganges dividiert durch seine *a priori*-Wahrscheinlichkeit) berechnet und seinem Phonemsymbol zugeordnet und dieses Symbol dann ausgegeben. Es entsteht also eine Phonemfolge, wobei gleiche Phoneme in Folge zu einem Ausgabesymbol zusammengefaßt werden. Der gesamte Prozeß ist in Abb. 3.3.1 illustriert. Die so entstandene Phonemfolge kann nun mit der Original-Phonemtranskription verglichen werden, wobei die gleichen Methoden (Levenstein-Distanz) wie in den Abschnitten 4.8

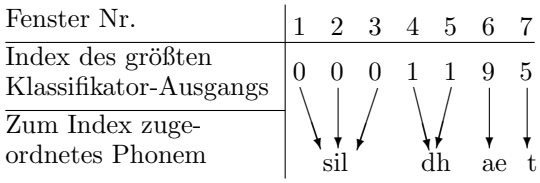


Abbildung 3.3.1.: Entstehung der Phonemfolge zur Berechnung der PFR

und 5.5 zur Auswertung von Wortfehlerraten zum Einsatz kommen. Es entstehen also analog zu Abschnitt 4.8 Auslassungen (D), Ersetzungen (S) und Einfügungen (I), die in Tabelle 3.3 zusammen mit der *Accuracy* (Acc) und der *Correctness* (Cor) angegeben werden. Die PFR definiert sich dann als PFR $= \frac{D}{T}$, wobei T die Gesamtzahl der untersuchten Fenster ist.
Bei der theoretischen Erläuterung der Klassifikator-Trainingsverfahren der bisherigen Abschnitte bleibt der Einfluß des Wertebereichs der Eingangsdaten auf die Qualität des trainierten Netzes unberücksichtigt. Nach [Joost u. Schiffmann, 1998] kann durch eine Normierung der Trainingsdaten ein schnelleres Training (bessere Ergebnisse mit weniger Iterationen) des Netzes erreicht werden. Der normierte Eingangswert $x_n(t)$ ergibt sich zu

$$x_n(t) = \frac{f_n(t) - \bar{f}_n}{\sigma_n} \ , \tag{3.3.2}$$

wobei $\bar{f}_n = \sum_{t=1}^{T} f_n(t)$ der Mittelwert und $\sigma_n = \sqrt{\frac{1}{T-1} \sum_{t=1}^{T} f_n(t) - \bar{f}_n}$ die Standardabweichung über alle Trainingsbeispiele bezeichnet. Es ergeben sich dann mittelwertfreie Daten mit einer Varianz von 1. Beim Test der Netze ist diese Annahme nur noch näherungsweise erfüllt, da Mittelwert und Standardabweichung der Testbeispiele nicht den errechneten Werten (aus den Trainingsdaten) entsprechen. Die Normierung nach Gl. (3.3.2) wird bei allen vorgestellten Klassifikatoren verwendet.

3.3.1. Ergebnisse mit neuronalen Netzen

Die Tabellen 3.1 bis 3.5 zeigen die FFR verschiedener neuronaler Netze. Variiert werden neben dem Netztyp (MLP, bzw. RNN), die Anzahl der Eingänge und Neuronen, sowie die Art der Gewichtsneuberechnung. Die Angabe nach dem Netztypkürzel bezeichnet die Anzahl der Eingangswerte, die Anzahl der versteckten bzw. rückgekoppelten Neuronen, sowie die Anzahl der Ausgangsneuronen. Der Parameter τ gibt die Verzögerung des Ausgangsvektors gegenüber dem Eingangsvektor an (s. Abschnitt 3.1.3), $2m$ bezeichnet die Anzahl an zusätzlichen Kontextfenstern neben dem aktuellen Fenster. Die Gesamtzahl der trainierbaren Parameter der jeweiligen Netze ist in der 4. Spalte (*#Param. NN*) abzulesen. Netze mit 47 Ausgängen klassifizieren 45 Phoneme und 2 Pausenmodelle, Netze mit 139 Klassen unterscheiden 139 HMM-Zustände der insgesamt 47 Modelle (das *sp*-Pausenmodell besitzt nur 1 aktiven Zustand). Die NN für die AURORA2-Daten verwenden neben Phonemen auch aus Ganzwortmodellen generierte Klassen (s. die Erläuterung zu Tabelle 3.5). Datengrundlage für die folgenden Tabellen sind 724 Sätze aus dem

Trainingsset *si-84* der WSJ0-Datenbasis, bzw. 840 Sätze des AURORA2-Trainingssets mit Geräuschen (s. Anhänge B.2 bzw. B.3), die nicht für das Klassifikatortraining benutzt worden sind. Die verwendeten Merkmalsvektoren für die WSJ bestehen aus 12 MFCC mit Energie und dynamischen Merkmalen (insgesamt 39 Komponenten). Bei der AURORA2-Datenbasis werden alternativ auch 9 RASTA-Merkmale mit Energie und dynamischen Merkmalen verwendet (RASTA30, vergl. Abschnitt 2.2.3).

NN	**m**	**#Param. NN**	**FFR**
MLP273-1000-47	3	321047	27,20%
MLP117-1000-47	1	165047	29,25%
MLP39-1000-47	0	87047	34,79%
MLP273-500-47	3	160547	28,96%
MLP117-500-47	1	82547	30,97%
MLP39-500-47	0	43547	36,02%
MLP273-1000-139	3	413139	34,08%

Tabelle 3.1.: FFR verschiedener MLPs (Neuberechnung der Gewichte mit Momentum-Erweiterung nach Gl. (3.1.13))

Aus der FFR ist ein Trend erkennbar, MLPs mit möglichst vielen versteckten Knoten und möglichst viel Kontext im Eingangsvektor zu benutzen. Die FFR des MLP273-1000-139 ist nicht direkt mit den anderen MLP vergleichbar, da hier 139 Klassen zu unterscheiden sind.

NN	τ	**#Param. NN**	**FFR**
RNN39-400-47	3	196680	25,66%
RNN39-300-47	3	117980	28,20%
RNN39-300-47	0	117980	29,22%
RNN39-400-139	3	237160	34,58%

Tabelle 3.2.: FFR verschiedener RNN (Neuberechnung der Gewichte mit dem RPROP-Verfahren)

Die reine Betrachtung der FFR läßt zunächst auf eine sehr gute Qualität der RNN gegenüber den MLP schließen. Die Ergebnisse aus Kapitel 5 bestätigen allerdings die Schlüsse aus [Shire, 2001], daß eine gute FFR nicht unbedingt auf ein gutes Spracherkennungssystem schließen läßt. In Tabelle 3.3 wird daher die PFR der besten Netze aus den obigen Tabellen miteinander verglichen. Angegeben sind neben der PFR die Anzahl der Auslöschungen (D), der Ersetzungen (S) und der Einfügungen (I), sowie die Genauigkeit (Acc) und *correctness* (Cor). Werden die Ergebnisse aus Tabelle 3.3 mit

den entsprechenden Fehlerraten der Gesamtsysteme aus den Tabellen 5.2 und 5.4 verglichen, so scheint die Anzahl der Auslassungen (D) bei Berechnung der PFR wesentlich die Qualität des Gesamtsystems zu bestimmen, während die Anzahl der Einfügungen weitgehend irrelevant ist. Ebenfalls abzulesen ist hier das Bestreben der RNN, ihren Zustand beizubehalten - charakterisiert durch die geringe Anzahl an Einfügungen und die erhöhte Anzahl Auslassungen. Eine Integration der PFR in den Trainingsalgorithmus ist aufgrund der nichtlinearen Berechnung bisher nicht gelungen. Um die Qualität der RNN

NN	**D**	**S**	**I**	**Acc**	**Cor**	**PFR**
MLP273-1000-47	3465	15695	59482	-10,77%	73,01%	4,9%
RNN39-400-47	5715	13405	30904	29,54%	73,07%	8,0%
MLP273-1000-139	16094	43708	51415	40.32%	67.91%	8,6%
RNN39-400-139	18684	42680	36839	47.30%	67.07%	10,0%
RNN39-400-139+Geschlecht	18216	35805	31101	54,32%	71,01%	9,8%

Tabelle 3.3.: PFR verschiedener NN

weiter zu verbessern, werden die Netze mit zusätzlichen Aufgaben in einer Struktur nach Abschnitt 3.1.5 trainiert. Neben der Klassifikation von Phonemen oder HMM-Zuständen wird jeweils eine zusätzliche Aufgabe klassifiziert. Im Rahmen dieser Arbeit sind das Geschlecht des Sprechers, verallgemeinerte Phonemklassen und Grapheme (Buchstaben) [Killer u. a., 2003] als zusätzliche Aufgaben trainiert worden. Alle Zusatzaufgaben werden genauso, wie die Hauptaufgabe, für jedes Fenster trainiert bzw. klassifiziert. Die Anzahl an trainierbaren Parametern ist durch die Einführung der zusätzlichen Aufgaben angestiegen, da diese Parameter aber in der Erkennungsphase nicht benutzt werden, bleibt die effektive Größe der Netze unverändert zum RNN39-400-139 aus Tabelle 3.2.

NN	**Zusatzaufgabe**	**FFR Haupt**	**FFR Neben**
RNN39-400-47	Geschlecht	26,86%	7,55%
RNN39-400-47	Graphem	29,31%	37,23%
RNN39-400-47	Phonemklassen	32,17%	22,62%
RNN39-400-139	Geschlecht	34,70%	4,77%
RNN39-400-139	Graphem	34,36%	32,93%
RNN39-400-139	Phonemklassen	34,69%	17,88%

Tabelle 3.4.: FFR verschiedener RNN (Neuberechnung der Gewichte mit dem RPROP-Verfahren) mit zusätzlichen Aufgaben

Die Tabelle 3.5 zeigt das Verhalten von NN unter Benutzung der AURORA2-Trainingsdaten mit Hintergrundgeräusch (vergl. Abschnitt 7.3). Die FFR ist mit 840 Sätzen des Trainingssets berechnet worden.

NN	#Param. NN	Merkmale	Zielwerte	m/τ	FFR
MLP210-500-48	129548	RASTA30	48 P-Ph	3	32,59%
RNN30-200-48	57288	RASTA30	48 P-Ph	3	29,30%
MLP273-500-48	161048	MFCC39	48 P-Ph	3	36,51%
RNN39-200-48	59520	MFCC39	48 P-Ph	3	30,92%
MLP210-500-47 (22)	129047	RASTA30	22 Phoneme	3	19,98%
RNN30-200-47 (22)	57057	RASTA30	22 Phoneme	3	28,75%
MLP273-500-47 (22)	160547	MFCC39	22 Phoneme	3	19,56%
RNN39-200-47 (22)	59280	MFCC39	22 Phoneme	3	17,00%

Tabelle 3.5.: FFR verschiedener NN mit AURORA2-Daten (MLP: Momentum-Neuberechnung der Gewichte, RNN: RPROP-Neuberechnung)

Da bei den AURORA2-Daten nur 20 Phoneme und 2 Pausenmodelle vorkommen, klassifizieren die phonembasierten NN mit 47 Ausgängen effektiv nur 22 Klassen, was die gute FFR erklärt. Die NN mit 48 Ausgängen sind auf 48 Pseudo-Phoneme (*P-Ph*) trainiert, die aus Ganzwortmodellen zusammengefaßt worden sind (s. Abschnitt 7.3). Auch hier schneiden die meisten RNN sichtbar besser ab als MLP, trotzdem ist auch in diesem Fall der Gesamterkenner mit einem MLP der bessere (vergl. Abschnitt 7.3.2).

3.3.2. Ergebnisse mit Support-Vektor-Maschinen

Die Ergebnisse der SVM-Klassifikatoren sind mit 1000 Sätzen der AURORA2 Trainingsets (ohne Hintergrundgeräusch) berechnet, die für das Training der SVM nicht verwendet worden sind. Die Notation der SVM-Klassifikatoren (1. Spalte aus Tabelle 3.6) bezieht sich auf die Anzahl der SVM (erste Zahl, identisch mit der Anzahl der Klassen), sowie der Anzahl der Sätze aus dem AURORA2-Trainingset (zweite Zahl der 1. Spalte). 13 SVM klassifizieren die 11 Zahlwörter, die in der AURORA2-Datenbasis vorkommen, plus 2 Pausenmodelle. 23 SVM klassifizieren Worthälften, eine detaillierte Erklärung hierzu findet sich im Abschnitt 5.6.
FFR ohne Sigmoid bezeichnet die Fenster-Fehlerrate, bei der nur der Wert der einzelnen SVM ausgewertet wird und diejenige Klasse gewinnt, deren SVM den größten Abstand zur Trennebene aufweist. *FFR* ist dann die Fehlerrate mit Anwendung der Sigmoid-Funktionen zur Erzeugung von Wahrscheinlichkeiten für jede SVM (vergl. Abschnitt 5.1.2), hierbei wird der Index der SVM mit der größten Wahrscheinlichkeit mit der Referenz verglichen. Der Merkmalsvektor besteht aus 9 RASTA-Merkmalen, sowie der Energie und dynamischen Merkmalen (RASTA30, s. Abschnitt 2.3). Für alle SVM ist ein Gauß-Kernel gewählt worden, da dieser sich in [Salomon u. a., 2002] für die Klassifikation von Sprachdaten als beste Wahl herausgestellt hat. Weiterhin ist in [Salomon, 2001] zusätzliche Kontextinformation für die Phonemklassifikation mit SVM vorteilhaft eingesetzt worden, daher sind auch hier die Eingangsvektoren der SVM nach Abschnitt 3.1.3 um zeitlichen Kontext (insgesamt $2m + 1$ Fenster) erweitert.

SVM	**Anzahl SV**	**Klassen**	m	**FFR ohne Sigmoid**	**FFR**
SVM13-2000	112297	13	0	-	17,90%
SVM23-2000	106009	23	1	-	16,63%
SVM23-3000	184504	23	1	14,78%	14,75%
SVM23-6000	273578	23	1	12,67%	12,90%

Tabelle 3.6.: FFR verschiedener SVM Konfigurationen mit RASTA30-Merkmalsvektor auf der AURORA2-Datenbasis

Die 13 wortbasierten SVM schneiden am schlechtesten ab, da kein zeitlicher Kontext vorliegt. Die Aufgabe, das Wort aus einem Merkmalsvektor als Repräsentant für 25ms des Sprachsignals zu erkennen, ist schwierig und birgt eine große Verwechslungsgefahr (*t*wo, eigh*t* - o*ne*, ni*ne*). Durch Kontext und den Übergang zur Klassifikation von halben Wörtern (23 Klassen) kann das Problem offensichtlich etwas günstiger gestaltet werden. Eine Erhöhung der Anzahl der Trainingssätze führt zu weiterer Verbesserung, allerdings steigt hierdurch auch die Anzahl der Support-Vektoren, was das System verlangsamt (jeder Support-Vektor hat 90 Komponenten bei $m = 1$).
Die Parameter der Sigmoid-Funktion sind mit allen 8440 Sätzen bestimmt worden, also ist hier (zumindest für die Sigmoid-Parameter) ein Teil der Daten re-klassifiziert worden, trotzdem hat sich das System SVM23-6000 durch Verwendung der Sigmoid-Funktionen leicht verschlechtert, während sich das System SVM23-3000 leicht verbessert. Bei der Kombination mit HMM nach Kapitel 5 sind Klassifikationsergebnisse vonnöten, die als Wahrscheinlichkeiten aufgefaßt werden können, so daß auf die Sigmoid-Funktionen im kompletten Erkenner nicht verzichtet werden kann.

4. Spracherkennung mit Hidden-Markov Modellen

Hidden-Markov-Modelle (HMM) sind ein zentraler Bestandteil des akustischen Modells. Dieses Kapitel führt Schritt für Schritt durch den kompletten Prozeß der automatischen Spracherkennung mit HMM und parametrischen Wahrscheinlichkeitsdichtefunktionen für den Merkmalsraum. Der Schwerpunkt liegt dabei auf der Beschreibung eines gängigen Trainingsverfahrens für die Parameter eines HMM, dem Baum-Welch-Algorithmus. Anschließend werden die in dieser Arbeit verwendeten Verfahren der Sprachmodellierung und der Dekodierung (Erkennung) einer gesprochenen Äußerung vorgestellt.

4.1. Spracherkennung mit einem statistischen Modell

Die bisher beste Methode, die statistischen Variationen eines Sprachsignals zu erfassen, stellen HMM dar. Hierbei werden Sprachuntereinheiten (Wörter, Phoneme[1]) durch einzelne Modelle m_{nu} beschrieben, deren Aneinanderreihung $M_n = (m_{n1}, \ldots, m_{nU})$ dann z.B. einen gesamten Satz abbilden. Das Prinzip der Erkennung ist dabei, daß zu jedem Zeitpunkt alle möglichen Modellkombinationen miteinander konkurrieren und die Modellfolge gewinnt, welche das unbekannte Signal mit der größten Wahrscheinlichkeit generiert. Formal ausgedrückt, sollte idealerweise das Maximum der *a posteriori*-Wahrscheinlichkeiten für jede Modellfolge M_n berechnet werden, wobei dann der Index n^* des besten Modells in Gl. (4.1.1) als Schätzung für die tatsächliche Äußerung zurückgeliefert wird.

$$n^* = \underset{n}{\operatorname{argmax}} \{\Pr(M_n|X)\} \tag{4.1.1}$$

$X = \vec{x}(1), \ldots, \vec{x}(T)$ bezeichnet die Folge von Merkmalvektoren für das zu erkennende Sprachsignal. Da diese Wahrscheinlichkeit in dieser Form schlecht zu berechnen ist, wird auf Gl. (4.1.1) der Satz von Bayes angewandt und es ergibt sich

$$n^* = \underset{n}{\operatorname{argmax}} \{\Pr(M_n|X)\} = \underset{n}{\operatorname{argmax}} \left\{ \frac{p(X|M_n)\Pr(M_n)}{p(X)} \right\} = \underset{n}{\operatorname{argmax}} \{p(X|M_n)\Pr(M_n)\} \tag{4.1.2}$$

Die Wahrscheinlichkeitsdichte $p(X|M_n)$ kann durch das akustische Modell, eine Kette von HMM, berechnet werden (s. Bild 4.2.1). Die Größe $\Pr(M_n)$ ist die *a priori*-Wahrscheinlichkeit für das Auftreten der Modellsequenz M_n und kann durch ein Sprachmodell (vergl. Abschnitt 4.6) beschrieben werden. Eine implizite Annahme ist dabei, daß

[1] kleinste bedeutungsunterscheidende Einheit der Sprache

das Sprachmodell getrennt vom akustischen Modell berechnet werden kann. Die Größe $p(X)$ ist die Wahrscheinlichkeitsdichte der akustischen Vektorsequenz und kann bei der Maximierung bezüglich Variation von n als Konstante angesehen werden.
Die Parameter des akustischen Modells können aus Trainingsdaten geschätzt werden, sofern einige Vereinfachungen und Annahmen, wie im nächsten Abschnitt beschrieben, getroffen werden.

4.2. Hidden-Markov Modelle für die akustische Modellierung

In diesem Abschnitt wird eine Notation zur mathematischen Beschreibung der HMM eingeführt, sowie Vereinfachungen, die die HMM für die akustische Modellierung handhabbar machen, vorgestellt. Bei einem Markovprozeß geht man von einer zeitlichen Abfolge von Zuständen aus

$$Q = (q(t=1), \ldots, q(t=T)) \,; q(t) \in \mathbb{Q} \tag{4.2.1}$$

deren einzelne Zustände der endlichen Menge $\mathbb{Q} = \{q_1, \ldots, q_I\}$ angehören. Die Ausgabe der einzelnen Zustände einer Hidden-Markovkette ist nicht deterministisch, sondern durch eine Zufallsvariable bestimmt. Charakterisiert wird die Ausgabe durch die Wahrscheinlichkeitsdichte für einen Merkmalsvektor $\vec{x}$ gegeben alle bisherige Beobachtungen $X(1, \ldots, t-1)$, das Modell M_n, sowie den aktuellen und vorausgegangenen Zustand $q(t)$, bzw. $q(t-1)$:

$$p(\vec{x}(t)|q(t), q(t-1), X(1, \ldots, t-1), M_n)$$

Daraus folgt, daß bei Beobachtung der Zustandsausgaben nicht mehr eindeutig auf die Abfolge der Zustände Q zu schließen ist, die Abfolge der Zustände ist also versteckt[2]. Um eine effiziente Berechnung der Modellparameter zu ermöglichen, sind eine Reihe von

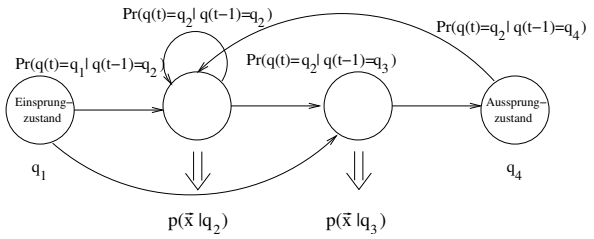


Abbildung 4.2.1.: Beispiel für ein Hidden-Markov-Modell mit den beschriebenen Vereinfachungen

Annahmen [Bourlard u. Morgan, 1994] notwendig:

1. Der zu modellierende Prozeß ist eine Markovkette 1. Ordnung

$$\Pr(q(t)|q(t-1), X(1, \ldots, t-1), M_n) = \Pr(q(t)|q(t-1), M_n) \tag{4.2.2}$$

Einzelheiten und Konsequenzen dieser Annahme werden weiter unten ausgeführt.

[2] engl. *hidden*

2. Die Ausgaben $b_i(t)$ der einzelnen Zustände der HMM hängen nur von den Zuständen ab und sind modellunabhängig.

$$b_i(t) = p(\vec{x}(t)|q(t) = q_i) \tag{4.2.3}$$

In Annahme 2 kann auch die Abhängigkeit vom Vorzustand behalten werden, man erhält dann Ausgaben, die von den Übergängen zwischen den Zuständen abhängig sind [Jelinek, 1976], dies bringt allerdings zusätzlichen Rechenaufwand und wird hier nicht weiter verfolgt. Die Vereinfachungen führen also zu einem Hidden-Markov Modell 1. Ordnung, das einen kausalen, stationären und einfachen[3] Prozeß abbilden kann. Eine grafische Darstellung dieses HMM zeigt Abbildung 4.2.1. Aus Kapitel 2 ist bereits bekannt, daß ein Sprachsignal weder stationär noch einfach ist, der Kompromiß zwischen Rechenaufwand und Genauigkeit des Modells hat jedoch zu den obigen Vereinfachungen geführt.
Der Übergang von einem Zustand zum nächsten ist beim Markovprozeß 1. Ordnung durch die Übergangswahrscheinlichkeit

$$a_{il} = Pr\left(q(t) = q_i|q(t-1) = q_l\right) \tag{4.2.4}$$

ausgedrückt (vergl. Annahme 1). Die Übergangswahrscheinlichkeiten, die nur von den Zuständen $q(t) = q_i$ und $q(t-1) = q_l$ abhängen und zeitunabhängig sind, können in einer Übergangsmatrix

$$\mathbf{A} = \begin{pmatrix} a_{11} & \cdots & a_{I1} \\ \vdots & \ddots & \vdots \\ a_{1I} & \cdots & a_{II} \end{pmatrix} \tag{4.2.5}$$

zusammengefaßt werden. Hierbei gelten für die Wahrscheinlichkeiten a_{il} die Stochastizitätsbedingungen

$$a_{il} \geq 0, \sum_l a_{il} = 1 \tag{4.2.6}$$

Für Einsprünge in die Markovkette können nach [Schukat-Talamazzini, 1995] Einsprungwahrscheinlichkeiten für die einzelnen Zustände der Kette bestimmt werden. Eine für die Verarbeitung im Rechner effizientere Methode ist das Hinzufügen von Ein- und Ausprungzuständen am Anfang bzw. Ende der Kette [Young u. a., 2000]. Damit ist festgelegt, daß die Markovkette immer am Anfangszustand beginnt und nur über den Endzustand verlassen werden kann (vergl. Bild 4.2.1).
In der akustischen Modellierung werden fast ausschließlich vorwärtsgerichtete[4] Markovketten verwendet, daraus folgt eine Dreiecksform der Übergangsmatrix $\mathbf{A}$. Übliche Topologien sind Links-Rechts-Ketten (Bild 4.2.2), Bakisketten (Bild 4.2.3) und lineare Ketten (Bild 4.2.4). Die Experimente dieser Arbeit sind ausschließlich mit linearen Zustandsketten durchgeführt worden.

Die Übergangsmatrix $\mathbf{A}$ beschreibt zusammen mit den Ausgabedichten $b_i(t)$ das HMM vollständig, das Set von Parametern für ein HMM mit I Ausgabedichten wird allgemein mit $\lambda = (\mathbf{A}, b_1, \ldots, b_I)$ bezeichnet.

[3] Der Prozeß hängt nur vom unmittelbar vorausgehenden Zustand der Markovkette ab.
[4] nur Übergänge in Richtung des Endzustandes möglich

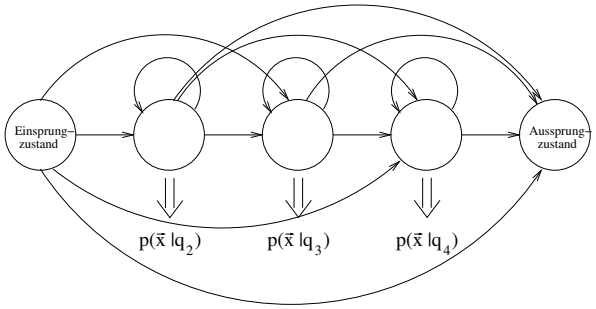


Abbildung 4.2.2.: Links-Rechts-Modell mit drei aktiven Zuständen, der Anfangs- und der Endzustand haben keine Ausgabe

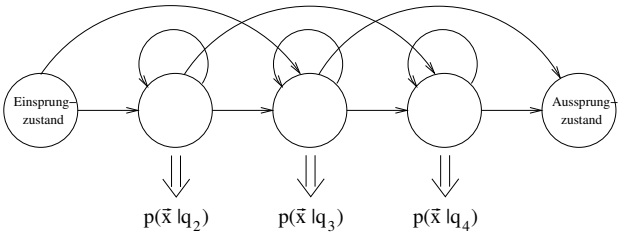


Abbildung 4.2.3.: Bakis-Modell mit drei aktiven Zuständen, der Anfangs- und der Endzustand haben keine Ausgabe

4.3. Modellierungen der Zustandsausgabe

Für die Ausgabedichte $b_i(t)$ ist ein beliebiges Modell denkbar, im Folgenden sind drei häufig verwendete Modellarten beschrieben, die sich durch eine gewisse Flexibilität und eine einfache Berechenbarkeit auszeichnen. In allen Fällen setzt sich die Ausgabedichte aus einer Summe von elementaren Wahrscheinlichkeitsdichten zusammen:

$$b_i(t) = \sum_{j=1}^{J} c_{ij} b_{ij}(t) \tag{4.3.1}$$

Zu beachten ist hierbei, daß die Normierungsbedingung

$$\int_{\mathbb{R}^N} \cdots \int b_i(\vec{x}) d\vec{x} = 1 \tag{4.3.2}$$

eingehalten wird. Dies erfordert $\sum_{j=1}^{J} c_{ij} = 1$. Die Festlegung einer Elementardichte (z.B. eine Gaußfunktion oder eine diskrete Wahrscheinlichkeit) erlaubt eine einfache mathematische Formulierung und eine einfache Bestimmung der Dichteparameter durch Maximierung der Produktionswahrscheinlichkeit des Modells, schränkt jedoch die Modellierungsfähigkeit durch die getroffenen Annahmen der Elementardichte deutlich ein. Die Verwendung eines Klassifikators (vergl. 3) zur Modellierung der Ausgabedichte (ausführlich in Kapitel 5 dargestellt) versucht, diese Limitierung zu umgehen.

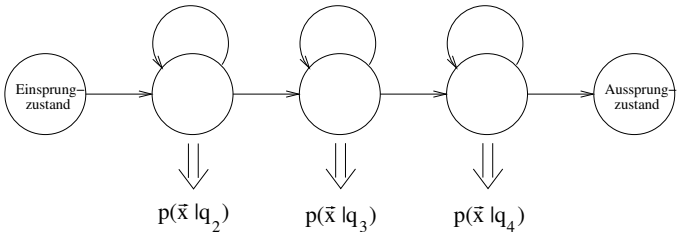


Abbildung 4.2.4.: Lineares Modell mit drei aktiven Zuständen, der Anfangs- und der Endzustand haben keine Ausgabe

4.3.1. Diskrete Modelle

Diskrete Modelle erhält man, indem die Ausgabedichte (Gl. (4.3.1)) durch J verschiedene diskrete Wahrscheinlichkeiten ersetzt wird. Dieser Ansatz setzt dann implizit voraus, daß eine Zuordnung der kontinuierlichen Vektoren $\vec{x}$ zu den J Klassen des diskreten Modells existiert. Eine solche Zuordnung geschieht z.B. durch Vektorquantisierung [Rigoll, 1994b]. Die Wahrscheinlichkeitsdichte b_i läßt sich dann als Summe von gewichteten Diracstößen schreiben, wobei die Gewichtungen die diskreten Wahrscheinlichkeitswerte der jeweiligen Klasse sind:

$$b_i(t) = \sum_{j=1}^{J} c_{ij}\delta(\vec{x}(t) - \vec{\kappa}_j) \tag{4.3.3}$$

Die Vektoren $\vec{\kappa}_j$ beschreiben das Set von J Klassen des Vektorquantisierers, sie werden unabhängig von den HMM und global für alle Modelle bestimmt, die diskreten Wahrscheinlichkeiten c_{ij} können mit dem Baum-Welch Algorithmus (Abschnitt 4.4) trainiert werden.

4.3.2. Semi-kontinuierliche Modelle

Eine detailliertere Modellierung der Ausgabedichten erhält man, wenn die Menge von diskreten Wahrscheinlichkeiten aus Abschnitt 4.3.1 durch eine Menge von kontinuierlichen Gauß-Funktionen, sogenannten Mixturen[5], ersetzt wird. Die Ausgabedichte ergibt sich dann zu

$$b_i(t) = \sum_{j=1}^{J} c_{ij} \frac{1}{(2\pi|\mathbf{\Sigma_j}|)^{\frac{N}{2}}} \exp\left(-\frac{1}{2}\left(\vec{x}(t) - \vec{\mu_j}\right)^T \mathbf{\Sigma_j}^{-1} \left(\vec{x}(t) - \vec{\mu_j}\right)\right) = \sum_{j=1}^{J} c_{ij} \mathrm{G}(\vec{\mu}_j, \mathbf{\Sigma_j}) \tag{4.3.4}$$

In Kapitel 2 ist dargestellt, daß die Komponenten des Merkmalsvektors $\vec{x}$ näherungsweise unkorrelliert sind. Aus diesem Grund ist die Annahme einer voll besetzten Kovarianzmatrix im Modell nicht notwendig und kann vereinfacht werden: Werden nur die Abhängigkeiten der Komponenten mit sich selber berücksichtigt, so ist nur die Hauptdiagonale

[5] eine Mixtur ist eine Komponente der Gesamtwahrscheinlichkeitsdichte und hier synonym für eine einzelne Gaußfunktion

der Kovarianzmatrix besetzt und die rechenintensive Invertierung der Kovarianzmatrix reduziert sich zu einer Division:

$$b_i(t) = \sum_{j=1}^{J} c_{ij} \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi\sigma_{n,j}^2}} \exp\left(-\frac{(x_n(t)-\mu_{n,j})^2}{2\sigma_{n,j}^2}\right) = \sum_{j=1}^{J} c_{ij} \mathrm{G}(\vec{\mu}_j, \vec{\sigma}_j^2)\,, \qquad (4.3.5)$$

hierbei bezeichnet $\sigma_{n,j}^2$ ein Element der Hauptdiagonalen der j-ten Gaußfunktion. Da alle HMM-Zustände Zugriff auf alle Gaußfunktionen haben, spricht man auch von verbundenen Mixturen[6].

4.3.3. Kontinuierliche Modelle

Eine noch genauere Modellierung, als in Abschnitt 4.3.2 beschrieben, entsteht, wenn man nicht mehr ein für alle Zustände gemeinsames Set von Gaußfunktionen bereitstellt, sondern jedem HMM-Zustand seine eigenen Gaußdichten (Mixturen) zur Verfügung stellt. Wird die obige Vereinfachung der diagonalen Kovarianzmatrix beibehalten, so ergibt sich dann für eine Ausgabedichte eines Zustandes q_i:

$$b_i(t) = \sum_{j=1}^{J} c_{ij} \prod_{n=1}^{N} \frac{1}{\sqrt{2\pi\sigma_{n,ij}^2}} \exp\left(-\frac{(x_n(t)-\mu_{n,ij})^2}{2\sigma_{n,ij}^2}\right) = \sum_{j=1}^{J} c_{ij} \mathrm{G}(\vec{\mu}_{ij}, \vec{\sigma}_{ij}^2) \qquad (4.3.6)$$

Durch diese Erweiterung wird die Anzahl an Parametern meist vergrößert, da nun für jeden Zustand ein komplettes Set an Mixturen zur Verfügung steht und trainiert werden muß. Außerdem ist damit zu rechnen, daß akustisch ähnliche Zustände, die sich im System nach Abschnitt 4.3.2 die Gaußfunktionen geteilt haben, nun jeder für sich diese Funktionen trainieren müssen. Da die Anzahl an Trainingsbeispielen pro Parameter damit abnimmt, ist ein solches System nur geeignet, falls eine genügend große Menge an Trainingsdaten bereitsteht. Um die Qualität selten auftauchender Modelle zu verbessern, bietet es sich hier dann wiederum an, einzelne Wahrscheinlichkeitsdichten oder Gaußfunktionen mit mehreren HMM-Zuständen zu verbinden.

4.4. Training der freien Parameter

Die freien Parameter des akustischen Modells M sind im Parameterset eines HMMs λ zusammengefaßt (vergl. Abschnitt 4.2), das aus Übergangsmatrix und Ausgabedichten besteht. Bei der statistischen Modellierung ist ein mögliches Ziel für die Einstellung dieser Parameter, die (logarithmierte) Wahrscheinlichkeitsdichte $\mathfrak{L}$ für das Erzeugen einer Merkmalsvektorfolge X zu maximieren (*engl.*: Maximum Likelihood principle (ML)):

$$\mathfrak{L} = \log\left(p(X|M)\right) = \log\left(p(X|\lambda)\right) \stackrel{!}{=} max. \qquad (4.4.1)$$

[6] *engl.*: Tied-Mixture

Der Logarithmus ist streng monoton, verändert also nicht das Ergebnis, vereinfacht aber die nachfolgenden Berechnungen Das Optimieren durch Maximieren der Produktionswahrscheinlichkeit führt allerdings nur dann zu global optimalen Parametern, wenn unendlich viele Trainingsbeispiele vorhanden sind. Aus diesem Ziel kann durch Betrachtung der Entropie zwischen Modellzuständen und Merkmalsvektoren auch eine allgemeinere Formulierung hergeleitet werden, die sowohl das ML-Prinzip, als auch eine Maximierung der Transinformation zwischen Modellzustandsfolge und Merkmalsvektorfolge enthält [Rigoll, 1990]. Letzteres Prinzip ist diskriminativ, führt also neben der Maximierung der Wahrscheinlichkeit auch zur Minimierung konkurrierender Modelle. Das diskriminative Training [Reichl, 1996] der HMM-Parameter ist jedoch nur mit wesentlich mehr Aufwand zu realisieren und wird daher bei der Kombination eines diskriminativ trainierten Klassifikators mit HMM (s. Kapitel 5) nicht verwendet. Da der Klassifikator die Berechnung der Ausgabedichte zum großen Teil übernimmt, ist beim diskriminativen Training der verbliebenen HMM-Parameter kein großer Gewinn gegenüber einem ML-Training zu erwarten. Im Folgenden wird deshalb nur auf das Training der freien Parameter nach dem ML-Prinzip eingegangen.
Eine Umformulierung von Gl. (4.4.1) ergibt für die gesuchten, optimalen Parameter $\hat{\lambda}^*$

$$\hat{\lambda}^* = \underset{\lambda}{\operatorname{argmax}}\{\mathfrak{L}(X|\lambda)\} \tag{4.4.2}$$

Zum Training liegen im allgemeinen Wortfolgen bzw. Modell- oder Modellzustandsfolgen, sowie die dazugehörigen akustischen Beobachtungen als Merkmalsvektorsequenzen vor (vergl. Bild 4.4.1). Vor der Neuschätzung der Parameter ist zunächst eine Zu-

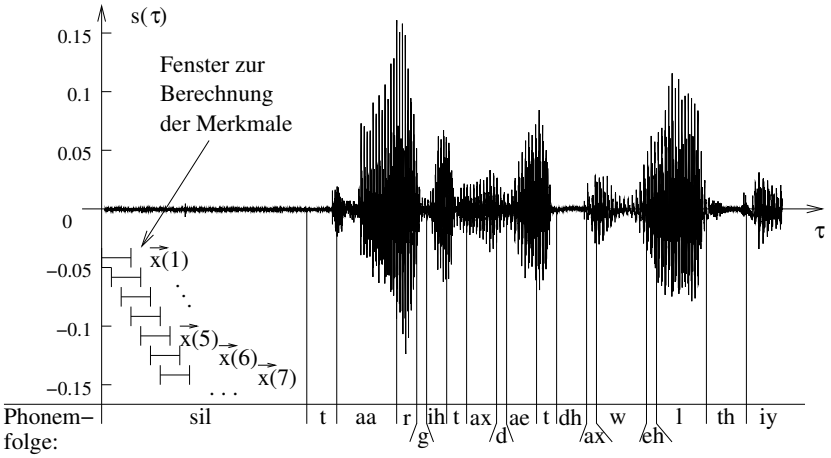


Abbildung 4.4.1.: Beispiel einer segmentierten Modellzustandsfolge der Worte *targeted at the wealthy*, die Folge der Merkmalsvektoren ist angedeutet

ordnung zwischen diesen beiden Folgen zu finden. Ferner stellt sich das Problem, daß die Parameter λ zur Berechnung der *Likelihood* $\mathfrak{L}$ (vergl. Gl. (4.4.1)) noch unbekannt sind, andererseits aber die Maximierung von $\mathfrak{L}$ zur Bestimmung der optimalen Parameterschätzwertes $\hat{\lambda}^*$ benötigt wird [Schukat-Talamazzini, 1995]. Als Lösung bietet sich

hier der *EM*-Algorithmus an, der iterativ abwechselnd den Erwartungswert der zu optimierenden Größe aus den alten Parametern berechnet (E-Schritt) und daraus eine verbesserte Schätzung der Parameter liefert (M-Schritt). Die nachfolgend beschriebene Formulierung des EM-Algorithmus für HMM ist als *Baum-Welch* Algorithmus [Baum, 1972] bekannt.

4.4.1. Hilfsgrößen für den EM-Algorithmus

Für die nachfolgenden Betrachtungen geht man von einer bekannten Folge von Merkmalsvektoren $\vec{x}(1), \ldots, \vec{x}(T)$, sowie einer bekannten Folge von HMM-Zuständen $q_1, \ldots, q_i, \ldots, q_I$ (deren Zuordnung zu den Zeitpunkten $1, \ldots, T$ zunächst unbekannt ist) aus.
Zunächst wird die Verbundwahrscheinlichkeitsdichte $\alpha_i(t) = p\left(X(1, \ldots, t), q(t) = q_i|\lambda\right)$ für das Aufhalten im Zustand q_i zum Zeitpunkt t, bei Durchlaufen der Merkmalsvektorfolge von $\vec{x}(1)$ bis $\vec{x}(t)$ definiert. Vorteil dieser Definition ist die Möglichkeit, diese Größe rekursiv für jeden Zeitschritt zu berechnen. Nach Durchlaufen aller Zeitschritte ergibt sich dann die Wahrscheinlichkeitsdichte $p(X|\lambda) = \alpha_I(T)$ für die Erzeugung der gesamten Merkmalsfolge gegeben das Modell. Die Rekursionsvorschrift lautet [Young u. a., 2000]:

$$\alpha_1(1) = 1 \ , \ \alpha_i(1) = a_{1i} b_i(1) \text{ für } 1 < i < I \tag{4.4.3}$$

$$\alpha_i(t) = \left[\sum_{l=2}^{N-1} \alpha_l(t-1) a_{li} \right] b_i(t) \ \text{ für } 1 < t \leq T; 1 < i < I \tag{4.4.4}$$

I ist hier die Anzahl von HMM-Zuständen[7]. In der Literatur [Schukat-Talamazzini, 1995] wird $\alpha_i(t)$ auch als *Vorwärtswahrscheinlichkeit* bezeichnet, da die Werte für die einzelnen Zeitpunkte schrittweise vorwärts in der Zeit berechnet werden. Zur Zuordnung eines bestimmten Modellzustandes zu einem bestimmten Merkmalsvektor wird außerdem noch die Verbundwahrscheinlichkeitsdichte $\beta_i(t) = p\left(X(t+1, \ldots, T), q(t) = q_i|\lambda\right)$ benötigt, die auch als *Rückwärtswahrscheinlichkeit* bekannt ist, da die Werte rekursiv vom Ende der Beobachtung beginnend, rückwärts berechnet werden:

$$\beta_i(t) = \sum_{l=2}^{I-1} a_{il} b_l(t+1) \beta_l(t+1) \ \text{ für } 1 \leq t < T; 1 < i < I \tag{4.4.5}$$

$$\beta_i(T) = a_{iI} \tag{4.4.6}$$

$$\beta_1(1) = \sum_{l=2}^{I-1} a_{1l} b_l(1) \beta_l(1) \tag{4.4.7}$$

Streng genommen sind die vorgestellten Größen $\alpha_i(t)$ und $\beta_i(t)$ nur bei Verwendung diskreter HMM-Ausgabesymbole echte Wahrscheinlichkeiten [Deller u. a., 1993]. Um aus

[7] Für die Formulierung des Baum-Welch-Algorithmus ist vereinfachend angenommen, daß die zu trainierende Zustandsfolge nur aus einem HMM besteht. Bei einer Modellfolge, die aus mehreren HMM zusammengesetzt ist, muß die Gesamtzahl aller Zustände der Folge verwendet werden.

den vorgestellten Dichten echte Wahrscheinlichkeiten zu machen, müßte das Integral der Dichte über ein Intervall um den betrachteten Merkmalsvektor herum ausgewertet werden. Da jedoch die Verarbeitung im Rechner nur diskrete Vektoren zuläßt und somit direkt mit den Dichten gerechnet werden kann, sind, wie oben erwähnt, die Bezeichnungen Vorwärts- und Rückwärtswahrscheinlichkeit auch bei "kontinuierlichen" Merkmalsvektoren üblich. Nun läßt sich die Auftrittswahrscheinlichkeit für einen Modellzustand bei Beobachtung der gesamten Merkmalsfolge X angeben:
Die Größen

$$\gamma_i(t) = Pr\left(q(t) = q_i | X, \lambda\right) = \frac{p(X, q(t) = q_i|\lambda)}{\sum_{n=1}^{I} p(X, q(t) = q_n|\lambda)} \tag{4.4.8}$$

und

$$\xi_{il}(t) = Pr\left(q(t) = q_i, q(t+1) = q_l | X, \lambda\right) = \frac{p(X, q(t) = q_i, q(t+1) = q_l|\lambda)}{\sum_{n=1}^{I} p(X, q(t) = q_n|\lambda)} \tag{4.4.9}$$

definieren die Wahrscheinlichkeiten für das Ereignis $q(t) = q_i$ bzw. das Verbundereignis $q(t) = q_i$ und $q(t+1) = q_l$ (mit dem Parametersatz λ) unter der Bedingung, die komplette Merkmalsvektorfolge X beobachtet zu haben [Bilmes, 1998]. Mit den Abkürzungen $\alpha_i(t)$ und $\beta_i(t)$ gilt dann:

$$\xi_{il}(t) = \frac{\alpha_i(t) a_{il} b_l(t+1) \beta_l(t+1)}{\sum_{n=1}^{I} \alpha_n(t)\beta_n(t)} \text{ und } \gamma_i(t) = \sum_{l=1}^{I} \xi_{il}(t) = \frac{\alpha_i(t)\beta_i(t)}{\sum_{n=1}^{I} \alpha_n(t)\beta_n(t)} \tag{4.4.10}$$

Zur Betrachtung des allgemeinen Falls muß noch berücksichtigt werden, daß die Ausgabewahrscheinlichkeitsdichte jedes Zustandes $b_i(t)$ verschiedene, nicht direkt beobachtbare Mixturen (vergl. Abschnitt 4.3.3) annehmen kann [Bilmes, 1998]. Die Abfolge dieser Mixturen sei $K = \left(k_{q(1)}(1), \ldots, k_{q(T)}(T)\right)$, wobei $k_{q(t)}(t) = j$ den Index der Mixtur $b_{q(t)j}(t)$ im Zustand $q(t)$ zum Zeitpunkt t der Ausgabedichte $b_{q(t)}(t)$ bezeichnet (vergl. auch Gl. (4.3.1)). Analog zu Gl. (4.4.9) bzw. (4.4.10) ist dann die Verbundwahrscheinlichkeit $\zeta_{i,j}(t)$ für das Auftreten des HMM-Zustandes q_i und der Mixtur j

$$\zeta_{i,j}(t) = Pr\left(q(t) = q_i, k_{q_i}(t) = j | X, \lambda\right) = \frac{\sum_{l=2}^{I-1} \alpha_l(t-1) a_{li} c_{ij} b_{ij}(t) \beta_i(t)}{p(X|\lambda)} \tag{4.4.11}$$

4.4.2. Lösung des EM-Algorithmus für HMM

Im zuerst durchzuführenden E-Schritt des EM-Algorithmus wird der bedingte Erwartungswert der Verbundwahrscheinlichkeitsdichte $p(X, K, Q)$ aus Merkmalsfolge, Mixturfolge und Zustandsfolge bezüglich der neuen Parameterschätzung $\hat{\lambda}$ gebildet, wobei als Bedingung die Merkmalsfolge X, sowie "alte" Parameter λ vorliegen (eine Herleitung der Gl. (4.4.12) ist im Anhang D zu finden). Es ergibt sich:

$$\mathfrak{Q}(\hat{\lambda}, \lambda) = \frac{1}{p(X|\lambda)} \sum_{q \in Q} \sum_{k \in K} \log\left(p(X, q, k|\hat{\lambda})\right) p(X, q, k|\lambda) \tag{4.4.12}$$

Aus der HMM-Struktur ergibt sich für die Verbundwahrscheinlichkeitsdichte

$$p(X,q,k|\hat{\lambda}) = a_{q(0)q(1)} \cdot \prod_{t=1}^{T} a_{q(t-1)q(t)} \cdot \prod_{t=1}^{T} c_{q(t)k} b_{q(t)k}(t) \tag{4.4.13}$$

Durch Einsetzen von Gl. (4.4.13) in (4.4.12) ergeben sich drei Teilgleichungen. Im M-Schritt wird durch Nullsetzen des Gradienten bezüglich der gesuchten Parameter in jeder Gleichung ein neuer Parametersatz ermittelt. Es ergeben sich mit $\lambda = \{\mathbf{A}, b_1, \ldots, b_I\}$ bei insgesamt R verschiedenen Trainingsfolgen (Worte bzw. Sätze) die verbesserten Parameter zu [Young u. a., 2000]:

$$a_{il} = \frac{\sum_{r=1}^{R} \frac{1}{p(X_r|\lambda)} \sum_{t=1}^{T_r} \xi_{il}^r(t)}{\sum_{r=1}^{R} \frac{1}{p(X_r|\lambda)} \sum_{t=1}^{T_r} \gamma_i^r(t)} \text{ und } c_{ij} = \frac{\sum_{r=1}^{R} \frac{1}{p(X_r|\lambda)} \sum_{t=1}^{T_r} \zeta_{ij}^r(t)}{\sum_{r=1}^{R} \frac{1}{p(X_r|\lambda)} \sum_{t=1}^{T_r} \gamma_i^r(t)} \tag{4.4.14}$$

Die obigen Gleichungen sind für das Training von diskreten Ausgabedichten und für hybride akustische Modelle nach Kapitel 5 ausreichend, da der Vektorquantisierer bzw. der Klassifikator separat trainiert wird. Im Falle von Elementarwahrscheinlichkeitsdichten nach Abschnitt 4.3.2 oder 4.3.3 ergeben sich weiter:

$$\vec{\mu}_{ij} = \frac{\sum_{r=1}^{R} \frac{1}{p(X_r|\lambda)} \sum_{t=1}^{T_r} \zeta_{ij}^r(t) \vec{x}(t)}{\sum_{r=1}^{R} \frac{1}{p(X_r|\lambda)} \sum_{t=1}^{T_r} \zeta_{ij}^r(t)} \tag{4.4.15}$$

$$\mathbf{\Sigma_{ij}} = \frac{\sum_{r=1}^{R} \frac{1}{p(X_r|\lambda)} \sum_{t=1}^{T_r} \zeta_{ij}^r(t) \left(\vec{x}(t)\vec{x}^T(t) - \vec{\mu}_{ij}\vec{\mu}_{ij}^T \right)}{\sum_{r=1}^{R} \frac{1}{p(X_r|\lambda)} \sum_{t=1}^{T_r} \zeta_{ij}^r(t)} \tag{4.4.16}$$

Wie schon im Abschnitt 4.1 erwähnt, besteht ein HMM für eine Merkmalsvektorfolge allgemein aus verschiedenen aneinandergereihten HMM, die Untereinheiten (meist Worte oder Phoneme) modellieren. Die dargelegten Beziehungen sind auch für solche aneinandergereihten HMM prinzipiell gültig. Die erweiterten Gleichungen, die Übergänge zwischen aneinandergereihten HMM berücksichtigen, sind [Young u. a., 2000] zu entnehmen.
Für das Training der HMM-Parameter läßt sich ebenfalls der zur Dekodierung benötigte Viterbi-Algorithmus (vergl. Abschnitt 4.7.1) verwenden, er unterscheidet sich vom Baum-Welch-Algorithmus dadurch, daß anstelle aller Zustandsfolgen durch das HMM nur der wahrscheinlichste Pfad verfolgt wird [Schukat-Talamazzini, 1995]. Die Rekursionsvorschrift durch die Merkmalsvektorfolge lautet dann:

$$\theta_1(1) = 1 \ , \ \theta_i(1) = b_i(t) \text{ für } 1 < i < I \tag{4.4.17}$$

$$\theta_i(t) = \max_l \left(\theta_l(t-1) a_{li}\right) b_i(t) \text{ für } t > 1 \text{ und } 1 < i < I \tag{4.4.18}$$

Der Näherungswert für die Produktionswahrscheinlichkeit ist dann $p(X|\lambda) = \max_i \theta_i(T)$. Wird zusätzlich zu jedem Zeitschritt der Index des bis dahin maximalen Pfades notiert, so läßt sich am Ende der Merkmalsfolge eine optimale Zuordnung zwischen Zuständen

und Merkmalen durch Rückverfolgen dieser Indizes finden (vergl. Abschnitt 4.7.1). Auf Kosten der Genauigkeit der trainierten Parameter läßt sich ein Geschwindigkeitsvorteil erreichen, insbesondere bei sehr großen Datenmengen geht der Verlust an Genauigkeit gegen Null. Da das Parametertraining mit dem Viterbi-Algorithmus jedoch im beschriebenen Baum-Welch-Verfahren, das alle möglichen Pfade durch das Modell betrachtet, als Spezialfall enthalten ist, wird hier auf weitere Betrachtungen dieses Trainingsverfahrens verzichtet.

4.4.3. Ablauf des Modelltrainings

Wie schon das Training der Klassifikatoren mittels Gradientenabstieg aus Kapitel 3, so ist auch der Baum-Welch-Algorithmus ein iteratives Verfahren. Daher ist auch hier eine Initialisierung notwendig, die sich wie folgt zusammenfassen läßt:

- In der Übergangsmatrix $\mathbf{A}$ werden alle erlaubten Übergänge mit positiven Werten initialisiert, wobei die Stochastizitätsbedingungen eingehalten werden müssen. Alle anderen Übergangswahrscheinlichkeiten werden auf Null gesetzt.

- Alle Mittelwert- und Varianzvektoren[8] $\vec{\mu}_{ij}$ und $\vec{\sigma}^2_{ij}$ der Gaußmixturen werden auf den Mittelwert- und den Varianzvektor aller Merkmalsvektoren des Trainingsmaterials gesetzt.

- Die Gewichtungsfaktoren c_{ij} der Mixturen werden mit $\frac{1}{J}$ initialisiert.

Das Training von Gauß-HMM nach dem Baum-Welch-Algorithmus läuft dann folgendermaßen ab:

1. Initialisierung der Modelle (siehe oben)

2. Durchlaufen aller Trainingsbeispiele und berechnen der Hilfsgrößen aus Abschnitt 4.4.1

3. Neuschätzung aller Parameter aller HMM mit den Gleichungen aus Abschnitt 4.4.2

4. Zurück zu Schritt 2, bis eine feste Anzahl an Iterationen (meist 4 oder 5) durchlaufen worden ist

5. Falls erwünscht, Erhöhen der Anzahl der Mixturen um 1 in ausgewählten HMM-Zuständen durch Kopieren der vorhandenen Gaußdichte mit dem größten Gewichtungskoeffizienten, Halbieren der Koeffizienten und Verschieben der Mittelwerte um 0,2 Standardabweichungen

6. Zurück zu 2, falls die Anzahl der Mixturen erhöht worden ist, sonst Abbruch des Verfahrens

[8] In dieser Arbeit werden nur diagonale Kovarianzmatrizen eingesetzt

4.5. Kontextabhängige Modelle

Eine wichtige Erweiterung der in 4.3.3 beschriebenen HMM führt zu einer kontextabhängigen Modellierung: Statt fester Phonemmodelle wird ein separates Modell für jedes Phonem innerhalb eines bestimmten Kontextes vorgesehen. Der Fall der Betrachtung des Vorgänger- und Nachfolgerphonems ist in Bild 4.5.1 dargestellt, diese kontextabhängigen Modelle sind in der Literatur als *Triphone* bekannt. Werden separate

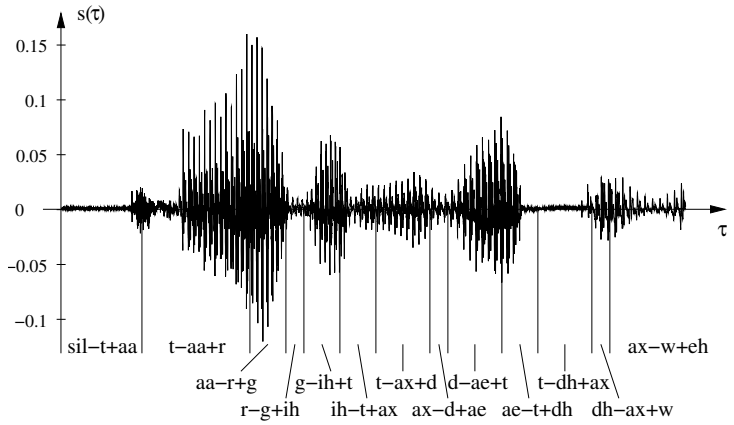


Abbildung 4.5.1.: Kontextabhängige Segmentierung der Worte *targeted at the*

Modelle nur für das Phonem mit Vorgänger- *oder* Nachfolgerphonem erstellt, so erhält man *Biphone*. Diese Modelle haben allerdings nur innerhalb einer Triphon-Modellierung Bedeutung (s. unten).
Das erste Problem, was sich bei einer Triphon-Modellierung ergibt, ist die Anzahl der Modelle: Bei einem Phonemsatz von $M = 50$ Phonemen ergibt sich theoretisch eine Menge von $M_{\mathrm{Tri}} = 50^3 = 125\,000$ Triphon-Modellen. Praktisch sind darunter jedoch viele Kontext-Kombinationen, die in der natürlichen Sprache nicht auftreten. Trotzdem verbleiben je nach Aufgabe und Datenbasis etwa 10 000 Modelle, die aus dem begrenzten Datenvorrat geschätzt werden sollen, was zu erhöhter statistischer Unsicherheit führt. Zusätzlich ist davon auszugehen, daß nicht alle erlaubten Triphone in der Trainingsdatenmenge auftreten, so daß in den Testdaten Triphone auftreten, für die kein Triphon-Modell existiert. Um trotzdem statistisch gut geschätzte Modelle zu erhalten, gibt es mehrere Strategien [Schukat-Talamazzini, 1995]:

- Tritt ein unbekanntes Triphon auf, so wird anstelle dieses Triphons ein Biphon oder ein Monophon verwendet, für das ein gut trainiertes Modell existiert

- Parameter ähnlicher Triphone werden zusammengefaßt, dadurch kann die Trainingsdatenmenge besser auf die Modelle aufgeteilt werden. Neben dem Zusammenfassen der Übergangsmatrizen entsprechend dem ursprünglichen Monophon werden durch ein geeignetes Verfahren auch die Parameter der Ausgabedichten oder sogar ganze Modelle zusammengefaßt. Das hier verwendete Verfahren – Zu-

sammenfassen anhand eines Entscheidungsbaums (TBC[9]) – welches auch unbekannte Triphone einem ähnlichen, existierenden Modell zuordnen kann, wird im Anhang E beschrieben.

Das Training der freien Parameter kann unverändert mit den Gleichungen aus Abschnitt 4.4 ablaufen.

4.6. Sprachmodelle

Das Sprachmodell liefert eine Abschätzung der in Gl. (4.1.2) benötigten Größe $Pr(M)$. Steht M für einen ganzen Satz, so beschreibt $M = (m_1, \ldots, m_U)$ die Folge von Wörtern und es gilt $Pr(M) = Pr(m_1, \ldots, m_U)$. Um diese Wahrscheinlichkeit zu schätzen, empfiehlt sich zunächst eine Faktorisierung [Willett, 2000]:

$$Pr(m_1, \ldots, m_U) = Pr(m_1) \cdot Pr(m_2|m_1) \cdot Pr(m_3|m_2, m_1) \cdot \ \ldots \ \cdot Pr(m_U|m_{U-1}, \ldots, m_1) \tag{4.6.1}$$

Je nach gewünschtem Aufwand kann Gl. (4.6.1) nach N Wörtern abgebrochen werden. Das daraus entstehende Modell ist dann ein N-Gramm-Sprachmodell, die nur Kontexte bis $N-1$ berücksichtigt. In der Praxis üblich sind Unigramme ($N = 1$), Bigramme ($N = 2$) oder Trigramme ($N = 3$)[10], Gl. (4.6.2) gibt eine Bigramm-Näherung für Gl. (4.6.1) an:

$$Pr(m_1, \ldots, m_U) \approx Pr(m_1) \cdot Pr(m_2|m_1) \cdot Pr(m_3|m_2) \cdot \ \ldots \ \cdot Pr(m_U|m_{U-1}) \tag{4.6.2}$$

Zum Schätzen der Parameter eines N-Gramm-Sprachmodells sind bereits bei kleinem Vokabular Millionen von Wörtern notwendig. Trotzdem kommen prinzipiell Wortkombinationen vor, deren N-Gramm Wahrscheinlichkeit nicht vorliegt bzw. aufgrund einer zu geringen Menge von Trainingsdaten nicht sicher geschätzt werden kann. Daher bedient man sich in der Praxis eines sogenannten Backoff-N-Gramm Modells, bei dem nicht vorhandene Kontexte durch Wahrscheinlichkeiten mit weniger Kontext ersetzt werden. Ein Backoff-Trigramm ist also durch

$$\begin{aligned} Pr(m_u|m_{u-1}, m_{u-2}) &= \begin{cases} Pr^{\text{tri}}(m_u|m_{u-1}, m_{u-2}) \text{ falls das Trigramm vorliegt} \\ Pr^{\text{bi}}(m_u|m_{u-1})B(m_u, m_{u-1}) \end{cases} \\ Pr(m_u|m_{u-1}) &= \begin{cases} Pr^{\text{bi}}(m_u|m_{u-1}) \text{ falls das Bigramm vorliegt} \\ Pr^{\text{uni}}(m_u)B(m_u) \end{cases} \end{aligned} \tag{4.6.3}$$

definiert. Die Backoff-Gewichte $B(m_u, m_{u-1})$ und $B(m_u)$ dienen zur Normierung der Schätzwerte, da durch den Austausch der Wahrscheinlichkeiten unter Umständen die Stochastizitätsbedingung verletzt wird. Falls die zu erwartenden Äußerungen im Vorfeld

[9] TBC - *engl.*: Tree-Based Clustering

[10] Der Sonderfall $N = 0$ wird als Zerogramm bezeichnet und ist mit der Vokabulargröße B durch $Pr(m_u) = \frac{1}{B}$ gegeben.

eingegrenzt werden können, läßt sich das Sprachmodell als kontextfreie Grammatik formulieren. Die einzelnen Wörter dürfen nur an der in der Grammatik angegebenen Stelle auftauchen, nicht in der Grammatik notierte Wörter haben die Auftrittswahrscheinlichkeit Null. In dieser Arbeit ist eine solche Grammatik für Experimente mit der AURORA2 Datenbasis definiert (vergl. Abbildung 7.3.5), wo nur Ketten von Zahlwörtern vorkommen können. Denkbar wäre eine solche Grammatik auch für Dialogsysteme, bei denen die Äußerungen automatisch weiterverarbeitet werden müssen und die zu erwartenden Äußerungen klar strukturiert sind.

4.7. Dekodierung

Die Dekodierung (Erkennung) eines unbekannten Satzes ist letztlich das Ziel eines trainierten Spracherkenners. In dieser Arbeit werden hierzu zwei prinzipiell verschiedene Erkennungsstrategien mit unterschiedlichen Vorteilen verwendet. Der Viterbi-Dekoder erlaubt eine einfache zeitsynchrone Dekodierung und basiert auf dem schon erwähnten Viterbi-Algorithmus (s. Abschnitt 4.4). Die verwendete Implementierung des Viterbi-Dekoders ist jedoch aus Gründen der Effizienz auf ein Bigramm-Sprachmodell beschränkt. Aus diesem Grund wird für Trigramm-Sprachmodelle der im Abschnitt 4.7.2 vorgestellte Stack-Dekoder eingesetzt. Das Prinzip der Stack-Dekodierung beruht auf der Erweiterung von Satzhypothesen und ist ausführlich in [Willett, 2000] beschrieben. Ein für beide Verfahren problematischer Sachverhalt ist die Tatsache, daß das akustische Modell im allgemeinen eine Wahrscheinlichkeitsdichte liefert, während das Sprachmodell eine echte Wahrscheinlichkeit berechnet. Um die Wertebereiche anzugleichen, wird in der Praxis ein Sprachmodellfaktor ϕ eingeführt, der Gl. (4.1.2) zu

$$n^* = \underset{n}{\operatorname{argmax}} \{\Pr(M_n|X)\} = \underset{n}{\operatorname{argmax}} \left\{ p(X|M_n) \Pr(M_n)^{\phi} \right\} \tag{4.7.1}$$

ändert. Der Wert dieses Sprachmodellfaktors ist vom verwendeten Sprachmodell und von der Aufgabe abhängig und kann empirisch bestimmt werden.
Das Ergebnis einer Erkennung ist im allgemeinen die beste Hypothese des verwendeten Dekoders. Eine absolute Bewertung der Hypothese findet dabei nicht statt, zur Berechnung der Konfidenz einer Hypothese sind weitere Maßnahmen notwendig. Zusätzlich zur besten Hypothese und deren Segmentierung können mit den beschriebenen Verfahren auch die nächsten $N-1$ besten Hypothesen (N-Best-Liste) oder ein vom Dekoder bewertetes Netzwerk[11] der durchlaufenen Worthypothesen geliefert werden .

4.7.1. Dekodierung mit dem Viterbi-Algorithmus

Das Dekodierungsproblem ist bereits in Gl. (4.1.2) beschrieben, es gilt nun, das akustische Modell mit dem Sprachmodell zu verbinden. Bei der zeitsynchronen Viterbi-Suche erzeugt man zunächst ein Netzwerk, das alle erlaubten Worte enthält. Das einzelne Wort ist hierbei durch das akustische Modell als Abfolge von Phonem-HMM oder einem Ganzwort-HMM repräsentiert. Im einfachsten Fall werden die Wortübergänge durch

[11] in der englischsprachigen Literatur auch als *lattice* bezeichnet

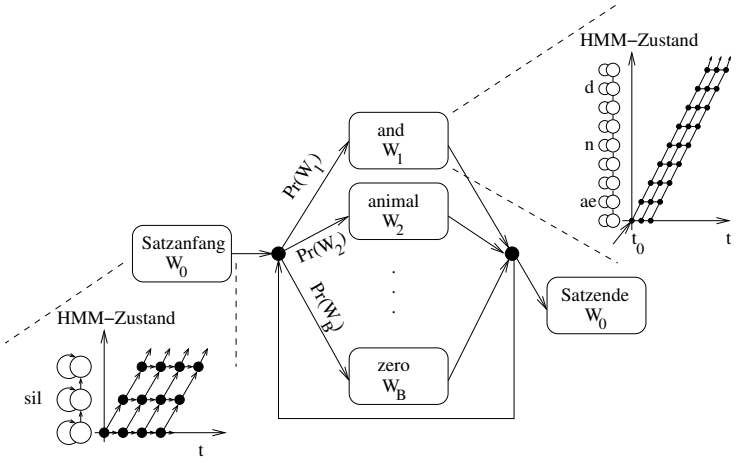


Abbildung 4.7.1.: Erkennungsnetzwerk mit Unigramm-Sprachmodell

das Sprachmodell gewichtet, was *a priori* bestimmte Wortfolgen gegenüber anderen bevorzugt. Weitergehende Strategien zur Verknüpfung von Sprachmodell und akustischem Modell finden sich in [Willett, 2000]. Das Verfahren aus [Woodland u. a., 1995], nach dem in dieser Arbeit dekodiert worden ist, hat in der englischsprachigen Literatur den Namen *Token Passing*. Im ersten Zeitschritt wird mit einer Hypothese (einem *Token*) für ein Satzanfangsmodell (z.B. das Pausenmodell *sil*) gestartet. Bei jeder Verzweigung im Erkennungsnetzwerk wird die aktuelle Hypothese kopiert und parallel in den erlaubten und möglichen Abzweigungen weiterverfolgt. Bei jedem Zeitschritt vollführen alle Hypothesen einen Zustandübergang im gerade betrachteten HMM und aktualisieren die Produktionswahrscheinlichkeit bis zu diesem Zeitschritt (vergl. Gl. (4.4.18)). Die Eingrenzung der Anzahl möglicher Hypothesen geschieht dabei durch Definition eines Suchstrahls: Der Wert der Produktionswahrscheinlichkeit der aktuell besten Hypothese wird mit dem aller anderen Hypothesen verglichen. Diejenigen, deren Werte zu gering im Verhältnis zum Maximalwert sind, werden gelöscht. In der Literatur ist dieses Vorgehen als *pruning*[12] von Hypothesen bekannt. Bei jedem Eintritt in ein neues Wort wird die Sprachmodellwahrscheinlichkeit für dieses Wort gewichtet mit dem Sprachmodellfaktor in die Produktionswahrscheinlichkeit der aktuellen Hypothese eingerechnet. Das Durchlaufen der einzelnen Hypothesen durch die verschiedenen Wortsequenzen wird mitprotokolliert. Am Ende der Beobachtung kann dann durch Rückverfolgen der Wortsequenz der Hypothese mit maximalem Wahrscheinlichkeitswert die beste Wortsequenz gefunden werden. Daher liefert eine Viterbi-Dekodierung neben der besten Wortsequenz auch die optimale Zuordnung (Segmentierung) zwischen den Modellzuständen und den einzelnen Merkmalsvektoren. Ist die Wortfolge gegeben und nur das Auffinden dieser Zuordnung gesucht, so findet eine *Viterbi-Segmentierung* statt, zum Beispiel zum Bestimmen von Zielwerten für das überwachte Training eines Klassifikators.

[12] *engl.* to prune: abschneiden, kürzen

4.7.2. Stack-Dekodierung

Die Grundidee der Stack-Dekodierung ist der Suche des kürzesten Weges in einem mit Weglängen markierten Graphen entlehnt. Bei diesem, in der Literatur als A*-Suche bekannten Verfahren, ist die Betrachtung der bisher zurückgelegten Weglänge im Graphen und eine garantierte Unterschätzung (A*-Kriterium [Nillson, 1971]) der Restweglänge entscheidend. In der Übertragung auf die Dekodierung von Sprachsignalen ist das zentrale Element eine Hypothese H mit sämtlichen bis zu einem Zeitpunkt t_H dekodierten Worten. Die Bewertung der Hypothese (analog zur Länge der Wegstrecke) geschieht über die Summe der akustischen Likelihood $\mathfrak{L}$ (Gl. (4.4.1)) und der Wahrscheinlichkeit des Sprachmodells (Gl. (4.6.3)) der bis dahin durchlaufenden Modelle. Durch die unterschiedliche Länge der Hypothesen und der Verarbeitung auf Wortebene ist die Stack-Dekodierung in dieser Form nicht zeitsynchron. In [Willett, 2000] finden sich Änderungen gegenüber diesem Grundprinzip zur besseren Anpassung und Umsetzung auf die Spracherkennung, sowie ein detaillierter Ablauf der Dekodierung: So werden anstelle eines Stacks mehrere zeitlich aufeinanderfolgende Stacks angelegt, auf eine Unterschätzung des Restweges wird aufgrund ihrer aufwendigen Realisierung komplett verzichtet. Damit wird der beste Stack zum Zeitpunkt t_H nicht mehr ausgewählt, sondern ist immer der Stack, der diesem Zeitpunkt zugeordnet ist. Beendet ist das Verfahren, wenn das Äußerungsende erreicht ist, in diesem Fall steht im letzten Stack die beste Gesamthypothese der Äußerung. Um einer Hypothese neue Wörter hinzuzufügen ist eine Einzelwortdekodierung notwendig. Dazu wird das Verfahren nach Abschnitt 4.7.1 eingesetzt, durch das Vorwissen der bisherigen Hypothese kann der Suchraum hier aber deutlich eingeschränkt werden. Eine Beschreibung des in dieser Arbeit verwendeten Stack-Dekoders ist in [Willett u. a., 1998] zu finden. Der große Vorteil des Stack-Dekoders ist die Möglichkeit beliebige N-Gramm-Sprachmodelle zu verarbeiten, durch Betrachtung der auf den einzelnen, linear angeordneten Stacks liegenden Hypothesen mit den bisherigen möglichen Wortketten kann leicht die zum aktuellen Zeitpunkt notwendige Sprachmodellwahrscheinlichkeit gefunden werden.

4.8. Ergebnisse

Die folgende Evaluation gaußbasierter HMM-Systeme wird auf dem sprecherunabhängigen *si-05*-Test der WSJ0-Datenbank (s. Anhang B.2) durchgeführt. Alle Parameter der Gaußdichten und der HMM werden mit dem sprecherunabhängigen Trainingsset *si-84* geschätzt. Zum Einsatz kommen sowohl kontextunabhängige Modelle (*Monophone*) mit 47 Einzelmodellen (45 Phoneme und 2 Pausenmodelle, s. Anhang B.2) als auch kontextabhängige Modelle (*Triphone*). Die kontextabhängigen Modelle werden vor dem Training noch anhand eines Entscheidungsbaums (TBC) (vergl. Anhang E) zusammengefaßt.
Das Gütemaß für die Beurteilung der Qualität eines akustischen Modells ist im allge-

meinen die Wortfehlerrate (WFR) des kompletten Spracherkenners. Die WFR berechnet sich zu

$$\text{WFR} = \frac{I + S + D}{N} \quad , \tag{4.8.1}$$

wobei I die Anzahl der Einfügungen[13], S die Anzahl der Ersetzungen[14] und D die Anzahl der Auslassungen[15] bezeichnet. Zur Berechnung dieser drei Fehlerarten wird die Levenstein-Distanz zwischen dem erkannten Satz und dem Originalsatz mittels dynamischer Programmierung minimiert. Alternative Angaben zur WFR sind die *accuracy* (Acc) (Acc $= \frac{N-(I+S+D)}{N} = 1 - \text{WFR}$) und die *correctness* (Cor) (Cor $= \frac{N-(S+D)}{N}$). Letzteres Maß läßt Einfügungen unberücksichtigt und eignet sich z.B. zur Beurteilung von Schlüsselworterkennern, die nur nach ganz bestimmten Wörtern im akustischen Datenstrom suchen und den Rest verwerfen.

System	**Anzahl HMM**	**Mixturen**	**#Parameter HMM-Dichten**	**WFR**
Monophon kont.Gauß	47	6	65052	29,01%
Monophon kont.Gauß	47	10	108420	16,98%
Monophon kont.Gauß	47	12	130104	14,87%
Triphon, kont.Gauß, TBC	8510	6	$2.71 \cdot 10^6$	12,52%
Triphon, kont.Gauß, TBC	8510	12	$5.4 \cdot 10^6$	13,69%

Tabelle 4.1.: WFR verschiedener Gauß-Systeme mit dem *si-05*-Testset, Bigramm Sprachmodell, Viterbi-Dekoder

Tabelle 4.1 zeigt Ergebnisse verschiedener Gauß-HMM-Systeme mit dem *si-05* Testset. Der Erkenner benutzt den Viterbi-Algorithmus (Abschnitt 4.7.1) und verwendet ein Backoff-Bigramm-Sprachmodell (s. Abschnitt 4.6). Das *Monophon*-Modellset verwendet 45 lineare HMM mit je 3 aktiven[16] Zuständen für die 45 Phoneme der WSJ, sowie ein Pausenmodell für lange Pausen am Satzanfang und -ende (*sil*) mit 3 aktiven Zuständen und ein Pausenmodell (*sp*) für kurze Wortübergangspausen mit 1 aktivem Zustand. Die Modelle werden mit globalem Mittelwert und globaler Varianz initialisiert (s. Abschnitt 4.4.3). Das Training erfolgt dann dem Schema aus Abschnitt 4.4.3. Um numerischen Problemen während des Trainings zu begegnen, werden Mixturen mit zu kleinem Gewicht gelöscht. Außerdem wird der Varianzvektor nach unten begrenzt, damit die geschätzte Dichte nicht unnatürlich große Werte (ähnlich einem Dirac-Stoß) durch zu kleine Varianzwerte annimmt. Dieser unerwünschte Fall kann vor allem dann auftreten, wenn die Varianz durch zu wenige Trainingsbeispiele geschätzt werden muß. Weitere Strategien, wie das automatische Erhöhen der Mixturen je nach Menge der Trainingsdaten, wie in [Willett, 2000] beschrieben, werden für diese Experimente nicht verwendet.

[13] *engl.*: insertions
[14] *engl.*: substitutions
[15] *engl.*: deletions
[16] immer zuzüglich einem Anfangs- und einem Endzustand für Modellübergänge

Der im Abschnitt 4.7 beschriebene Sprachmodellfaktor ϕ ist für alle Experimente mit dem *si-05*-Test der Kapitel 4 und 5 auf $\phi = 5.0$ gesetzt. Das Verwerfen (pruning) von Hypothesen ist so eingestellt, das keine Erkennungsfehler entstehen. Bei den kontextabhängigen Modellen ist die Anzahl der Modelle nach dem Zusammenfassen von Parametern angegeben.

System	**Mixturen**	**#Parameter HMM-Dichten**	**WFR**
Triphone, kont.Gauß, TBC	6	$2.71 \cdot 10^6$	11,04%
Triphone, kont.Gauß, TBC	12	$5.4 \cdot 10^6$	11,84%

Tabelle 4.2.: WFR verschiedener Gauß-Systeme mit dem *si-05*-Testset, Trigramm-Sprachmodell, Stack-Dekoder

Die Ergebnisse aus Tabelle 4.2 benutzen die gleichen akustischen Modelle wie in Tabelle 4.1, diesmal jedoch mit einem Backoff-Trigramm-Sprachmodell. Ermittelt worden sind diese Zahlen mit dem Stack-Dekoder aus Abschnitt 4.7.2. Zur Auswertung der Signifikanz dient die in Abschnitt 5.5 vorgestellte Wahrscheinlichkeit für eine systematische Verbesserung PrV. Erwartungsgemäß sind die Ergebnisse mit einem Trigramm ungefähr 2,5% absolut besser (PrV = 100%), als die WFR mit einem Bigramm. Während sich bei den Monophon-Systemen bei steigender Anzahl der Mixturen eine signifikante Verbesserung (PrV = 100%) ergibt, ist bei den Triphonen das Minimum der Fehlerrate bereits bei 6 Mixturen pro Zustand erreicht. Grund hierfür ist die große Anzahl an Parametern, die bei Triphon-Modellen geschätzt werden muß und für die bei 12 Mixturen die Menge an Trainingsdaten nicht mehr ausreicht.
Da die WSJ0-Datenbank international verfügbar ist, lassen sich die Ergebnisse direkt mit Spracherkennungssystemen anderer Forschungseinrichtungen vergleichen: In [Wendt u. a., 2001] beträgt die beste WFR für ein semi-kontinuierliches System mit Bigramm-Sprachmodell auf dem *si-05*-Test $12,1\%$. Ein kontinuierliches System mit MFCC-Merkmalen, zusammengefaßten Triphonen und einem Trigramm-Sprachmodell erreicht in [Launay u. a., 2002] eine WFR von $4,6\%$.
Ergebnisse von Gaußmodellen mit AURORA2-Daten finden sich im Abschnitt 7.3.1 in Tabelle 7.1, da sie dort mit den Resultaten verteilter Gauß-Systeme und dem AURORA2-Referenzsystem verglichen werden.

5. Hybride Ansätze zur Kombination der Klassifikatoren mit Hidden-Markov-Modellen

In diesem Kapitel werden die statischen Klassifikatoren mit Hidden-Markov-Modellen zusammengeführt. Das Ziel dieser Kombination ist, die Vorteile der beschriebenen Klassifikatoren (s. Kapitel 3) für zeitabhängige Signale zu nutzen, deren Dynamik durch die Markov-Modelle (s. Kapitel 4) erfaßt wird. Der folgende Unterabschnitt stellt die Basis zur Kombination von NN bzw. SVM mit HMM vor. Verbesserungen zur Klassifikator/HMM-Kombination und Details zur Implementierung finden sich in den Abschnitten 5.2 und 5.3. Abschnitt 5.4 stellt alternative NN/HMM-Kombinationsmöglichkeiten vor und die Abschnitte 5.5 und 5.6 vergleichen die vorgestellten Methoden, implementiert im kompletten Spracherkennungssystem.

5.1. Schätzung der Ausgabedichte des Hidden-Markov-Modells

Anstelle der Gaußdichten aus Kapitel 4 werden nun die Klassifikatoren aus Kapitel 3 zur Schätzung der HMM-Dichten eingesetzt. Damit die Berechnung der Dichte unabhängig vom gewählten Klassifikator ablaufen kann, müssen für NN (Abschnitt 5.1.1) und SVM (Abschnitt 5.1.2) unterschiedliche Vorkehrungen getroffen werden.

5.1.1. Schätzung mit neuronalen Netzen

In [Bourlard u. Morgan, 1994] und [Santini u. Bimbo, 1995] ist gezeigt worden, daß MLPs und RNN in der Lage sind, Symbolauftrittswahrscheinlichkeiten zu schätzen. Das Netz ermittelt die *a posteriori*-Wahrscheinlichkeit $Pr(\rho_j|\vec{x}(t))$ für das Auftreten eines Symbols ρ der Klasse j gegeben einen Eingangsvektor $\vec{x}(t)$. Werden die Phoneme des jeweiligen Phonemsets als zu unterscheidende Klassen aufgefaßt, so schätzt das Netz die *a posteriori*-Auftrittswahrscheinlichkeit eines Phonems gegeben einen Merkmalvektor. Bild 5.1.1 zeigt den Verlauf des Phonems *dh* und der Pausenklasse *sil* für einen gesprochenen Satz. Zusätzlich werden, analog zu Kapitel 4.1, HMM zur Modellierung des zeitlichen Verlaufs der Phonemfolge benötigt. Eine Verbindung zwischen dem NN und den HMM schafft hier die Anwendung des Satzes von Bayes:

$$Pr(\rho_j|\vec{x}(t)) = \frac{p(\vec{x}(t)|\rho_j)Pr(\rho_j)}{p(\vec{x}(t))} \Leftrightarrow p(\vec{x}(t)|\rho_j) = \frac{Pr(\rho_j|\vec{x}(t))}{Pr(\rho_j)}p(\vec{x}(t)) \tag{5.1.1}$$

Die Größe $p(\vec{x}(t)|\rho_j)$ stellt die Wahrscheinlichkeitsdichte des Merkmalvektors $\vec{x}$ für ein bestimmtes Phonem ρ_j dar. Nimmt man nun für jedes Phonem ein HMM mit nur einem

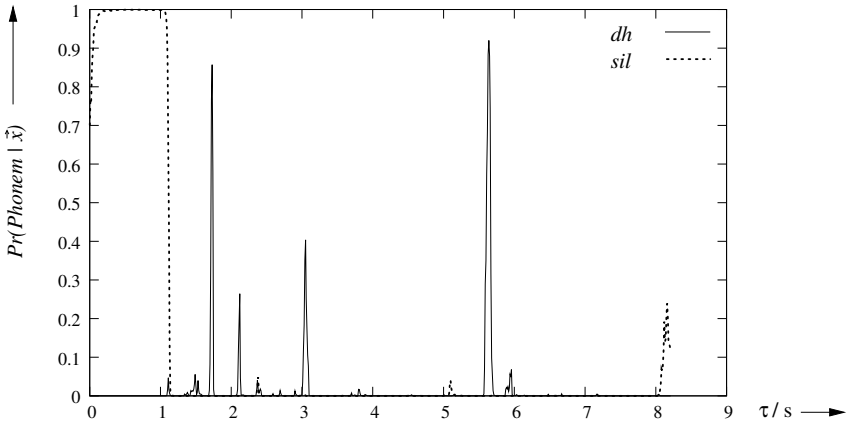


Abbildung 5.1.1.: Verlauf der *a posteriori*-Wahrscheinlichkeiten für die Klassen *dh* und *sil*, berechnet mit einem RNN

emittierenden Zustand, so ist damit die Emissionsdichte des HMM $b_j(t)$ (Gl. (4.2.3)) beschrieben. $Pr(\rho_j)$ ist die *a priori*-Wahrscheinlichkeit für das Auftreten der Klasse ρ_j (vergl. Abbildung 5.5.1 für *a priori*-Phonemauftrittswahrscheinlichkeiten). Diese Größe kann aus den Trainingsdaten für das NN gewonnen werden, indem die relativen Häufigkeiten der einzelnen Klassen bestimmt werden. Die allgemeine Wahrscheinlichkeitsdichte der Merkmalvektoren $p(\vec{x})$ ist nur mit relativ großem Aufwand zu berechnen (z.B. aus N-best-Listen der Erkennung); der absolute Wert der HMM-Emissionen ist jedoch sowohl für das Training der HMM (Abschnitt 4.4), als auch für die Erkennung (Abschnitt 4.7.1) nicht von Belang. Beim Vergleich der HMM – also der Variation von j – ist die Größe $p(\vec{x})$ hier als konstant angenommen und wird nicht weiter betrachtet (vergl. dazu die Überlegungen aus Abschnitt 6.3.1). Für die einzelnen HMM-Ausgabewahrscheinlichkeitsdichten ergibt sich also

$$b_j(t) = p(\vec{x}(t)|q_j) \propto \frac{Pr(\rho_j|\vec{x}(t))}{Pr(\rho_j)} \tag{5.1.2}$$

In Abbildung 5.1.2 ist die Struktur der Klassifikator/HMM-Verknüpfung aus Gl. (5.1.2) grafisch dargestellt. Wesentliche Einschränkungen dieses Ansatzes nach [Bourlard u. Morgan, 1994], der eine feste Zuordnung der Netzausgänge zu dem HMM-Zuständen nach Gl. (5.1.2) vorsieht, sind

- Die HMM-Zustände sind starr mit entsprechenden Ausgängen des Klassifikators verbunden
- Um für jeden HMM-Zustand eine Dichte zu berechnen, müssen Werte für alle Klassifikatorausgänge vorhanden sein
- Die Einführung von kontextabhängigen Modellen (z.B. Triphonen, s. Abschnitt 4.5) ist nur mit größeren Schwierigkeiten möglich

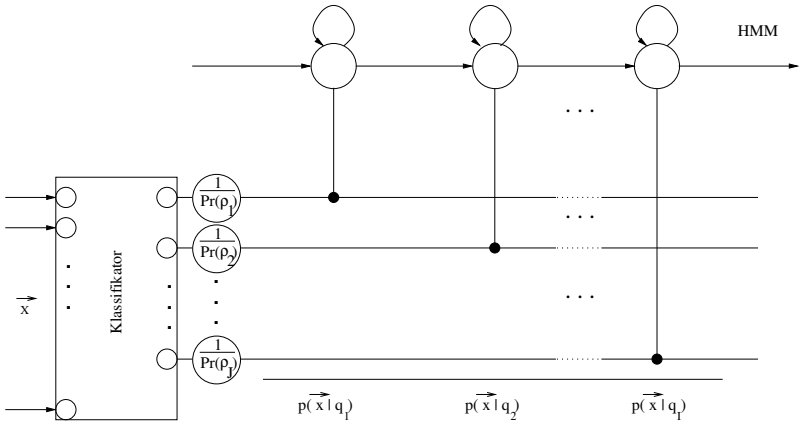


Abbildung 5.1.2.: Entstehung der Wahrscheinlichkeitsdichten in einem hybriden HMM-System mit starrer Klassifikator/HMM-Verknüpfung

Die Einschränkungen bei der Auslegung der HMM können zum Teil bereits im NN kompensiert werden: Durch Vergrößerung der Eingangsschicht mit den Merkmalvektoren benachbarter Zeitpunkte kann Kontextwissen in die Phonemklassifizierung eingebracht werden (vergl. Abschn. 3.1.3). In [Bourlard u. Morgan, 1994] und [Fritsch u. a., 1997] sind auch andere Möglichkeiten zur Modellierung kontextabhängiger HMM unter Verwendung eines deutlich komplexeren neuronalen Netzes beschrieben, der Ansatz nach Abschnitt 5.2 erlaubt demgegenüber eine kontextabhängige Modellierung ohne Veränderung des Klassifikators.

5.1.2. Schätzung mit Support-Vektor-Maschinen

In [Ganapathiraju u. a., 1998, 2003] finden sich Ansätze für hybride SVM/HMM-Spracherkenner. Im Gegensatz zu den in diesem Kapitel präsentierten Möglichkeiten ist in [Ganapathiraju u. a., 2003] neben den SVM ein vollständiges Gauß-HMM-Set notwendig, mit dem das Sprachsignal segmentiert wird. Aus den Segmenten werden Merkmalsvektoren gleicher Länge berechnet, die dann von den SVM klassifiziert werden. Um diesen mehrstufigen Prozeß während der Erkennung zu vermeiden, werden die SVM als ein Klassifikator für Symbolauftrittswahrscheinlichkeiten aufgefaßt und nach der Idee von Abschnitt 5.2 mit den HMM verbunden.
Während der Ausgang eines NN bereits eine Schätzung für die *a posteriori*-Auftrittswahrscheinlichkeit einer Klasse liefert, läßt sich der Ausgang einer *Support-Vektor*-Maschine (SVM) nicht in dieser Weise interpretieren. Das Ergebnis einer binären SVM-Klassifikation ist die Distanz des Testmusters zur Hyperebene (vergl. Abschnitt 3.2) und hat somit prinzipiell einen unbeschränkten Wertebereich. Eine Transformation in den Wertebereich $[0 \ldots 1]$ und die Interpretation als Auftrittswahrscheinlichkeit für eine Klasse läßt sich durch Aufstellen eines Histogramms mit einer geeigneten Stichprobe bestimmen. [Platt, 2000] zeigt, daß die Histogrammbestimmung durch die Transforma-

tion mit einer Sigmoidfunktion (Abb. 3.4(a)) ersetzt werden kann. Die *a posteriori*-Wahrscheinlichkeit für das Auftreten eines Symbols ρ_j ergibt sich danach zu

$$Pr(\rho_j|\vec{x}) = \frac{1}{1+\exp(Ay_j(\vec{x})+B)} \tag{5.1.3}$$

wobei y_j den Ausgang der SVM bezeichnet (s. Gl. (3.2.10)). Die Parameter A und B können durch Minimierung der negativen logarithmierten Produktionswahrscheinlichkeit der Trainingsdaten gefunden werden. Werden die Zielwerte wie in Gl. (3.1.8) auf den Bereich $y'_j \in \{0;1\}$ gesetzt, so ist die zu minimierende Funktion die Kreuzentropie

$$E = -\sum_{t=1}^{T} y'_j(t)\log Pr(\rho_j|\vec{x}(t)) + (1-y'_j(t))\log\left(1-Pr(\rho_j|\vec{x}(t))\right) \tag{5.1.4}$$

Bei nichtlinearen Kernelfunktionen ist es nach [Platt, 2000] notwendig, die Parameter der Sigmoidfunktion zumindest teilweise mit im SVM-Training nicht verwendeten Trainingsdaten zu bestimmen. Nach dem SVM-Training gibt es keinen Datenpunkt, der zwischen den von den Support-Vektoren aufgespannten Hyperebenen liegt (vergl. Bild 3.2.1). Diese Situation entspricht meist nicht den Testkonditionen, vor allem, wenn viele Trainingsbeispiele durch das Training zu Support-Vektoren bestimmt worden sind. Würde das Training der Sigmoid-Parameter mit denselben Daten durchgeführt, so entstünde eine fehlerhafte Schätzung der Wahrscheinlichkeiten. Als Gegenmaßnahme kann auch hier das Kreuzvalidierungsverfahren benutzt werden, die Sigmoid-Parameter werden dann auf den Evaluationsdaten berechnet. Eine Verbesserung der Konvergenzeigenschaften des Algorithmus zur Bestimmung der freien Parameter A und B ist in [Lin u. a., 2003] zu finden.
Wie im Abschnitt 3.2.3 beschrieben, werden die SVM nach dem Prinzip *eins gegen alle* trainiert, so daß jede SVM mit der Sigmoidfunktion die Auftrittswahrscheinlichkeit für eine Klasse ρ_j schätzt. Die Tatsache, daß die Summe der Wahrscheinlichkeiten dabei größer als 1 werden kann, wird im Dekoder ignoriert, da das Verhältnis der Hypothesen untereinander unverändert bleibt. Durch Trainieren einer Softmax-Funktion für alle SVM [Duan u. a., 2003] kann dieser Umstand korrigiert werden.

5.2. Verbundene Auftrittswahrscheinlichkeiten

Die Ursache für die Einschränkungen, die in Abschnitt 5.1.1 beschrieben sind, liegt in der festen Zuordnung der Klassifikator-Ausgänge zu den Ausgabewahrscheinlichkeitsdichten der HMM, wie in Gl. (5.1.2) dargestellt. Diese feste Zuordnung erlaubt nur einem NN-Ausgang die HMM-Emission zu bestimmen und verhindert so kontextabhängige Modelle (aufgrund der großen Anzahl an HMM) oder verteilte Spracherkennung (s. Kapitel 7). Im Vergleich mit kontinuierlichen HMM mit gaußförmigen Ausgabedichten entspricht dies einer festen Gewichtung und festen Zuordnung der Gaußfunktionen zu den jeweiligen Modellzuständen. Betrachtet man nun semi-kontinuierliche Systeme (Abschnitt 4.3.2), so eröffnet sich eine Lösung aus dieser starren Zuordnung, da hier eine Sammlung von

verschiedenen Gaußfunktionen allen HMM-Zuständen zur Verfügung steht. Wird dieses System auf die hybride Architektur übertragen, so führt dies zu einem Klassifikator, dessen Ausgänge von allen HMM-Zuständen verwendet werden können (TP[1]). Analog zum Gauß-Modell geschieht auch hier die Verknüpfung der einzelnen Klassifikator-Ausgänge (äquivalent zu den einzelnen Gaußdichten) mit den HMM-Zuständen über zustandsabhängige Gewichtungsfaktoren [Rottland u. Rigoll, 2000]. Im Einzelnen ergibt sich also - ausgehend von Gl. (5.1.2) - folgender Ausdruck für die HMM-Ausgabedichten als Summe der *verbundenen a posteriori*-Wahrscheinlichkeiten:

$$b_j(t) = p(\vec{x}(t)|q_i) \propto \sum_{j=1}^{J} c_{ij} \frac{Pr(\rho_j|\vec{x}(t))}{Pr(\rho_j)}\,, \tag{5.2.1}$$

wobei für die Summe der Gewichtungsfaktoren $\sum_{j=1}^{J} c_{ij} = 1$ gilt. Wird dieser Zusammenhang veranschaulicht, so entsteht eine Darstellung gemäß Abb. 5.2.1. Die Gewich-

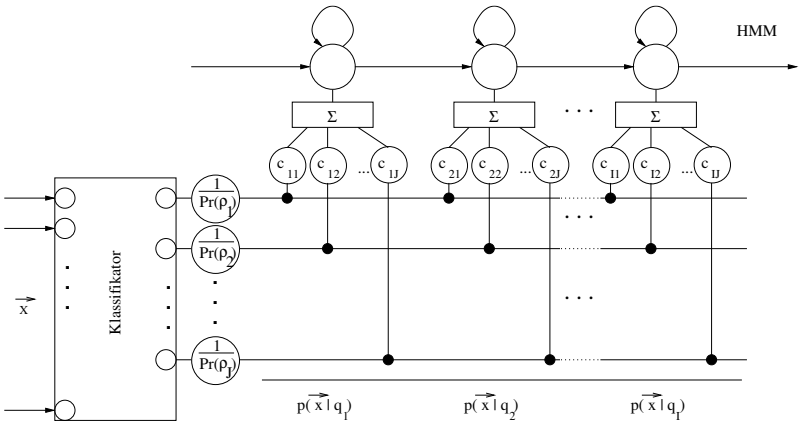


Abbildung 5.2.1.: Entstehung der Wahrscheinlichkeitsdichten in einem hybriden HMM-System mit verbundenen Auftrittswahrscheinlichkeiten

tungsfaktoren c_{ij} und Übergangswahrscheinlichkeiten der Modelle werden, wieder analog zu Abschnitt 4.3.2, mit dem Baum-Welch-Algorithmus trainiert. Die Größe $Pr(\rho_j|\vec{x}(t))$ wird vom Klassifikator geliefert, die *a priori*-Wahrscheinlichkeiten $Pr(\rho_j)$ (s. Bild 5.5.1) können aus den relativen Häufigkeiten der einzelnen Klassen in der Segmentierung der Trainingsdaten geschätzt werden. Dieser Ansatz für die Berechnung der HMM-Ausgabedichten erlaubt, die Vorteile eines diskriminativen Klassifikators mit der vollen Flexibilität der HMM zu kombinieren:

- Möglichkeit der Einbeziehung von zeitlichem Kontext in die Klassifikation durch Erweiterung des Eingangsvektors (s. Abschnitt 3.1.3)
- Diskriminatives Training des Klassifikators nach Kapitel 3

[1] verbundene Auftrittswahrscheinlichkeiten - *engl.*:Tied-Posteriors

- Benutzung beliebiger HMM-Topologien, da die Anzahl der HMM-Zustände hier unabhängig von der Anzahl der NN-Ausgänge ist
- Berechnung von Ausgabedichten für alle HMM-Zustände trotz fehlender Klassifikatorwerte (s. Kapitel 7)
- Benutzung kontextabhängiger HMM (z.B. Triphone, s. Abschnitt 4.5) ohne Veränderung des Klassifikators, die Kontextabhängigkeit der Ausgabedichten bestimmt sich allein durch die Koeffizienten c_{ij}

5.3. Training der hybriden Systeme

Eine noch nicht diskutierte Aufgabe ist das Training des gesamten Systems. Wie im Kapitel 3 erwähnt, erfordern die Klassifikatoren, die für die beschriebenen hybriden Systeme benötigt werden, ein überwachtes Training. Zu jedem Merkmalvektor muß also ein entsprechender Zielvektor für das Netztraining vorliegen. Systeme, die auf MMI-trainierten Netzen (s. Abschnitt 5.4.2) basieren, benötigen zwar für das Training keine Zielvektoren, sind aber sehr abhängig von der Initialisierung des Netzes, wofür wiederum meist auch Zielvektoren benötigt werden [Neukirchen, 1999]. Üblicherweise stehen zum Training der akustischen Modelle nur Transkriptionen ohne zeitliche Zuordnung zur Verfügung, so daß die Zielvektoren erst aus den Transkriptionen erzeugt werden müssen. Nach [Bourlard u. Morgan, 1994] können die Zielvektoren mit folgendem Verfahren iterativ erzeugt und verbessert werden:

1. Gleichmäßige Segmentierung der akustischen Beobachtungen (Merkmalvektoren) auf die Phonemfolge[2] der Transkription
2. Training des NN mit den entstandenen Zielvektoren (s. Abschnitt 3.1)
3. Training der HMM (s. Abschnitt 4.4) mit dem zuvor trainierten Netz
4. Viterbi-Segmentierung (s. Abschnitt 4.7.1) der akustischen Beobachtungen gemäß der Transkriptionen mit dem zuvor trainierten System
5. Test der akustischen Modelle (NN mit HMM) auf einem Evaluierungsdatensatz, Abbruch des Trainings, falls keine Verbesserung mehr eintritt
6. Austauschen der Zielvektoren durch die in 4 entstandenen Ziele
7. Start der nächsten Iteration (Sprung zu 2)

Da dieses Verfahren sehr zeitaufwendig ist, kann die zeitliche Segmentierung, die zum NN-Training benötigt wird, alternativ auch aus einem System mit Gauß'schen HMM gewonnen werden. Nach dem Training der HMM, die mit reinen Worttranskriptionen auskommen, kann die zeitliche Abfolge mit dem Viterbi-Algorithmus (s. Abschnitt 4.7.1)

[2] die Phonemfolgen werden z.B. aus Wort-Transkriptionen unter Benutzung des Lexikons erzeugt

hinzugefügt werden. Da dieser Prozeß nur mit den Trainingsdaten arbeitet, sind keine besonders hohen Anforderungen an die Qualität der zur Segmentierung verwendeten Modelle notwendig. Der Trainingsablauf vereinfacht sich wie folgt:

- Training von Gauß-HMM mit den Trainingsdaten
- Erzeugung von Zielvektoren durch Viterbi-Segmentierung der Trainingsdaten mit Hilfe der Gauß-HMM
- Training des NN mit den entstandenen Zielvektoren (s. Abschnitt 3.1)
- Training der hybriden HMM (s. Abschnitt 4.4) mit dem zuvor trainierten Netz
- Test der akustischen Modelle (NN mit HMM) auf einem Evaluierungsdatensatz

Soll die Klassifikation einzelne HMM-Zustände oder Gruppen von HMM-Zuständen anstelle von Phonemen unterscheiden, so bleibt der grundsätzliche Ablauf unverändert, lediglich die Viterbi-Segmentierung muß dann mit den gewünschten Symbolen erfolgen.

5.4. Weitere Möglichkeiten zur NN/HMM Kombination

5.4.1. Tandem Ansatz

Eine alternative Möglichkeit der NN/HMM-Kombination ist die Modellierung der aus dem NN gewonnenen Auftrittswahrscheinlichkeiten mit Gaußmodellen nach Abschnitt 4.3.3. Bei diesem, in der Literatur als *Tandem*-Modell [Hermansky u. a., 2000] bekanntem Ansatz, wird das NN als zusätzlicher Teil der Merkmalsextraktion verstanden. Der diskriminativ trainierte Ausgangsvektor des NN ist neuer Merkmalsvektor des nachfolgenden Gauß-HMM-Systems.
Die Ausgabe des Klassifikators ist in unveränderter Form nicht durch eine Gaußverteilung darstellbar, da die korrekte Klasse Werte nahe 1 erzeugt, während alle anderen Klassen Wahrscheinlichkeiten nahe 0 erzeugen. Diesem Problem kann entweder durch Logarithmierung der Ausgänge oder durch Weglassen der Nichtlinearität in der Ausgangsschicht (vergl. Gl, (3.1.4)) des NN begegnet werden [Ellis u. a., 2001]. Die logarithmierten Netzausgänge l_j sind in dieser Arbeit nach [Sivadas u. Hermansky, 2002]

$$l_j = \log\left(Pr(\rho_j|\vec{x})\right) - \frac{1}{J}\sum_{k=1}^{J}\log\left(Pr(\rho_k|\vec{x})\right) \tag{5.4.1}$$

implementiert. Zusätzlich wird der Ausgangsvektor einer Hauptachsentransformation unterworfen, mit der die Komponenten des Merkmalsvektors dekorreliert werden, um weiterhin eine Modellierung mit einem Varianzvektor anstelle einer vollen Kovarianzmatrix zu ermöglichen.

5.4.2. Neuronale Vektorquantisierung

Bei Verwendung diskreter Modelle (s. Abschnitt 4.3.1) ist ein Vektorquantisierer nötig, der in der einfachsten Form aus einem *K-Means*-Klassifikator [Linde u. a., 1980] besteht, der alle Merkmalsvektoren in J disjunkte Klassen aufteilt und jeder Klasse einen Mittelwertvektor $\vec{\kappa}_j$ zuordnet. Wird dieser Vektorquantisierer durch ein NN ersetzt, erhält man ein diskretes, hybrides NN/HMM-System [Neukirchen, 1999; Rigoll, 1994a].
Optimierungskriterium dieses NN ist die Maximierung der Transinformation (MMI[3]) zwischen den Netzausgängen und der Phonemfolge bzw. HMM-Zustandsfolge. Beim Training wird, im Gegensatz zu den Verfahren aus Kapitel 3 die Zuordnung der Neuronen zur (segmentierten) Phonem-/Zustandsfolge unüberwacht gefunden, daher spielt die Initialisierung des Netzes hier eine noch wichtigere Rolle, als bei den in dieser Arbeit beschriebenen Algorithmen. Ergebnisse zum diskreten, hybriden NN/HMM-System finden sich in [Neukirchen, 1999] und [Rottland, 2000], dort wird außerdem ein Verfahren zur Adaption des Vektorquantisierers vorgestellt.

5.5. Ergebnisse mit neuronalen Netzen und HMM

Die folgenden Ergebnisse der kompletten hybriden akustischen Modelle sind mit dem sprecherunabhängigen Testset *si-05* der WSJ0-Datenbasis ermittelt worden (s. Anhang B.2). Alle in diesem Abschnitt verwendeten Klassifikatoren sind bereits einzeln im Abschnitt 3.3 evaluiert worden. Für die Abschnitte 5.5.1 und 5.5.2 werden die NN und HMM jeweils mit den Daten des sprecherunabhängigen Trainingssets *si-84* der WSJ trainiert.
Neuronale Netze mit 47 Ausgängen schätzen die Auftrittswahrscheinlichkeiten der 45 Phoneme und 2 Pausenmodelle des LIMSI-Phonemsets (s. Anhang B.2), 139 Ausgänge entsprechen den 139 HMM-Zuständen der genannten 47 HMM. Die einzelnen Phoneme und die lange Pause (*sil*) werden dabei als lineare HMM mit 3 Zuständen modelliert, die kurze Pause *sp* mit einem Zustand (vergl. Abschnitt 4.8). Bild 5.5.1 zeigt die *a priori*-Häufigkeitsverteilung für die einzelnen Phoneme für die WSJ0-Datenbasis. Bei kontextabhängigen Modellen (Triphonen, s. Abschnitt 4.5) ist das NN unverändert, lediglich die kontextunabhängigen HMM sind durch ein Set von Triphonen ausgetauscht worden. Kontextunabhängige Modelle werden in den Ergebnistabellen durch *Mono* gefolgt von der Anzahl der Modelle gekennzeichnet, kontextabhängige Modelle durch *Tri* gefolgt von der Anzahl der HMM. Wie im Abschnitt 4.8 ist auch hier die Wortfehlerrate (WFR) (s. Gl. (4.8.1)) das Maß zur Einschätzung der Qualität der einzelnen Systeme. Der Zusatz *TBC* weist auf zusammengefaßte Parameter (vergl. Anhang E) hin.
Die in den Tabellen 5.1 bis 5.5 verwendeten Netzbezeichner enthalten den Netztyp, die Anzahl der Eingänge, die Anzahl der inneren Neuronen und die Anzahl der Ausgangsneuronen. Beispielsweise ist das in Tabelle 5.2 verwendete MLP mit 273 Eingängen, 1000 versteckten Neuronen und 47 Ausgangsneuronen als *MLP273-1000-47* bezeichnet. Die Anzahl der Eingänge ergibt sich aus Merkmalsvektoren $2m + 1$ aufeinanderfolgender

[3]MMI - *engl.*: Maximum Mutual Information

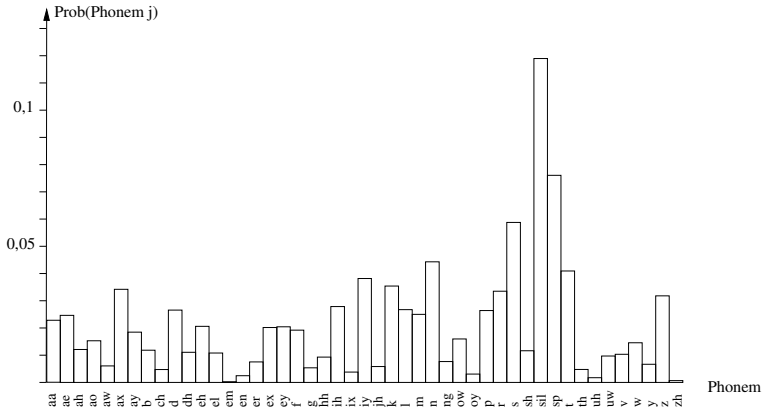


Abbildung 5.5.1.: *A priori*-Wahrscheinlichkeiten der einzelnen Phoneme der WSJ0-Datenbasis, geschätzt auf dem Trainingsset *si-84*

Fenster zu $\vec{x} = (\vec{f}(t-m), \ldots, \vec{f}(t), \ldots, \vec{f}(t+m)^T$ (vergl. Kapitel 3.1.3). Der einzelne Merkmalsvektor besitzt 39 Komponenten (12 MFCCs + Kurzzeitenergie mit dynamischen Merkmalen, s. Kapitel 2).

$$\vec{f}(t) = (c_1, \ldots, c_{12}, e, \Delta c_1, \ldots, \Delta c_{12}, \Delta e, \Delta\Delta c_1, \ldots, \Delta\Delta c_{12}, \Delta\Delta e)^T$$

Merkmalsvektor mit 39 Komponenten

Tabelle 5.1 zeigt Ergebnisse für hybride NN/HMM nach Abschnitt 5.1, während die Tabellen 5.2 bis 5.5 Ergebnisse für verbundene Auftrittswahrscheinlichkeiten (TP) nach Abschnitt 5.2 angeben. Die Ergebnisse sind ferner nach den verwendeten Sprachmodellen (vergl. Abschnitt 4.6) unterteilt: Die für die Tabellen 5.3 und 5.5 erstellten Systeme verwenden den Stack-Dekoder (Abschnitt 4.7.2) mit einem Trigramm-Sprachmodell während die anderen Ergebnistabellen mit dem Viterbi-Dekoder (Abschnitt 4.7.1) und einem Bigramm-Sprachmodell erstellt worden sind.
Zur Einschätzung einer Veränderung der WFR ist eine Betrachtung der Signifikanz dieser Änderung notwendig, allerdings sind die üblichen statistischen Tests aufgrund des stets gleichen Testsets und dem Einfluß des Sprachmodells nicht anwendbar [Gillick u. Cox, 1989]. Als Maß für die Signifikanz wird deshalb die in [Bisani u. Ney, 2004] entwickelte Wahrscheinlichkeit für eine systematische Verbesserung PrV verwendet. Grundlage der Berechnung dieser Wahrscheinlichkeit ist die Vervielfältigung der Testdaten durch wiederholtes Ziehen mit Zurücklegen aus der vorhandenen Menge[4]. Anschließend werden die Ergebnisse der beiden zu vergleichenden Modelle auf diesen vielfachen Testsets evaluiert. Die Wahrscheinlichkeit für eine Verbesserung (des einen Modells gegenüber dem anderen) ergibt sich aus der relativen Häufigkeit, bei der das zweite Modell weniger Fehler als das erste produziert.

[4]Es entsteht eine Monte-Carlo-Schätzung der Statistik der Testdaten

5.5.1. Ergebnisse mit Multi-Layer-Perzeptrons

Tabelle 5.1 präsentiert die Ergebnisse eines hybriden akustischen Modells mit fester NN/HMM-Verknüpfung nach Abschnitt 5.1. Um dem Originalsystem aus [Bourlard u. Morgan, 1994] nahezukommen, wird hier zunächst ein Set von Phonem-HMM mit nur jeweils einem emittierenden Zustand verwendet (gekennzeichnet mit *Mono47-1*, vergl. Abb. 5.1.2). Bei Verwendung der üblichen HMM mit 3 emittierenden Zuständen (*Mono47*) sind im Fall MLP273-1000-47 je drei Zustände eines HMM fest mit dem gleichen, zugehörigen Ausgangsneuron verbunden. Im Fall MLP273-1000-139 sind dann 3 Zustände pro HMM zwingend notwendig, jeder Zustand ist mit dem passenden Ausgangsneuron verbunden. Neben der WFR gibt Tabelle 5.1 auch die Anzahl der Netzparameter an. Bei einer festen Verknüpfung zwischen NN und HMM sind zur Berechnung der HMM-Dichten keine weiteren Parameter notwendig. Mit einer Wahrscheinlichkeit

MLP System	**HMM-System**	**#Param. NN**	**#Param. HMM-Dichten**	**WFR**
MLP273-1000-47	Mono47-1	321047	0	13,64%
MLP273-1000-47	Mono47	321047	0	10,11%
MLP273-1000-139	Mono47	413139	0	8,59%
RNN39-400-47	Mono47-1	196680	0	16,55%
RNN39-400-139	Mono47	237160	0	12,27%

Tabelle 5.1.: WFR von direkt verknüpften NN/HMM-Systemen mit dem *si-05*-Testset, Bigramm-Sprachmodell, Viterbi-Dekoder

MLP System	**HMM-System**	**m**	**#Param. NN**	**#Param. HMM-Dichten**	**WFR**
MLP39-500-47	Mono47	0	43547	6533	12,74%
MLP117-500-47	Mono47	1	82547	6533	11,28%
MLP273-500-47	Mono47	3	160547	6533	11,53%
MLP39-1000-47	Mono47	0	87047	6533	12,14%
MLP273-1000-47	Mono47	3	321047	6533	9.98%
MLP273-1000-139	Mono47	3	413139	19321	8,41%
MLP273-1000-47	Tri10534	3	321047	$1,49 \cdot 10^6$	9,15%
MLP273-1000-47	Tri8379 TBC	3	321047	242614	9,73%
MLP273-1000-139	Tri8379 TBC	3	413139	717518	8,26%
MLP273-1000-139	Tri10534	3	413139	$4,39 \cdot 10^6$	7,55%

Tabelle 5.2.: WFR von MLP/TP-HMM-Systemen mit dem *si-05*-Testset, Bigramm-Sprachmodell, Viterbi-Dekoder

PrV = 89% für MLP273-1000-47 bzw. PrV = 81,7% für MLP273-1000-139 zeigt sich eine signifikante Verbesserung des Modells mit verbundenen Wahrscheinlichkeiten (TP) gegenüber einer festen Verknüpfung mit dem NN (Vergleich der Tabellen 5.1 und 5.2). Eine Triphon-System ist mit einer festen NN/HMM-Verknüpfung nicht möglich, gegenüber einem kontextunabhängigen TP-Monophon-System ergibt sich bei Triphonen eine relative Verbesserung von 8,3% (MLP273-1000-47) bzw. 10,2% (MLP273-1000-139) mit PrV = 100% in beiden Fällen. Diese Verbesserung muß allerdings durch eine stark vergrößerte Anzahl an Parametern in den HMM erkauft werden. Das Zusammenfassen von Triphon-Zuständen (TBC) führt zu einer Verschlechterung (das nicht zusammengefaßte Modell ist mit PrV > 97% besser) bei einer stark verringerten Anzahl an Parametern.
Im Vergleich zu Gauß-HMM (vergl. Tabellen 4.1 und 4.2) fällt der Gewinn vom Übergang Monophon-Triphon hier geringer aus. Ein Grund dafür ist hauptsächlich die gute Qualität des Monophon-NN/TP-HMM-Systems, die die aller Gauß-Modelle deutlich übertrifft, zusätzlich kann die Kontextabhängigkeit der Triphone nur in den Mixturgewichten modelliert werden, da der Klassifikator unverändert bleibt.
Bei Verwendung des Trigramm-Sprachmodells ergibt sich erneut eine signifikante Ver-

MLP System	**HMM-System**	**m**	**#Param. NN**	**#Param. HMM-Dichten**	**WFR**
MLP273-1000-47	Mono47	3	321047	6533	7,49%
MLP273-1000-139	Mono47	3	413139	19321	6,16%
MLP273-1000-47	Tri10534	3	321047	$1,49 \cdot 10^6$	6,54%
MLP273-1000-139	Tri10534	3	413139	$4,39 \cdot 10^6$	5,38%

Tabelle 5.3.: WFR von MLP/TP-HMM-Systemen mit dem *si-05*-Testset, Trigramm-Sprachmodell, Stack-Dekoder

besserung (PrV = 100% in allen Fällen), der absolute Gewinn liegt bei ungefähr 2,5%. Die absolute Verbesserung beim Übergang zur kontextabhängigen Modellierung liegt, wie in Tabelle 5.2, bei ungefähr 1%, die relative Verbesserung ist hier aufgrund der niedrigeren WFR etwas größer (12,7% bei 47 und 139 NN-Ausgängen).
Das Optimum bezüglich der gegenläufigen Größen WFR und Anzahl Parameter des Systems ergibt sich beim TP-Modell mit MLP273-1000-139 (trainiert auf HMM-Zustände): Bei rund $4,3 \cdot 10^5$ Parametern ist die WFR niedriger als bei einem Triphon-Modell mit 47 NN-Ausgängen und $1,5 \cdot 10^6$ Parametern.

5.5.2. Ergebnisse mit rekurrenten neuronalen Netzen

Die rekurrenten Netze (RNN) benutzen im Gegensatz zu den MLPs nur jeweils einen Merkmalsvektor mit 39 Komponenten als Eingang (12 MFCCs mit Kurzzeitenergie und dynamischen Merkmalen). Zur Modellierung zukünftiger Kontexte ist der Ausgangsvektor um τ Zeitschritte (Fenster) verzögert (s. Abschnitt 3.1.3). Die RNN/TP-HMM-

RNN System	HMM-System	τ	#Param. NN	#Param. HMM-Dichten	WFR
RNN39-400-47	Mono47	3	196680	6533	12,82%
RNN39-300-47	Mono47	3	117980	6533	13,88%
RNN39-300-47	Mono47	0	117980	6533	14,85%
RNN39-279-47	Mono47	3	103994	6533	13,04%
RNN39-400-139	Mono47	3	237160	19321	10,91%
RNN39-400-139	Tri10534	3	237160	$4,39 \cdot 10^6$	9,49%
RNN39-400-139	Tri8379 TBC	3	237160	717518	9,56%

Tabelle 5.4.: WFR von RNN/TP-HMM-Systemen mit dem *si-05*-Testset, Bigramm-Sprachmodell, Viterbi-Dekoder

Systeme aus Tabelle 5.4 zeigen generell eine etwas schlechtere WFR, als MLP/HMM-Systeme aus Tabelle 5.2, aber auch hier ist das TP-Modell signifikant besser als die fest verknüpften RNN/HMM aus Tabelle 5.1 (PrV = 100%). Das RNN39-400-139 weist nur rund 20% mehr Parameter als das RNN39-400-47 bei einer signifikant verbesserten Erkennungsrate (PrV = 99%) auf. Offensichtlich ist beim RNN das Training mit möglichst vielen überwachten Parametern entscheidend, um das RNN auch an schwierige, schnell aufeinanderfolgende Phonemwechsel anzupassen. Auch bei den RNN/HMM-Systemen

RNN System	HMM-System	τ	#Param. NN	#Param. HMM-Dichten	WFR
RNN39-400-47	Mono47	3	196680	6533	9,62%
RNN39-400-139	Mono47	3	237160	19321	8,56%
RNN39-400-139	Tri10534	3	237160	$4,39 \cdot 10^6$	7,42%

Tabelle 5.5.: WFR von RNN/TP-HMM-Systemen mit dem *si-05*-Testset, Trigramm-Sprachmodell, Stack-Dekoder

läßt sich durch die Verwendung eines Trigramm-Sprachmodells ein absoluter Gewinn von ca. 2,5% erzielen. Die WFR-Reduktion durch das verbesserte Sprachmodell ist also unabhängig von der Art des verwendeten akustischen Modells.

Die Einführung von Zusatzaufgaben bei der Klassifikation mit RNN (vergl. Abschnitt 3.1.5) führt nur unter bestimmten Bedingungen zu einer Verbesserung der Erkennungsleistung. Aus Tabelle 5.6 kann geschlossen werden, daß die verwendeten Zusatzaufgaben unabhängig von der Hauptaufgabe sein müssen, um einen Gewinn zu erzielen. Die Anforderung, möglichst viele überwachte Ausgänge beim RNN-Training zu verwenden, muß um die Unabhängigkeit der Aufgaben erweitert werden. Die allgemeinen Phonemklassen sind komplett redundant, da sie aus dem Phonemset erzeugt worden sind und auch die Grapheme sind stark mit den Phonemen korreliert. Lediglich das Geschlecht des Spre-

RNN System	Zusatztask	HMM-System	τ	WFR
RNN39-400-47	Geschlecht	Mono47	3	12,65%
RNN39-400-47	Grapheme	Mono47	3	13,47%
RNN39-400-47	allg. Phonemklassen	Mono47	3	13,94%
RNN39-400-139	Geschlecht	Mono47	3	10,20%
RNN39-400-139	Grapheme	Mono47	3	11,30%
RNN39-400-139	allg. Phonemklassen	Mono47	3	11,28%
RNN39-400-139	Geschlecht	Tri10534	3	9,04%
RNN39-400-139	Geschlecht	Tri8379 TBC	3	9,53%

Tabelle 5.6.: WFR von RNN/TP-HMM-Systemen mit Zusatzaufgaben auf den *si-05*-Testdaten, Bigramm-Sprachmodell, Viterbi-Dekoder

chers ist unabhängig von gesprochenen Inhalt und erzeugt eine signifikante Verbesserung gegenüber dem RNN39-400-139-System ohne Zusatzaufgaben (PrV = 94%).
Das MLP273-1000-47 ist gegenüber dem besten RNN (RNN39-400-139 mit Sprechergeschlecht als Zusatzaufgabe, vergl. Tabelle 5.6) nur unwesentlich besser (PrV = 69%), allerdings sind beim MLP über 35% mehr Parameter notwendig.
Ergebnisse der NN/TP-HMM-Systeme mit verknüpften Wahrscheinlichkeiten unter Benutzung der AURORA2-Datenbasis finden sich im Kapitel 7, da sie dort zum Vergleich mit verteilten Spracherkennern herangezogen werden.
Insgesamt läßt sich die Wahl des geeigneten Klassifikators für ein TP-Modell also allein anhand der gewünschten Qualität und der Anzahl der Parameter treffen. Gemeinsam ist allen TP-Systemen die deutliche Überlegenheit gegenüber Gauß-HMM. Bei weniger Parametern ist die WFR der meisten NN/TP-HMM-Systeme deutlich niedriger als bei einem kontextabhängigen Triphon-Gauß-Modell.

5.5.3. Ergebnisse mit dem TANDEM-Ansatz

Obwohl der TANDEM-Ansatz ebenfalls ein NN enthält, ist die Struktur des Erkennungssystems mehr an dem im Kapitel 4 vorgestellten Ansatz orientiert, da hier ebenfalls Gauß-HMM mit mehreren Mixturen trainiert werden. Wie im Abschnitt 5.4.1 dargestellt, dient das NN hier als ein weiteres Element der Vorverarbeitung. Zur Evaluation des TANDEM-Ansatzes werden für den Test *si-05* der WSJ-Datenbasis die NN aus den Abschnitten 5.5.1 bzw. 5.5.2 verwendet. Die NN für Tests mit der AURORA2-Datenbasis sind im Abschnitt 7.3.2 beschrieben. Die Systeme mit dem Zusatz *log* schalten hinter dem NN einen Logarithmierer nach [Sivadas u. Hermansky, 2002], s. Abschnitt 5.4.1, während die anderen Systeme die Softmax-Funktion aus dem Netz deaktivieren. Letzteres ist mit PrV = 82% signifikant besser.

Die Ergebnisse des TANDEM-Ansatzes mit dem WSJ-Test sind im hier dargestellten Systemaufbau nicht überzeugend und deutlich schlechter als Gaußmodelle und als die anderen hybriden Modelle. In der Literatur [Zhu u. a., 2004] werden sehr gute Ergebnisse

System	Anzahl HMM	Mixturen	#Param. HMM-Dichten	WFR
MLP273-1000-47 log TANDEM	47	10	130660	20,99%
MLP273-1000-47 log TANDEM	47	12	156792	20,79%
MLP273-1000-47 TANDEM	47	10	130660	20,47%
MLP273-1000-47 TANDEM	47	12	156792	19,73%

Tabelle 5.7.: WFR verschiedener TANDEM-Systeme mit dem *si-05*-Testset, Bigramm-Sprachmodell, Viterbi-Dekoder

mit einer Kombination von MFCC oder PLP mit den "Tandem-Merkmalen" erreicht, der dann sehr große Merkmalsvektor muß dann allerdings nachträglich mit einer LDA[5] wieder verkleinert werden.

5.6. Ergebnisse mit Support-Vektor-Maschinen

Abschließend werden Ergebnisse mit HMM und den aus Abschnitt 3.2 bekannten SVM mit den Testsets A, B und C der AURORA2-Datenbasis (Anhang B.3) vorgestellt. Da bei der "1-gegen-alle"-Kombination binärer Klassifikatoren für jede Klasse eine SVM benötigt wird, ist für den AURORA2-Test daher zunächst eine wortbasierte Klassifikation realisiert worden, bei der 13 SVM für die 11 Zahlwörter und 2 Pausenmodelle notwendig sind (vergl. Abschnitte 7.3 und B.3). Bei der Evaluation zeigte sich jedoch, daß ein solches System Ziffernketten aus dem gleichen Zahlwort ("*one one*") nicht auflösen kann, sondern als ein Wort erkennt. Für den AURORA2-Test existieren Ganzwort-HMM mit

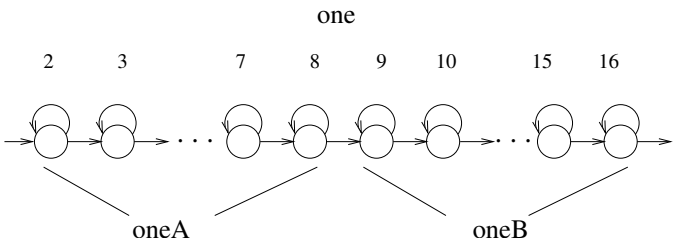


Abbildung 5.6.1.: Zusammenfassen von HMM-Zuständen, um Zielwerte für SVM zu erhalten

16 Zuständen pro Modell (s. Abschn. 7.3). Werden je 8 Zustände, wie in Abb. 5.6.1

[5]LDA - *engl.*: Linear Discriminant Analysis

zusammengefaßt, so entstehen Klassen jeweils für die 1. und 2. Worthälfte. Wird noch eine Pausenklasse hinzugenommen, so entstehen insgesamt 23 Klassen [Stadermann u. Rigoll, 2004], für die 23 SVM trainiert worden sind. Die Verbindung zwischen den SVM und HMM geschieht nach dem Ansatz aus Abschnitt 5.2 (TP), wobei alle SVM (jeweils mit einer Sigmoidfunktion zur Ausgabe von Wahrscheinlichkeiten) als ein Klassifikator aufgefaßt werden. Eine Verwendung der 48 Pseudo-Phoneme aus Abschnitt 7.3 wäre wünschenswert, ist aber aufgrund des Aufwandes zum Trainieren von 48 SVM mit den vorhandenen Algorithmen nicht möglich.

System	**m**	**Test A**	**Test B**	**Test C**	**Durchschnitt**
AURORA2-Ref.	–	38,66%	44,25%	33,86%	39,94%
13 SVM-RASTA-2000 TP	0	61,11%	59,76%	60,82%	60,51%
23 SVM-RASTA-6000 TP	1	37,55%	33,62%	37,24%	35,92%

Tabelle 5.8.: WFR verschiedener SVM/HMM-Systeme auf dem AURORA2-Testset

Tabelle 5.8 zeigt die WFR der kompletten SVM/HMM-Systeme trainiert mit den AURORA2-Trainingsdaten ohne Hintergrundgeräusch. Zum Vergleich ist ein AURORA2-Referenzergebnis aus [Hirsch u. Pearce, 2000] angegeben. Zu erkennen ist ein deutlicher Gewinn der 23 SVM-RASTA-6000 beim Test B (10,6% absolut). Das Ergebnis beim Test A ist um 1% absolut gegenüber der Referenz verbessert, während sich beim Test C trotz Verwendung von RASTA-Merkmalen, eine Verschlechterung von rund 3% einstellt. Die mit unverrauschten Merkmalen gefundenen Support-Vektoren sind also robust gegenüber additivem Rauschen, aber sehr empfindlich gegenüber Kanalveränderungen.

6. Adaption hybrider akustischer Modelle auf einen neuen Sprecher

Bei fast allen Anwendungen der Spracherkennung kommuniziert ein einzelner Sprecher[1] mit dem System. Daher ist es eigentlich immer sinnvoll, das akustische Modell auch schon bei kurzen Äußerungen an den aktuellen Sprecher anzupassen bzw. zu adaptieren, um Erkennungsfehler zu minimieren. Dieses Kapitel stellt nach einer kurzen Einführung in die Adaptionsverfahren mehrere Methoden vor, um die hybriden akustischen Modelle aus Kapitel 5 an neue Sprecher zu adaptieren.

6.1. Adaptionsverfahren zur Sprecheradaption

Die in den Kapiteln 4 und 5 vorgestellten Spracherkenner sind *sprecherunabhängig* (SI[2]) trainiert. Die Idee dabei ist, durch möglichst viele verschiedene Sprecher im Trainingsmaterial eine große Variation von Sprechereigenschaften zu erfassen, um auch Äußerungen neuer, unbekannter Sprecher mit akzeptabler Genauigkeit zu erkennen.
Auf der anderen Seite ist die bestmögliche Qualität eines Spracherkenners mit den beschriebenen Methoden dann zu erreichen, wenn ein *sprecherabhängiges* System mit Daten von nur einem Sprecher trainiert wird. Dieses System hat jedoch nur eingeschränkten Nutzen, da für jeden Sprecher, der das System benutzen möchte, ein eigenes Modell trainiert werden muß.
Als Mittelweg zwischen dem sprecherunabhängigen und -abhängigen System bietet sich die Adaption der sprecherunabhängigen Modelle auf einen neuen Sprecher an (vergl. Abbildung 6.1.1). Der Vorteil hierbei ist, daß zur Adaption eines akustischen Modells nur ein Bruchteil von Daten (verglichen mit einem Neutraining) erforderlich ist. Bereits eine Adaption mit einigen wenigen Äußerungen reicht aus, um eine Verbesserung der Erkennungsgenauigkeit für den adaptierten Sprecher zu erzielen.
Für die in Kapitel 4 beschriebenen HMM mit gaußförmigen Mixturdichten existieren eine Vielzahl von Adaptionsverfahren, von denen hier drei wichtige Grundideen vorgestellt werden:

- Bei der Adaption durch *Maximum Likelihood Linear Regression* (MLLR) [Leggetter u. Woodland, 1995; Gales, 1998] werden die Parameter der Ausgabedichten (meist nur die Mittelwerte der Gaußdichten) durch Regressionsmatrizen miteinander verknüpft. Die Komponenten dieser Matrizen werden dann aus den Adaptionsdaten geschätzt. Ein wichtiger Vorteil dieser Methode ist, daß prinzipiell immer

[1] *Sprecher* meint abkürzend sowohl männliche Sprecher, als auch weibliche Sprecherinnen
[2] SI - *engl.* : speaker independent

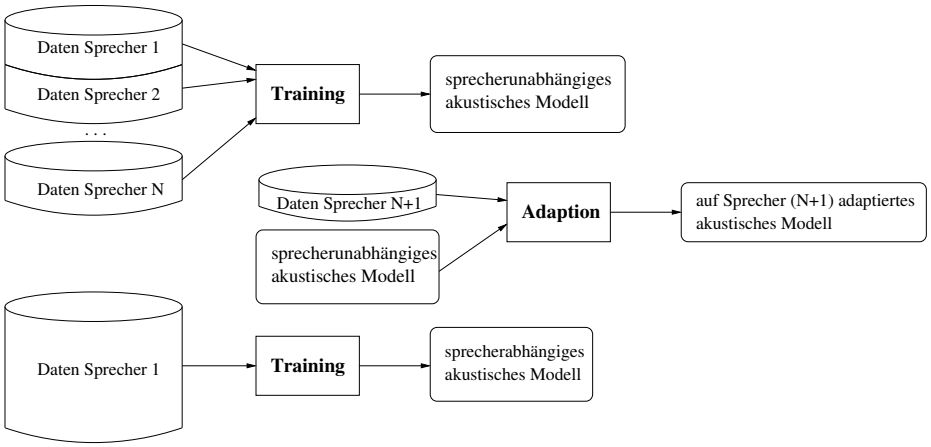


Abbildung 6.1.1.: Vergleich von Training und Adaption

alle Modelle durch die Adaption verändert werden. Zur deutlichen Verbesserung der MLLR-adaptierten Modelle sind jedoch verhältnismäßig viele Adaptionsdaten notwendig. Alle anderen Parameter der HMM (Übergangswahrscheinlichkeiten, Gewichtungskoeffizienten der Mixturen, Varianzen) bleiben unverändert.

- Die *Maximum A Posteriori*-Adaption (MAP) [Gauvain u. Lee, 1991, 1994] basiert auf der Neuschätzung der HMM-Parameter unter Benutzung von *a priori*-Annahmen bezüglich der zu schätzenden Dichte. Hierbei werden nur Parameter von Modellen adaptiert, die in den Adaptionsdaten beobachtet worden sind, allerdings reicht meist eine kleine Menge an Adaptionsdaten aus, um eine gute Adaption zu erreichen.

- Eine weitere Möglichkeit ist, die Trainingsdaten nach Sprechern zu sortieren und sprecherabhängige Modelle für jeden Sprecher zu trainieren. Aus diesen Modellen können dann durch Hauptachsentransformation die *Eigenvoices* [Kuhn u. a., 1998] gewonnen werden. Ein neuer, unbekannter Sprecher ist als Linearkombination der Eigenvoices darstellbar, die Gewichtung der einzelnen Eigenvoices wird aus den Adaptionsdaten ermittelt. Meist werden hier auch nur die Mittelwerte der Gaußdichten adaptiert, alle anderen Parameter bleiben unverändert. Ein Vorteil dieses Verfahrens ist der sehr geringe Bedarf an Adaptionsdaten, um einen neuen Sprecher zu adaptieren.

6.2. Adaption des neuronalen Netzes

Die oben beschriebenen Adaptionsverfahren eignen sich generell zunächst nicht zur Adaption von hybriden akustischen Modellen aus Kapitel 5, da in diesen Modellen keine Mittelwertvektoren von Ausgabedichten vorliegen. Es bieten sich jedoch neue Möglichkeiten, z.B. die Adaption des Klassifikators unabhängig von den HMM. In diesem Ab-

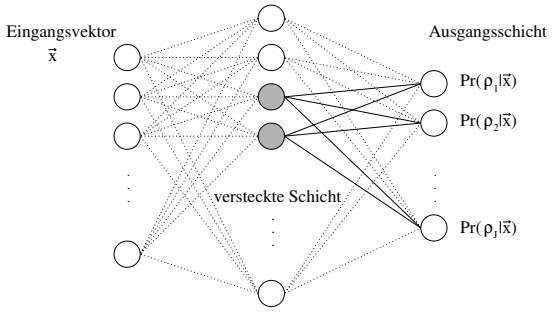


Abbildung 6.2.1.: Partielle Adaption eines MLPs, die zu adaptierenden Gewichte sind hervorgehoben

schnitt werden nur Adaptionsmethoden für NN-Klassifikatoren behandelt. Ansätze zur Adaption von SVM finden sich in [Syed u. a., 1999]. [Neto u. a., 1995] beschreibt mehrere Möglichkeiten zur Adaption eines NN in einem hybriden System, unter anderem das Nachtrainieren des kompletten Netzes mit den Adaptionsdaten oder das Hinzufügen einer weiteren Schicht vor dem Originalnetz. In ähnlicher Weise wird auch in [Ström, 1996] das Hinzufügen von Sprecherknoten beschrieben, die Einfluß auf die Phonemklassifikation nehmen. [Rottland u. a., 1998] beschreibt die Adaption von diskreten, hybriden NN/HMM Systemen (s. Abschnitt 5.4.2) durch Hinzufügen einer weiteren Netzschicht. In dieser Arbeit wird das Hinzufügen von weiteren Schichten oder Knoten zum Netz nicht weiter verfolgt, da vor allem bei MLPs bereits ein sehr großer Eingangsvektor (typisch sind 273 Elemente, s. Abschnitt 3.3.1) vorhanden ist. Ein Hinzufügen einer weiteren, voll verbundenen Adaptionsschicht vor dem NN würde hier $273^2 = 74529$ zu adaptierende Parameter bedeuten. Bei den für die Experimente vorliegenden Adaptionsdaten (etwa 28000 Zeitfenster bzw. 40 gesprochene Sätze eines Sprechers) wäre eine statistisch sichere Schätzung aufgrund der geringen Datenmenge nicht mehr möglich.
Stattdessen wird ein Algorithmus zum partiellen Nachtrainieren der Netzgewichte vorgestellt. In einem MLP ist eine Möglichkeit hierzu das Adaptieren von Gewichten von der versteckten Schicht zur Ausgangsschicht. Diese Parameter bestimmen unmittelbar den Wert der Auftrittswahrscheinlichkeiten und bieten sich demnach zur Adaption an [Stadermann u. Rigoll, 2005a]. Abbildung 6.2.1 zeigt die zu adaptierenden Gewichte in einem MLP. In einem RNN können analog dazu die Gewichte zwischen Rückkopplungs- und Ausgangsknoten adaptiert werden (Abbildung 6.2.2). Um die Zahl der Parameter zu begrenzen, werden nur jeweils die "wichtigen" Knoten der vor der Ausgangsschicht liegenden Ebene ausgewählt (in den Abbildungen hervorgehoben). Ein mögliches Kriterium zur Auswahl der zu adaptierenden Neuronen ist die Varianz der Neuronenausgänge berechnet mit den Adaptionstrainingsdaten. Folgende Schritte sind hierzu notwendig:

- Anlegen der gesamten Adaptionsdaten an das NN und Berechnen der Varianz $\mathrm{var}(z_k)$ der versteckten Neuronen bzw. der Rückkopplungsneuronen über diese Daten
- Bestimmen des Neurons mit maximaler Varianz

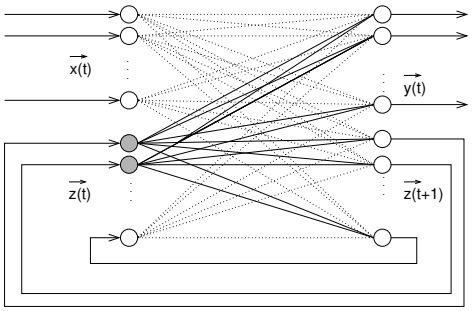


Abbildung 6.2.2.: Partielle Adaption eines RNNs, die zu adaptierenden Gewichte sind hervorgehoben

- Auswahl eines Neurons zur Adaption, falls die Bedingung

$$\mathrm{var}(z_k) > \eta \max_k (\mathrm{var}(z_k)) \tag{6.2.1}$$

  erfüllt ist

Der einstellbare Parameter η steuert dabei die Anzahl der zu adaptierenden Neuronen.

6.3. Adaption der Hidden-Markov Modelle

Akustische NN/TP-HMM-Systeme, bei denen die Netzausgänge mit allen HMM-Zuständen verbunden sind (s. Abschnitt 5.2), erlauben sowohl die Adaption des NNs (s. Abschnitt 6.2), als auch die Adaption von HMM-Parametern. Interessant sind hier die Gewichtungsfaktoren c_{ij}, mit denen die Netzausgänge für jeden HMM-Zustand gewichtet werden. Im Folgenden werden einige Verfahren zur Adaption dieser Faktoren vorgestellt. Wie schon bei den Adaptionsverfahren für Gaußmodelle werden die HMM-Übergangswahrscheinlichkeiten nicht adaptiert.

6.3.1. Adaption der HMM-Parameter durch Gradientenanstieg

Die Adaption durch Gradientenanstieg benutzt eine ähnliche Optimierungsfunktion, wie sie schon in [Wallhoff u. a., 2000, 2001] eingesetzt worden ist, um die Adaption von Gauß-Mittelwerten mittels linearer Regression skalierter Likelihoods[3] durchzuführen. Ausgangspunkt der skalierten Likelihood ist die Überlegung, daß in der Trainingsphase nicht nur die Wahrscheinlichkeitsdichte des akustischen Modells $p(X|M)$, sondern

[3] *engl.*: Scaled Likelihood Linear Regression, ein diskriminatives Adaptionsverfahren von MLLR-Regressionsmatrizen

auch die Größe $p(X)$ von den Parametern des Modells abhängt[4]. Der optimale HMM-Parametersatzes λ^* läßt sich dann schreiben als

$$\lambda^* = \underset{\lambda}{\operatorname{argmax}} \left\{ \frac{p(X|M,\lambda)}{p(X|\lambda)} \right\}, \tag{6.3.1}$$

wobei X wieder die Folge der akustischen Beobachtungen und M die zu der gesprochenen Äußerung passende HMM-Modell- bzw. Zustandsfolge bezeichnet. Zur Näherung der Größe $p(X|\lambda) = \sum_{alleM} Pr(M)p(X|M,\lambda)$ wird die Annahme getroffen, daß eine (zeit-)schrittweise Betrachtung $p(X|\lambda) \approx \sum_{t=1}^{T} p(\vec{x}(t))$ ausreicht und Abhängigkeiten zwischen den einzelnen Beobachtungen vernachlässigbar sind. Die Komplexität bei der Berechnung ist dadurch deutlich gesunken, da anstelle einer Auswertung einer N-Best-Liste oder ähnlicher Maßnahmen lediglich die Größe $p(\vec{x})$ mit $p(\vec{x}(t)|\lambda) \approx \sum_{q_i \in \mathbb{Q}} Pr(q_i)p(\vec{x}(t)|q_i)$ bestimmt werden muß.
Gl. (6.3.1) wird also durch

$$\lambda^* = \underset{\lambda}{\operatorname{argmax}} \left\{ \frac{p(\vec{x}(t)|q_V(t),\lambda)}{\sum_{q_i \in \mathbb{Q}} Pr(q_i)p(\vec{x}(t)|q_i)} \right\} \tag{6.3.2}$$

ersetzt, $q_V(t)$ bezeichnet hier den durch eine (Viterbi-)Segmentierung dem Fenster t zugeordneten HMM-Zustand. Die zu optimierende Wahrscheinlichkeitsdichte ist das Produkt der Einzelgrößen aus der rechten Seite von Gl. (6.3.2) über alle Adaptionsdaten. Durch Logarithmieren entsteht der einfachere Ausdruck

$$\mathfrak{L}' = \sum_{t=1}^{T} \log p(\vec{x}(t)|q_V(t),\lambda) - \log \left(\sum_{q_i \in \mathbb{Q}} Pr(q_i)p(\vec{x}(t)|q_i,\lambda) \right) \tag{6.3.3}$$

als Optimierungsfunktion. Bei hybriden akustischen Modellen nach Abschnitt 5.2 sind die Parameter λ gleich den Gewichtungsfaktoren c_{ij} aus Gl. (5.2.1). Zu berechnen wäre also der Gradient $\frac{\partial \mathfrak{L}'}{\partial c_{ij}}$. Da Gradientenverfahren den zu optimierenden Parameter jedoch prinzipiell unbeschränkt variieren, muß noch durch Einführung einer Transformation

$$c_{ij} = \frac{\exp(\kappa_{ij})}{\sum_{k=1}^{K} \exp(\kappa_{ik})} \tag{6.3.4}$$

dafür gesorgt werden, daß die Gewichtungsfaktoren immer den in Abschnitt 5.2 genannten Einschränkungen gehorchen. Der Gradient der Optimierungsfunktion ergibt sich damit (vergl. [Stadermann u. Rigoll, 2005a]) zu

$$\frac{\partial \mathfrak{L}'}{\partial \kappa_{ij}} = \sum_{t=1}^{T} \delta_{q_V(t),q_i} \frac{c_{ij}\left(Pr(\rho_j|\vec{x}(t)) - p(\vec{x}(t)|q_i)\right)}{p(\vec{x}(t)|q_i)} - \frac{Pr(q_i)c_{ij}\left(Pr(\rho_j|\vec{x}(t)) - p(\vec{x}(t)|q_i)\right)}{\sum_{q_k \in \mathbb{Q}} Pr(q_k)p(\vec{x}(t)|q_k)} \tag{6.3.5}$$

[4]Wird diese Überlegung vernachlässigt, so erhält man die übliche Maximum-Likelihood Formulierung (s. Abschnitt 4.4)

($\delta_{q_V(t),q_i}$ bezeichnet das Kroneckersymbol: $\delta_{q_V(t),q_i} = 1$, falls $q_V(t) = q_i$ und $\delta_{q_V(t),q_i} = 0$ sonst). Die *a priori*-Wahrscheinlichkeit $Pr(q_i)$ kann wieder aus den relativen Häufigkeiten der Segmentierung der Trainingsdaten ermittelt werden. Initialisiert wird das Verfahren mit den Parametern des sprecherunabhängigen Modells: $\kappa_{ij} = \log(c_{ij})$. Insgesamt ergibt sich also folgender Ablauf:

1. Initialisierung der Parameter $\kappa_{ij} = \log(c_{ij})$
2. Aufsummierung des Gradienten (Gl. (6.3.5)) über 75% der Adaptionsdaten
3. Neuberechnung der Gewichte mit $\kappa_{ij}^{(n+1)} = \kappa_{ij}^{(n)} + \beta \frac{\partial \mathfrak{L}'}{\partial \kappa_{ij}}$ oder nach dem RPROP-Verfahren [Igel u. Hüsken, 2000]
4. Neuberechnung der HMM-Parameter nach Gl. (6.3.4)
5. Kreuzvalidierung auf den verbliebenen 25% der Adaptionsdaten
6. Abbruch, falls eine festgelegte Anzahl Iterationen erreicht ist oder Start einer neuen Iteration (zurück zu Schritt 2)

Zur Validierung von Schritt 5 wird die Phonemfehlerrate[5] mit dem kompletten NN/HMM-System auf einem nicht verwendeten Teil der Trainingsdaten (typisch 10%-25%) berechnet. Die Phonemfehlerrate zeigt den Gewinn der Adaption bei der richtigen Klassifizierung der Phoneme ohne Einfluß des Sprachmodells und ist wesentlich schneller zu berechnen, als eine Worterkennung. Nach einer festgelegten Anzahl von Iterationen kann das Modell mit dem besten Kreuzvalidierungsergebnis für einen Test verwendet werden.
Ein praktisches Problem, welches bei einer kleinen Menge an Adaptionsdaten immer auftauchen kann, ist die Abwesenheit einiger Phoneme des Sets in diesen akustischen Daten. Im Allgemeinen wird eine Schwelle an Mindestbeobachtungen (typischerweise 2-4) gesetzt, um eine sinnvolle Schätzung zu erhalten. Bei der Adaption durch Gradientenanstieg werden die Modelle zu wenig beobachteter Phoneme durch das entsprechende sprecherunabhängige Modell ersetzt.

6.3.2. Adaption der HMM-Parameter durch Maximierung der a posteriori-Wahrscheinlichkeit

Bei dem im Abschnitt 4.4 beschriebenen ML-Training werden die HMM-Parameter so eingestellt, daß die Produktionswahrscheinlichkeitsdichte $p(X|\lambda)$ auf den Trainingsdaten für alle Parameter unabhängig voneinander maximiert wird. Dieses Verfahren garantiert allerdings nur für unendlich viele Trainingsdaten optimale Parameter. Bei der Adaptionsaufgabe mit nur wenigen Adaptionsdaten bietet sich daher eine Maximierung der *a posteriori* Wahrscheinlichkeitsdichte (MAP) $p(\lambda|X) = \frac{1}{p(X)} p(X|\lambda)p(\lambda)$ an. In [Gauvain u. Lee, 1994] finden sich Nachschätzformeln für die Parameter einer Gauß-Mixturdichte unter der Bedingung, daß sich die *a priori*-Dichte $p(\lambda)$ als ein Produkt einer

[5] hier mit dem kompletten Erkenner und nicht zu Verwechseln mit der PFR aus Abschnitt 3.3

Dirichlet- und einer Gamma-Normalverteilung ergibt. Weiter sind in [Huo u. Chan, 1995; Huo u. Lee, 1997] MAP-Nachschätzformeln für alle Parameter von diskreten und semi-kontinuierlichen HMM abgeleitet. Ausgehend von der Nachschätzformel für die Mixturgewichte für semi-kontinuierliche Modelle aus [Huo u. Chan, 1995] (mit angepaßter Notation)

$$c_{ij}^{(n+1)} = \frac{\beta(\nu_{ij}^{(n)} - 1) + \exp\left(\sum_{t=1}^{T} \zeta_{ij}(t)\right)}{\sum_{k=1}^{J}\left[\beta(\nu_{ik}^{(n)} - 1) + \exp\left(\sum_{t=1}^{T} \zeta_{ik}(t)\right)\right]} \tag{6.3.6}$$

ergibt sich mit der Annahme $\nu_{ij} - 1 = \frac{c_{ij}}{Pr(q_i)}$ (vergl. auch [Gauvain u. Lee, 1992]):

$$c_{ij}^{(n+1)} = \frac{\frac{\beta}{Pr(q_i)} c_{ij}^{(n)} + \exp\left(\sum_{t=1}^{T} \zeta_{ij}(t)\right)}{\frac{\beta}{Pr(q_i)} + \exp\left(\sum_{t=1}^{T} \gamma_i\right)} \tag{6.3.7}$$

Der Ablauf der Adaption ist ähnlich dem aus Abschnitt 6.3.1:

1. Initialisierung der Adaption mit den Parametern eines sprecherunabhängigen Modells
2. Berechnen der Größen ζ_{ij} und γ_i aus 75% der Adaptionstrainingsdaten mit dem Baum-Welch-Algorithmus aus Abschnitt 4.4
3. Anwendung von Gleichung (6.3.7) zur Bestimmung neuer Mixturgewichte c_{ij}, falls genügend Beobachtungen des entsprechenden HMM-Zustandes vorhanden sind, ansonsten wird der Gewichtungsfaktor nicht verändert
4. Kreuzvalidierung auf den verbliebenen 25% der Adaptionsdaten
5. Abbruch, falls eine festgelegte Anzahl Iterationen erreicht ist oder Start einer neuen Iteration (2)

Zur Validierung von Schritt 4 wird wiederum die Phonemfehlerrate mit dem kompletten NN/HMM-System auf einem nicht verwendeten Teil der Trainingsdaten berechnet. Aus Schritt 3 ist ersichtlich, daß nur Zustände adaptiert werden, die oft genug (die Schwelle liegt bei 3 Beobachtungen) in den Trainingsdaten vorkommen. Seltene Modelle werden bei diesem Verfahren also unter Umständen überhaupt nicht adaptiert.

6.3.3. Adaption der HMM-Parameter mit Eigenvoices

Geht es darum, aus sehr wenigen Adaptionsdaten verbesserte Modelle zu schätzen, so bietet sich die Adaption mit *Eigenvoices* nach [Kuhn u. a., 1998, 1999] an. In [Kuhn u. a., 2000] ist eine Interpretation und eine Ableitung des Verfahrens beschrieben, das hier kurz zusammengefaßt wird. Zunächst teilt sich das Verfahren in einen Schritt zur Berechnung der Eigenvoices und in den eigentlichen Adaptionsschritt auf.
Zur Berechnung der Eigenvoices werden zunächst komplette sprecherabhängige Parameter für möglichst viele verschiedene Sprecher trainiert. Anschließend werden die zu

adaptierenden Parameter mittelwertfrei als Spaltenvektoren in eine Matrix eingetragen, von der dann mit Hilfe der Hauptachsentransformation, die *Eigenvoices* berechnet werden. Im Falle von Gauß-HMM können in bewährter Weise die Mittelwertvektoren aller Gaußdichten in diese Matrix eingetragen werden [Botterweck, 2000]. Um die Eigenvoice-Adaption für hybride akustische Modelle zu nutzen, können bei TP-Modellen nach Abschnitt 5.2 wieder die Mixturgewichte c_{ij} der HMM adaptiert werden. Für S verschiedene Sprecher in den Trainingsdaten ist die Kovarianzmatrix $\mathbf{C} = \frac{1}{S}\sum_{s=1}^{S}\left(\vec{c^{(s)}} - \vec{\mu}_C\right)\left(\vec{c^{(s)}} - \vec{\mu}_C\right)^T$ mit den Parametervektoren $\vec{c^{(s)}} = \left(c_{11}^{(s)}, \ldots, c_{IJ}^{(s)}\right)^T$ und dem Mittelwertvektor $\vec{\mu}_C = \sum_{s=1}^{S}\vec{c^{(s)}}$ zu berechnen. Die zur Adaption benötigten Eigenvektoren[6] der Matrix $\mathbf{C}$ (die Eigenvoices) werden spaltenweise in die Matrix $\mathbf{E}$ eingetragen.
Im eigentlichen Adaptionsschritt wird aus den Adaptionsdaten ein sprecherabhängiges Modell trainiert, welches dann in den Eigenvoice-Raum projiziert wird. Ist $\vec{v} = (c'_{11}, \ldots, c'_{IJ})^T$ der Vektor aller aus den Adaptionsdaten trainierten Gewichtungsfaktoren, so ist

$$\vec{c} = \left(\mathbf{E}\cdot\mathbf{E}^T\cdot(\vec{v} - \vec{\mu}_C)\right) + \vec{\mu}_C \tag{6.3.8}$$

der Vektor der adaptierten Gewichtungsfaktoren. Können aus den Adaptionsdaten nicht alle Komponenten des Vektors $\vec{v}$ geschätzt werden, weil zu wenig Beobachtungen einzelner Modelle existieren, so müssen vor der Anwendung von Gl. (6.3.8) die fehlenden Komponenten durch diejenigen des sprecherunabhängigen Modells ersetzt werden. Details zur Implementierung der Eigenvoice-Adaption, insbesondere für hybride Modelle finden sich in [Sommer, 2004].
Im Falle von Gauß-HMM existiert das MLED[7]-Verfahren [Kuhn u. a., 1999], das iterativ die Gewichte der Eigenvoices durch Maximierung der Produktionswahrscheinlichkeit auf den Adaptionsdaten bestimmt. Eine Projektion, mit den damit verbundenen Nachteilen, ist hierbei nicht notwendig. Eine mathematische Überleitung einer verallgemeinerten Projektion zum MLED-Verfahren findet sich in [Westwood, 1999]. [Botterweck, 2000] zeigt den erfolgreichen Einsatz des MLED-Algorithmus zur Adaption von kontextabhängigen Gauß-Modellen bei Sprachdaten mit großem Vokabular. Da die beschriebenen Verfahren explizit nur für Gaußfunktionen gelten und deren Eigenschaften zur Lösung der Gleichungen maßgeblich sind, ist bisher keine Entsprechung des MLED-Verfahrens für hybride akustische Modelle bekannt.

6.4. Ergebnisse der Adaption

Die vorgestellten Techniken sind auf den Sprecheradaptionsaufgaben der WSJ-Datenbank (s. Anhang B.2) evaluiert worden. Verwendet wurde der S3-Test einmal mit Muttersprachlern (S3-C2) und einmal mit Nicht-Muttersprachlern (S3-P0). Beide Tests umfassen jeweils 10 Sprecher, von denen jeweils 40 gelesene Sätze zur Adaption

[6] aufgrund der Eigenschaften der Hauptachsentransformation reichen meist einige wenige Eigenvektoren aus
[7] MLED - *engl*: Maximum Likelihood Eigenvoice Decomposition

der Modelle zur Verfügung stehen. Die Wortfehlerrate wird dann auf neuen Testsätzen des gleichen Sprechers ermittelt. Die Anzahl der Testsätze pro Sprecher variiert und ist in Tabelle 6.1 angegeben. Alle Adaptionsexperimente sind mit bekannter Transkription durchgeführt worden (überwachte Adaption). Der in der Praxis relevantere Fall der unüberwachten Adaption kann hieraus abgeleitet werden, indem die Ausgabe des Erkenners (bewertet durch geeignete Konfidenzmaße) als bekannte Transkription angenommen wird. Die folgenden Tabellen 6.2 bis 6.9 geben zur besseren Übersicht an dieser Stelle nur die Mittelwerte und Standardabweichungen, ermittelt aus den Wortfehlerraten der jeweils 10 Sprecher, an. Detaillierte Ergebnisse für die einzelnen Sprecher sind im Anhang F.1 zu finden.

Sprecher S3-C2	**Testsätze**	**Sprecher S3-P0**	**Testsätze**
4OA	22	4ND	42
4OB	21	4NE	42
4OC	23	4NF	41
4OD	22	4NH	42
4OE	23	4NI	41
4OF	20	4NJ	42
4OG	20	4NK	40
4OH	21	4NL	43
4OI	22	4NM	41
4OJ	21	4NN	42

Tabelle 6.1.: Übersicht über die Testsprecher und die Menge an Testsätzen in den Adaptionstestsets

6.4.1. Adaption des neuronalen Netzes

Die Tabellen 6.2 und 6.3 zeigen die Ergebnisse der Adaption des neuronalen Netzes (s. Abschnitt 6.2). Das sprecherunabhängige Basiserkennungssystem (*SI*) benutzt das hybride akustische Modell aus Abschnitt 5 mit dem in Tabelle 5.2 diskutierten MLP273-1000-47 bzw. mit dem in 5.4 diskutierten RNN39-400-139. Die angegebene (mittlere) Anzahl an adaptierten Neuronen ist in mehreren Iterationen mit dem Kriterium "maximale Varianz" Gl. (6.2.1) ermittelt und auf den Adaptionstrainingsdaten nachtrainiert worden. Eine Kreuzvalidierung (Kriterium: Fenster-Fehlerrate) auf 25% der Adaptionstrainingsdaten hat das beste Netz aus dem Training für den Test herausgesucht. Als Maß für die Signifikanz der Verbesserung durch die Adaption ist wieder die Wahrscheinlichkeit für eine systematische Verbesserung PrV (s. Abschnitt 5.5) angegeben, hier immer bezogen auf die sprecherunabhängigen WFR angegeben in der dritten Spalte.

Insgesamt erlaubt die Adaption des NN eine signifikante Senkung der WFR unabhängig vom Netztyp. Allerdings sind beim MLP gegenüber dem RNN sowohl die Fehlerrate des Basissystems niedriger, als auch der Gewinn durch die Adaption größer.

Sprecher	**WFR MLP adapt.**	**WFR SI**	**Δ rel.**	**#adaptierte Neuronen**	PrV
Durchschnitt 4OA-4OJ	13,85%	15,31%	-9,5%	186,6	100%
Standardabweichung	(8,54%)	(10,34%)			
Durchschnitt 4ND-4NN	23,58%	30,07%	-22,0%	172	100%
Standardabweichung	(7,84%)	(8,15%)			

Tabelle 6.2.: WSJ S3-C2 und S3-P0 Adaptionsergebnis (Wortfehlerrate) nach Adaption des MLP (Neuronenauswahl mit $\eta = 0,25$ nach Gl. (6.2.1))

Sprecher	**WFR RNN adapt.**	**WFR SI**	**Δ rel.**	**#adaptierte Neuronen**	PrV
Durchschnitt 4OA-4OJ	16,26%	16,95%	-4,0%	110,5	64,1%
Standardabweichung	(10,23%)	(10,70%)			
Durchschnitt 4ND-4NN	29,66%	34,88%	-15,0%	107,4	100%
Standardabweichung	(8,44%)	(9,4%)			

Tabelle 6.3.: WSJ S3-C2 und S3-P0 Adaptionsergebnis (Wortfehlerrate) nach Adaption des RNN (Neuronenauswahl mit $\eta = 0,3$ nach Gl. (6.2.1))

Bei den Nicht-Muttersprachlern (4NX) ist der Gewinn erwartungsgemäß höher, da die Abweichung zu den Sprechern des sprecherunabhängigen Trainingssets deutlich größer ausfällt und damit die Netzadaption einen größeren Effekt hat.

6.4.2. Adaption der HMM-Gewichte

Neben dem NN lassen sich bei hybriden akustischen TP-Modellen auch die Gewichtungskoeffizienten c_{ij} der HMM adaptieren. Die Tabelle 6.4 zeigt die Adaption durch Gradientenanstieg nach Abschnitt 6.3.1, jeweils wieder für die Muttersprachler (4OX) und die Nicht-Muttersprachler (4NX). Das NN ist wieder das MLP273-1000-47 (s. Tabelle 5.2). Adaptiert werden die HMM durch Maximierung der Wahrscheinlichkeitsdichte nach Gl. (6.3.5) über 15 Iterationen mit Kreuzvalidierung auf 25% der Trainingsdaten.

Das Ergebnis für den Test S3-C2 zeigt hier im Mittel eine Verschlechterung gegenüber dem nicht-adaptierten SI-Modell. Wie Tabelle F.3 zeigt, können aber dennoch die WFR einzelner Sprecher deutlich gesenkt werden. Im S3-P0 zeigte sich wiederum eine deutliche mittlere Verbesserung.
Weitere Verfahren zur Adaption der HMM Parameter sind in den Abschnitten 6.3.2 und

Sprecher	**WFR HMM adapt.**	**WFR SI**	**Δ rel.**	PrV
Durchschnitt 4OA-4OJ	15,48%	15,31%	+1,1%	–
Standardabweichung	(10,61%)	(10,34%)		
Durchschnitt 4ND-4NN	26,06%	30,07%	-13,0%	96,1%
Standardabweichung	(7,76%)	(8,14%)		

Tabelle 6.4.: WSJ S3-C2 und S3-P0 Adaptionsergebnis (Wortfehlerrate) nach Adaption der HMM durch Gradientenmaximierung

6.3.3 beschrieben, die Evaluierung dieser Verfahren zeigen die Tabellen 6.5 und 6.9. Die MAP-Schätzung der HMM-Gewichtungskoeffizienten c_{ij} wird iterativ über 5 Iterationen durchgeführt und das beste Modell durch Ermittlung der Phonemerkennungsrate auf den Validierungsdaten (25% des Adaptionstrainingssets) bestimmt. Bei der Adaption mit Eigenvoices sind keine Iterationen notwendig. Sind die Eigenvoices bereits bestimmt, so können die adaptierten Gewichte c_{ij} in einem Schritt durch Projektion in den Raum der Eigenvoices bestimmt werden.

Sprecher	**WFR HMM adapt.**	**WFR SI**	**Δ rel.**	PrV
Durchschnitt 4OA-4OJ	15,02%	15,31%	-1,9%	71,2%
Standardabweichung	(9,94%)	(10,34%)		
Durchschnitt 4ND-4NN	19,81%	30,07%	-34,0%	100%
Standardabweichung	(7,12%)	(8,14%)		

Tabelle 6.5.: WSJ S3-C2 und S3-P0 Adaptionsergebnis (Wortfehlerrate) nach MAP-Adaption der HMM

Es ergibt sich bei der MAP-Adaption eine sehr kleine Verbesserung bei den S3-C2-Sprechern (4OA-4OJ), sowie eine deutliche, signifikante Verbesserung bei den Nicht-Muttersprachlern (4ND-4NN).

Die Muttersprachler zeigen wenige systematische Ausspracheänderungen gegenüber Sprechern aus dem SI-Trainingsset. was beim Adaptieren der HMM-Parameter nur wenig Veränderung der Parameter bewirkt, bzw. bei der Adaption mit Gradientenanstieg und bei der Eigenvoice-Adaption sogar die Generalisierung der Modelle verschlechtert. Im Gegensatz dazu ist ein Grund für das generell bessere Ansprechen der HMM-Adaptionsverfahren auf dem S3-P0-Test auf die veränderte Aussprache der Nicht-Muttersprachler zurückzuführen. Eine geänderte Phonemisierung der Wörter kann durch Änderung der HMM-Mixturgewichte kompensiert werden, während ein neuer Mutter-

Sprecher	**WFR HMM adapt.**	**WFR SI**	**Δ rel.**	PrV
Durchschnitt 4OA-4OJ	16,11%	15,31%	+6,0%	–
Standardabweichung	(10,94%)	(10,34%)		
Durchschnitt 4ND-4NN	25,37%	30,07%	-16,0%	100%
Standardabweichung	(7,83%)	(8,14%)		

Tabelle 6.6.: WSJ S3-C2 und S3-P0 Adaptionsergebnis (Wortfehlerrate) nach Eigenvoice-Adaption der HMM

sprachler mit annähernd unveränderter Aussprache, aber anderem Sprechtempo, anderer Betonung oder Stimmlage nur schlecht oder gar nicht von einer HMM-Adaption profitiert.

6.4.3. Adaption des neuronalen Netzes und der HMM-Gewichte

Das Verfahren zur Adaption des NNs (Abschn. 6.2) und die Verfahren zur Adaption der HMM (Abschn. 6.3) adaptieren verschiedene Größen und können somit unabhängig voneinander eingesetzt werden. Das bedeutet, daß auch eine Adaption des NN und der HMM kombinierbar ist, indem zunächst das NN adaptiert wird und anschließend die HMM-Gewichtungskoeffizienten unter Benutzung des adaptierten Netzes neu eingestellt werden. Die Ergebnisse der kombinierten Adaption zeigen die Tabellen 6.7, 6.8 und 6.9. Die Parameter der Adaptionsverfahren sind gegenüber der Einzelevaluation unverändert. Aus Gründen der Übersichtlichkeit sind hier nur Ergebnisse für das MLP (Tabelle 6.2) angegeben, da bei der Einzeladaption des MLP gegenüber der Adaption des RNN der größere Gewinn erzielt worden ist.

Generell läßt sich eine Verringerung der Standardabweichung der WFR der einzelnen Sprecher als weiterer Hinweis auf eine systematische Verbesserung der adaptierten akustischen Modelle deuten. Bei NN-Adaption und HMM-Gradientenmaximierung ergibt sich für alle Sprecher 4OX und 4NX eine zusätzliche Verbesserung gegenüber der Einzeladaption. Das Adaptieren des NNs bewirkt eine Änderung im Verhalten der Ausgangsschicht, auf die wiederum die HMM-Parameter mit weiterem Qualitätsgewinn angepaßt werden können. Die Verwendung der MAP-Adaption nach der NN-Adaption führt zu einer Verschlechterung der Erkennung der Muttersprachler (4OX) im Gegensatz zur reinen NN-Adaption wie der Vergleich der Tabellen 6.2 und 6.8 zeigt. Bei den Nicht-Muttersprachlern (4NX) kann hier jedoch das beste Ergebnis aller Adaptionsverfahren erreicht werden.
Zusammenfassend führt eine Adaption des NN generell zu einem besseren Modell, eine zusätzliche Adaption der HMM bringt, je nach den Eigenschaften der zu adaptierenden Sprecher, weitere Vorteile.

Sprecher	**WFR MLP-HMM adapt.**	**WFR SI**	**Δ rel.**	PrV
Durchschnitt 4OA-4OJ	13,50%	15,31%	-11,8%	88,8%
Standardabweichung	(8,51%)	(10,34%)		
Durchschnitt 4ND-4NN	20,91%	30,07%	-30,0%	100%
Standardabweichung	(6,65%)	(8,14%)		

Tabelle 6.7.: WSJ S3-C2 und S3-P0 Adaptionsergebnis (Wortfehlerrate) nach Adaption des NN und Adaption der HMM durch Gradientenmaximierung

Sprecher	**WFR MLP-HMM adapt.**	**WFR SI**	**Δ rel.**	PrV
Durchschnitt 4OA-4OJ	13,97%	15,31%	-8,8%	83,0%
Standardabweichung	(8,32%)	(10,34%)		
Durchschnitt 4ND-4NN	18,47%	30,07%	-39,0%	100%
Standardabweichung	(6,72%)	(8,14%)		

Tabelle 6.8.: WSJ S3-C2 und S3-P0 Adaptionsergebnis (Wortfehlerrate) nach Adaption des NN und MAP-Adaption der HMM

Sprecher	**WFR MLP-HMM adapt.**	**WFR SI**	**Δ rel.**	PrV
Durchschnitt 4OA-4OJ	14,81%	15,31%	-3,3%	88,2%
Standardabweichung	(8,95%)	(10,34%)		
Durchschnitt 4ND-4NN	21,40%	30,07%	-29,0%	100%
Standardabweichung	(7,77%)	(8,14%)		

Tabelle 6.9.: WSJ S3-C2 und S3-P0 Adaptionsergebnis (Wortfehlerrate) nach Adaption des NN und Eigenvoice-Adaption der HMM

7. Verteilte Spracherkenner

Die vergangenen Kapitel haben die theoretischen Grundlagen verschiedener Möglichkeiten der hybriden akustischen Modellierung und ihre allgemeinen Vor- und Nachteile in verschiedenen Experimenten aufgezeigt.
Im Folgenden wird ein Anwendungsbeispiel vorgestellt, in dem die hybriden Modelle mit verbundenen Auftrittswahrscheinlichkeiten (TP) besondere Vorzüge gegenüber Standard-Gauß-Modellen haben: die verteilte Spracherkennung (DSR[1]).

7.1. Einsatzgebiete verteilter Spracherkennung

DSR ist eine Möglichkeit, die Technologie der Spracherkennung auf kleine mobile Endgeräte, z.B. Mobiltelefone oder PDAs[2] zu portieren, ohne zu viel der Rechen- und Speicherkapazität des Endgerätes zu belegen. Möglich wird dies durch eine Aufteilung der Spracherkennungsaufgabe in zwei Blöcke: Der Teil, welcher große Rechenleistungs- und Speicheranforderungen hat, verbleibt auf einem großen Server, während die Vorverarbeitung auf das Endgerät (den Client) ausgelagert wird. Die Verbindung zwischen den beiden Blöcken stellt der Kanal her, z.B. die Luftschnittstelle, welcher im allgemeinen eine begrenzte Kapazität aufweist und Übertragungsfehler erzeugen kann [David u. Benkner, 1996]. Es stellt sich nun die Frage, warum überhaupt Teile der Spracherkennung auf das Endgerät ausgelagert werden. Eine (zumindest bei Mobiltelefonen) sinnvolle Alternative wäre die direkte Übertragung der Audiodaten und Durchführung der kompletten Erkennung auf dem Server. Der entscheidende Faktor zur Entscheidung ist hierbei die zur Verfügung stehende Kanalkapazität. Im Falle einer Festnetztelefonleitung ist eine Übertragung der Audiodaten einer verteilten Architektur vorzuziehen (vergl. die Ergebnisse in [Yuk u. Flanagan, 1999]). Geht man jedoch zu schmalbandigeren Kanälen über (wie z.B. der im GSM[3]-Standard definierten Luftschnittstelle), ist der Informationsverlust durch den Quellenkodierer zu groß, um eine automatische Spracherkennung mit ausreichender Genauigkeit durchzuführen. Ein Vergleich [Barras u. a., 2001] von Spracherkennungsergebnissen unter Verwendung des GSM-Sprachcoders und mit nach dem MPEG-Standard kodierter Sprache (MPEG 1, Layer 3) zeigt den deutlichen Unterschied (relative Verluste von 18% bzw. 40% im Vergleich zu unkomprimierter Sprache mit einer Fehlerrate von 31.5% und 4kHz Signalbandbreite). Die zweite mögliche Alternative wäre die Implementierung des kompletten Spracherkenners auf dem mobilen Endgerät. Die fortschreitende technologische Entwicklung macht diese Alternative durchaus attraktiv, jedoch werden

[1] DSR - *engl.*: Distributed Speech Recognition
[2] PDA - *engl.:* Personal Digital Assistant
[3] GSM - *engl.*: Global System for Mobile Communications

mittelfristig allenfalls Systeme mit mittlerer Vokabulargröße (maximal mehrere tausend Worte) realisierbar sein. Weiterhin würde ein solcher Erkenner die gesamten Ressourcen des Gerätes benötigen, weitere Anwendungen, die auf dem Erkennungsergebnis aufbauen, wären dann nicht möglich. Zuletzt wäre auch die Pflege eines solchen Systems, z.B. ein Update des Lexikons oder des Sprachmodells umständlich. Zusammenfassend ist also die Idee einer großen Basisstation, welche verschiedene Erkennungsaufträge mit großem Wortschatz verarbeiten kann, durchaus attraktiv. Eingangsdaten für ein solches System wären dann die Merkmalsvektorströme der einzelnen Anfragen, der Erkenner kann effizient mehrere domänenabhängige Sprachmodelle und Lexika verwalten und besitzt genügend Rechenleistung, um parallel die Aufträge abzuarbeiten. Auch die Fehlerrate eines solchen Systems kann in dieser Variante durch verschiedene parallel arbeitende Erkenner reduziert werden. Die in dieser Arbeit verwendete Datenrate folgt den Angaben aus [**?**] und beträgt – netto – 4,4kbit/s (Brutto: 4,8kbit/s), dies ist die Hälfte der Datenrate zur Datenübermittlung im GSM-Standard.

7.2. Aufbau eines verteilten Spracherkenners

Die Grundidee eines verteilten Spracherkenners zeigt Bild 7.2.1. Die speicher- und rechenintensiven Teile des Spracherkenners, wie das Sprachmodell und der Dekoder mit HMM verbleiben auf dem zentralen Server, während die Berechnung der Merkmale auf den Client ausgelagert wird. Über den Kanal werden dann nicht mehr die akustischen Daten selbst übertragen, sondern nur die daraus berechneten Merkmale, deren Datenmenge deutlich kleiner ist (vergl. Kapitel 2). Die eingesparte Bandbreite kann dann für andere Dienste und Anwendungen genutzt werden. In dieser Arbeit wird der Kanal verlustfrei angenommen, dies ist in der Praxis bei weitem nicht der Fall, allerdings können die Daten durch Kanalkodierung gegen Fehler geschützt werden [David u. Benkner, 1996].

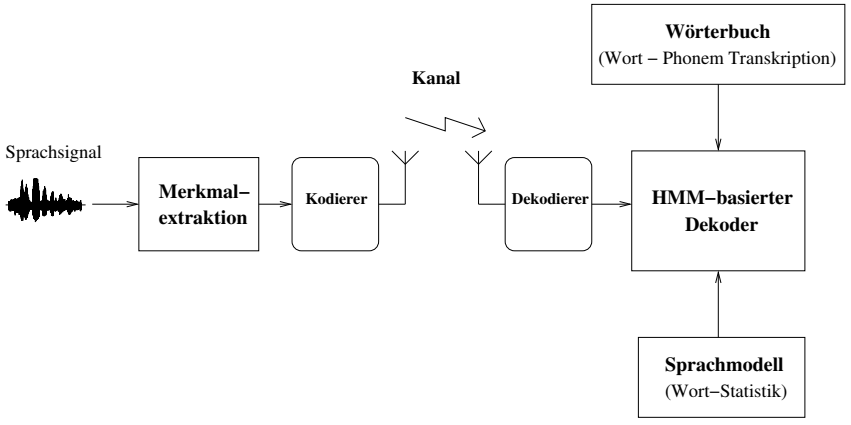


Abbildung 7.2.1.: Allgemeiner Aufbau eines verteilten Spracherkenners

7.2.1. Gauß'sche akustische Modelle

Der in Abschnitt 2.2.1 vorgestellte MFCC-Merkmalsvektor besteht aus 13 Komponenten ohne dynamische Merkmale (Cepstralkoeffizienten $c_1, \ldots, c_{12}$, Energie), im Rechner dargestellt durch 4 Bytes/Komponente (Datentyp *float*). Eine Übertragung dieses Vektors über einen verlustfreien Kanal benötigt bei einer Fensterverschiebung $T_V = 10\text{ms}$ eine Bitrate

$$\text{BR} = \frac{13 \cdot 4 \cdot 8\,\text{bit}}{10\,\text{ms}} = 41,6\,\text{kbit/s}, \tag{7.2.1}$$

die die zur Verfügung stehende Kapazität des Kanals ungefähr um das zehnfache übersteigt. Im AURORA-Standard [ETSI standard document, 2003] wird der Merkmalsvektor daher zunächst um den Cepstralkoeffizienten c_0 erweitert und anschließend mit 7 Vektorquantisierern (VQ) komprimiert. Den grundsätzlichen Aufbau des Systems zeigt Bild 7.2.2, die Kodebuchgrößen der Vektorquantisierer, sowie die Zuordnung zu den Komponenten des Merkmalsvektors zeigt Bild 7.2.3. Die VQ basieren auf dem *K-Means*-Algorithmus [Linde u. a., 1980] mit euklidischem Distanzmaß. Da über den Kanal nur

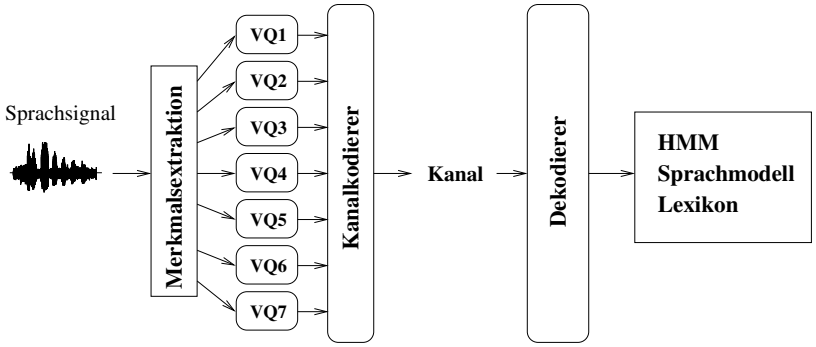


Abbildung 7.2.2.: Verteilter Spracherkenner mit Gauß'schem akustischen Modell nach [ETSI standard document, 2003]

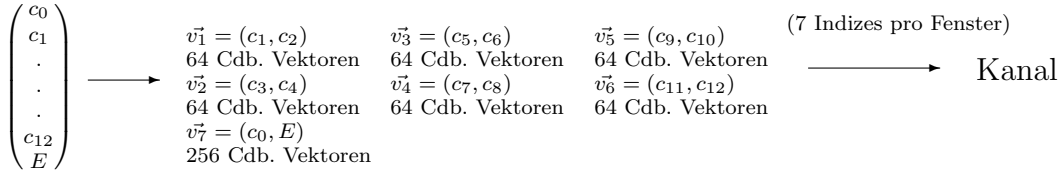


Abbildung 7.2.3.: Kodebuchanzahl und -größe für MFCC-basierte Gauß'sche Modelle

die VQ-Indizes übertragen werden, bestimmt sich die Datenmenge durch die Anzahl und Größe der Kodebücher. Beim System nach Abb. 7.2.3 ergibt sich bei gleicher Fensterverschiebung ($T_V = 10\text{ms}$) eine Bitrate von

$$\text{BR}_{7\text{VQ}} = \frac{6 \cdot 6\,\text{bit} + 8\,\text{bit}}{10\,\text{ms}} = 4,4\,\text{kbit/s} \tag{7.2.2}$$

Auf dem Server werden die übertragenen Indizes wieder durch die Prototypvektoren der VQ ersetzt. Vor der Verwendung mit HMM kann der Merkmalsvektor nun auf dem Server um dynamische Merkmale erweitert werden. Alternativ können die VQ-Indizes auch direkt für diskrete HMMs benutzt werden, dieser Ansatz führt nach [Stadermann u. a., 2001] aber zu einer erhöhten Wortfehlerrate gegenüber einem System mit kontinuierlichen Gaußdichten und wird deshalb hier nicht weiter betrachtet.

7.2.2. Hybride akustische Modelle

Gerade für die verteilte Spracherkennung bieten sich hybride akustische Modelle mit verbundenem Klassifikator (Abschnitt 5.2) als Alternative zu konventionellen Gauß'schen Modellen an. Der dann notwendige Klassifikator wird hierbei zusätzlich auf dem Client installiert. Der dadurch entstehende Nachteil einer größeren erforderlichen Rechenleistung kann durch Verwendung von kleinen Klassifikatoren mit wenigen Parametern teilweise kompensiert werden. Die über den Kanal zu übertragenden Größen sind beim

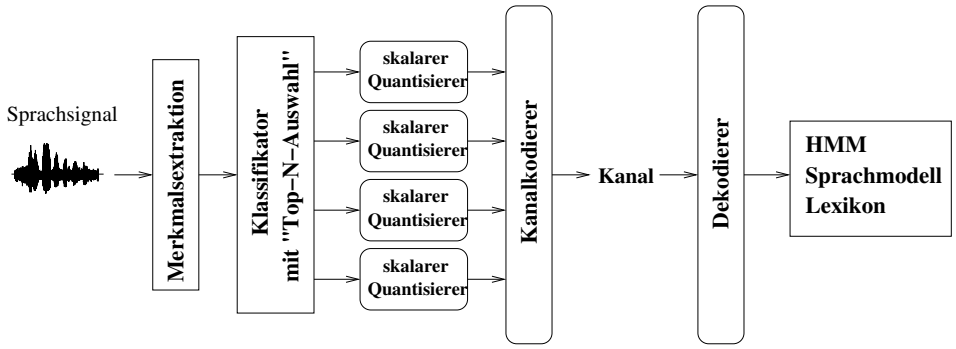
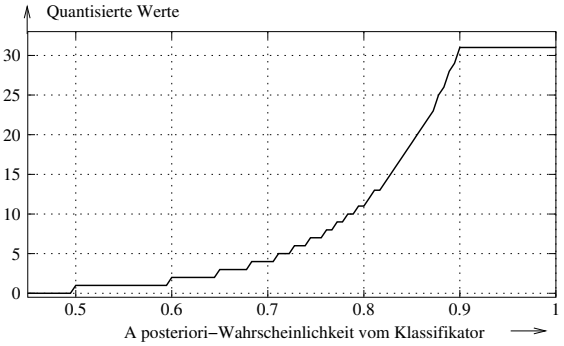


Abbildung 7.2.4.: Verteilter Spracherkenner mit hybridem akustischen Modell

hybriden akustischen Modell nicht die Merkmalsvektoren, sondern die Klassenauftrittswahrscheinlichkeiten. Bild 7.2.4 zeigt den Systemaufbau eines verteilten hybriden Erkenners. Um die geforderte Datenrate nicht zu überschreiten, werden von den J möglichen Wahrscheinlichkeiten nur die $\tilde{J}$ größten Werte übertragen und jeder dieser $\tilde{J}$ Werte mit einem skalaren Quantisierer nach Bild 7.2.5 quantisiert. b_p bezeichnet dabei die Anzahl an Bits, die für die quantisierten Werte zur Verfügung stehen, a steuert die Steilheit der Quantisierungskennlinie und die Gaußklammer [.] rundet zur nächsten Ganzzahl auf. Bei der Rekonstruktion der Wahrscheinlichkeiten auf der Server-Seite setzt man die nicht übertragenen Wahrscheinlichkeiten einfach zu Null, in Bild 7.2.6 ist dieser Prozeß illustriert. Durch die Übertragung der größten Wahrscheinlichkeitswerte tritt bei der Berechnung der Ausgabedichte (s. Gl. (5.2.1)) nur ein kleiner Fehler auf, der wenig Einfluß auf die Erkennung hat, da pro Fenster meist nur ein oder zwei Klassen hohe Auftrittswahrscheinlichkeiten erzeugen, während die übrigen Klassen sehr unwahrscheinlich sind. Der Wert der Ausgabedichte wird also durch wenige hohe Wahrscheinlichkeitswerte bestimmt, die Quantisierungskennlinie ist so gewählt, daß diese möglichst verlustfrei wiederhergestellt werden.



$$
\begin{aligned}
&\text{if } Pr(\rho_j|\vec{x}) < 0,5 \\
&\quad Q_j = 0 \\
&\text{else if } Pr(\rho_j|\vec{x}) > 0,9 \\
&\quad Q_j = 32 \\
&\text{else} \\
&Q_j = \left[(2^{b_p} - 1)\mathrm{e}^{a(Pr(\rho_j|\vec{x})-0,9)} - 0,5\right]
\end{aligned}
$$

Abbildung 7.2.5.: Quantisierungskennlinie ($a = 10$, $b_p = 5$)

Diese Annahme kann allerdings zu größeren Fehlern führen, wenn die *a priori*-Häufigkeiten der Klassen sehr ungleich verteilt ist, da dann auch kleine Klassenwahrscheinlichkeiten geteilt durch eine kleine *a priori*-Wahrscheinlichkeit einen signifikanten Beitrag leisten können, der im verteilten System dann vernachlässigt wird. Zum Erreichen der in [ETSI standard document, 2003] festgelegten Datenrate, werden die $\tilde{J} = 4$ größten Wahrscheinlichkeitswerte mit je $b_p = 5\,\text{bits}$ quantisiert und übertragen. Um auf der Serverseite den Vektor der Wahrscheinlichkeiten wieder richtig zu rekonstruieren, muß außerdem die Position im Vektor (der Index) mit übertragen werden (vergl. Bild 7.2.6). Für die Systeme aus Abschnitt 7.3.2 mit 47 bzw. 48 Klassen sind 6 Bit (max. 64 Indizes kodierbar) notwendig und die Bitrate ergibt sich zu

$$\mathrm{BR}_{\mathrm{TP}} = \frac{4 \cdot (5+6)\,\text{bits}}{10\,\text{ms}} = 4,4\,\text{kbit/s} \tag{7.2.3}$$

Ein besonderer Vorteil der hybriden akustischen Modelle nach Abschnitt 5.2 ist ihre Flexibilität. Da die Datenrate nur von der Anzahl und Quantisierung der Klassenauftrittswahrscheinlichkeiten abhängt, ist es möglich, im Client beliebige Merkmale oder Kombinationen von Merkmalen zu berechnen [Stadermann u. Rigoll, 2003c], während

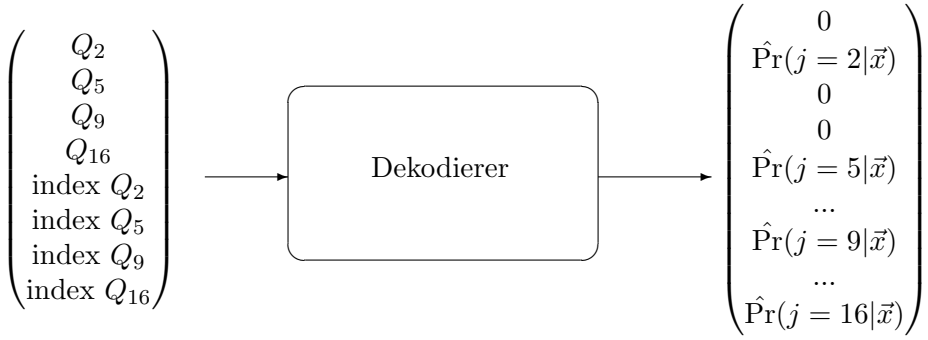


Abbildung 7.2.6.: Beispiel für die Rekonstruktion der übertragenen Wahrscheinlichkeiten ($\tilde{J} = 4$

sich die Datenmenge, die über den Kanal übertragen wird, nicht ändert. Genauso kann auf dem Server die Topologie der HMM der Aufgabe angepaßt werden (Ganzwortmodelle, Phonemmodelle), ohne das übrige System zu verändern. Auch kontextabhängige Modellierung kann ohne weitere Eingriffe in den Kanal oder den Client auf dem Server installiert werden. Zusätzlich sind auch alle weiteren im Kapitel 5 erwähnten Vorteile der hybriden Modelle in einem verteilten Erkenner nutzbar. Prinzipiell lassen sich auch hybride Modelle mit fest verbundenen Wahrscheinlichkeiten (vergl. Abschnitt 5.1) für DSR einsetzen, allerdings können dann beim oben beschriebenen Aufbau auf der Serverseite nicht mehr für alle HMM(-Zustände) Wahrscheinlichkeitsdichten berechnet werden, da nur $\tilde{J}$ Wahrscheinlichkeiten zur Verfügung stehen. Die Evaluierung (Abschnitt 7.3.2) zeigt, daß dies zu einem nicht mehr brauchbaren System führt, da der Dekoder keine vollständige, gültige Hypothese mehr durch die Testbeobachtung findet. Hybride Systeme, die auf dem TANDEM-Ansatz (Abschnitt 5.4.1) basieren, können den hier gezeigten Aufbau ebenfalls nicht verwenden, da auch hier ein vollständiger Merkmalsvektor auf der Serverseite zur sinnvollen Berechnung der Gaußdichten vorliegen muß.

7.3. Das AURORA-Projekt

Das AURORA Projekt wurde 2000 gestartet, mit dem Ziel, die Vorverarbeitung von Geräusch-gestörten Sprachsignalen zu standardisieren. Zusätzliches Ziel ist die Kompression von Merkmalen, um effektiv verteilte Spracherkennung zu betreiben. Teil dieses Projekts ist die ebenfalls im Jahr 2000 in [Hirsch u. Pearce, 2000] vorgestellte AURORA2-Datenbasis, ein ETSI-Standard zur Quantisierung und Kompression von vorverarbeiteten Merkmalen findet sich in [ETSI standard document, 2000] in der vorläufigen und in [ETSI standard document, 2003] in der endgültigen Version. Die AURORA2-Datenbasis besteht aus gesprochenen englischen Ziffernwörtern und Ziffernketten, die aus der TI-Digits-Datenbasis [Leonard, 1984] stammen. Zu diesen ungestörten Sprachdaten sind anschließend verschiedene Hintergrundgeräusche in unterschiedlichen Signal-zu-Rauschverhältnissen (SNR[4]) künstlich hinzuaddiert worden. Um die Daten noch weiter an den gewünschten Anwendungszweck – mobile Spracherkennung mit Umgebungsgeräuschen – anzupassen, ist die Abtastfrequenz auf 8kHz reduziert, sowie eine Filterung mit dem im Mobiltelefonbereich verwendeten Kanal G.712 [ITU recommendation G.712, 1996] durchgeführt worden. Ein Beispiel für ein verrauschtes Sprachsignal der AURORA2-Datenbais, sowie das zugehörige Spektrogramm finden sich in den Abbildungen 7.3.1 und 7.3.2. Zum Training von Spracherkennern existieren zwei Varianten der gleichen Sprachdaten, mit und ohne Hintergrundgeräusch. Details dazu, sowie zum Aufbau der Testdaten sind dem Anhang B.3 zu entnehmen. Der Aufbau der akustischen Modelle folgt der Beschreibung aus [Hirsch u. Pearce, 2000]: Für jedes Ziffernwort wird ein Ganzwort-HMM mit 16 Zuständen erstellt. Zwei Pausenmodelle für eine lange Pause (*sil*[5]) und eine kurze Pause (*sp*[6]) besitzen 3 bzw. 1 HMM-Zustand. Die Topologie der

[4] SNR - *engl.*: Signal-to-Noise Ratio
[5] *engl.*: silence
[6] *engl.*: short pause

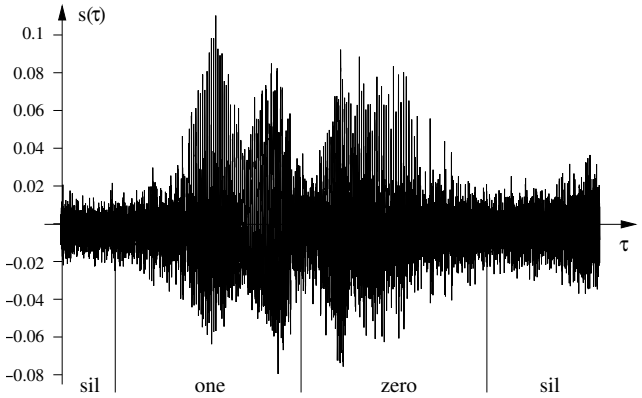


Abbildung 7.3.1.: Sprachsignal der Wörter *one zero*, SNR 5dB

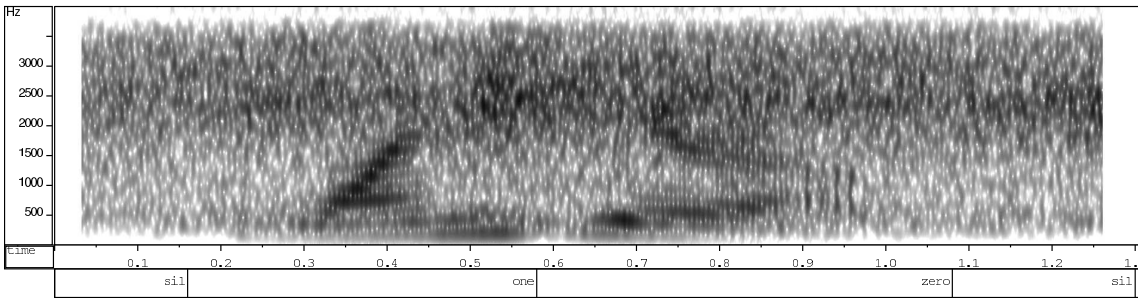


Abbildung 7.3.2.: Spektrogramm der Wörter *one zero*, SNR 5dB

Wort- und Pausenmodelle zeigt Abbildung 7.3.3. Speziell für die AURORA2-Aufgabe mit Ganzwort-HMM ist bei hybriden akustischen Modellen für das Training eine spezielle Segmentierung erstellt worden, die sich aus den Zuständen der Ganzwort-HMM ableitet. Abbildung 7.3.4 zeigt, wie für jedes der 11 Wortmodelle je 4 Zustände zu einem "Pseudo-Phonem" gruppiert werden. Zusammen mit 4 Klassen für insgesamt 4 Zustände der Pausenmodelle ergeben sich 48 Symbole, mit denen ein NN trainiert wird. Alternativ können auch phonembasierte HMM mit einem phonembasierten Klassifikator verwendet werden. Zu erwarten ist ein Qualitätsverlust gegenüber Ganzwortmodellen, jedoch sind Phonemmodelle wesentlich flexibler einsetzbar, da das mögliche Vokabular des Erkenners wesentlich größer gewählt werden kann und neue Wörter ohne Änderung des akustischen Modells durch Aufnahme der neuen Phonem-Transkription ins Wörterbuch (und einer Erweiterung des Sprachmodells) erkennbar sind. Die Erkennungsergebnisse der nachfolgenden Abschnitte sind alle mit einem Viterbi-Dekoder erzeugt worden, das Sprachmodell besteht aus einer einfachen Grammatik nach Bild 7.3.5. Zum Training der nachfolgenden Systeme sind alle 8440 Sätze des Trainingssets mit Hintergrundgeräusch

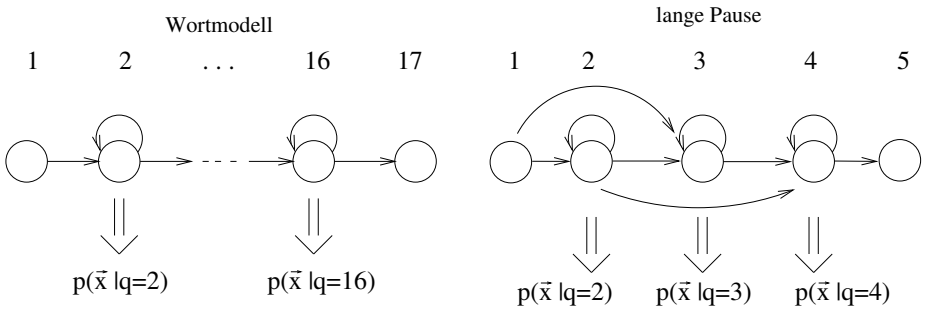


Abbildung 7.3.3.: HMM-Topologie für Ganzwort- und Pausenmodell

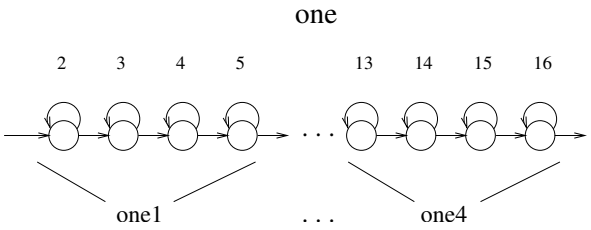


Abbildung 7.3.4.: Ableitung von Pseudo-Phonemen aus den HMM-Zuständen der Ganzwortmodelle

verwendet worden. Alle Ergebnisse in den nachfolgenden Tabellen, deren WFR niedriger als das AURORA2-Referenz-Ergebnis ist, sind hervorgehoben.

7.3.1. Ergebnisse mit Gauß'schen akustischen Modellen

Die folgenden Ergebnisse sind mit den in Kapitel 4 beschriebenen kontinuierlichen HMMs mit gaußförmigen Ausgabedichten erzeugt worden. Jeder Zustand eines Wort-HMM ist mit 3 Gaußmixturen modelliert, die Pausenzustände mit 6 Gaußmixturen. Bei den Phonemmodellen besitzt jeder HMM-Zustand einheitlich 5 Mixturen.
Zum Vergleich unterschiedlicher Merkmale auf gestörten (Hintergrundgeräusch und Kanalverzerrungen) akustischen Daten sind Modelle mit MFCC-Merkmalen und RASTA-PLP-Merkmalen (vergl. Kap. 2) trainiert worden. Der MFCC-Merkmalsvektor für Tabelle 7.1 (nicht-verteiltes System) besteht aus 12 Cepstralkoeffizienten, der logarithmierten Fensterenergie, sowie der 1. und 2. Zeitableitung dieser Komponenten (insgesamt 39 Komponenten). Der RASTA-PLP-Merkmalsvektor besteht aus 30 Komponenten (9 RASTA-PLP Koeffizienten, log. Energie und 1. und 2. Ableitung), bezeichnet als RASTA30.
Erwartungsgemäß schneiden Systeme basierend auf Phonem-Modellen schlechter ab, da hier durch die Verwendung gleicher Phoneme in unterschiedlichen Wörtern (z.B. o*ne* - ni*ne*) eine wesentlich größere Verwechslungsgefahr besteht. Der RASTA-Merkmalsvektor kann hier seine Stärke nur bei Kanalveränderungen (Test C) ausspielen, bei den Tests mit gleichem Kanal ergibt sich eine leichte Verschlechterung gegenüber dem Referenzer-

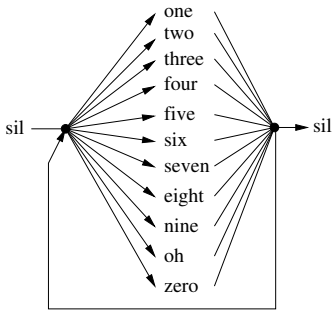


Abbildung 7.3.5.: Sprachmodell für das AURORA2-Szenario

System	Test A	Test B	Test C	Durchschnitt
AURORA2 Referenz-HMM	12,18%	13,73%	16,22%	13,61%
MFCC Ganzwort-HMM	**11,63%**	14,07%	16,55%	**13,59%**
RASTA30 Ganzwort-HMM	13,44%	14,79%	14,73%	14,24%
MFCC Phonem-HMM	16,69%	25,23%	21,94%	21,16%
RASTA30 Phonem-HMM	18,83%	20,96%	21,18%	19,94%

Tabelle 7.1.: WFR verschiedener Gauß-Systeme auf dem AURORA2-Testset

gebnis (*AURORA2 Referenz-HMM*) aus [Hirsch u. Pearce, 2000].
In Tabelle 7.2 sind die Ergebnisse für verteilte Spracherkenner wiedergegeben. Der Erkenner benutzt die Vektorquantisierer aus Abbildung 7.2.3, um die Merkmale mit einer Datenrate von 4,4 kbit/s vom Client zum Server zu übertragen, die übrigen Parameter entsprechen dem System aus Tabelle 7.1. Dynamische Merkmale (vergl. Abschnitt 2.3) werden auf der Serverseite hinzugefügt. Die Auslegung und die Anordnung der VQ sind für einen MFCC-Merkmalsvektor ausgelegt, so daß hier nur MFCC untersucht werden können. Während das System mit Ganzwort-HMM deutlich schlechter gegenüber

System	Test A	Test B	Test C	Durchschnitt
MFCC Ganzwort-HMM	13,31%	15,56%	18,02%	15,15%
MFCC Phonem-HMM	16,74%	22,63%	23,36%	20,42%

Tabelle 7.2.: WFR verschiedener verteilter Gauß-Systeme auf dem AURORA2-Testset, Bitrate 4,4 kbit/s

nicht quantisierten Merkmalen wird, kann sich das Phonem-System interessanterweise verbessern. Eine Erklärung dieses Phänomens fällt hier sehr schwer da beim Vergleich der Tabellen 7.3 und 7.5 das verteilte Ganzwort-Modell *MFCC Ganzwort-HMM TP* besser als das nicht verteilte System abschneidet, während das Phonem-Modell *MFCC Phonem-HMM TP* sich verschlechtert. Offensichtlich enthält der MFCC-Merkmalsvektor

irrelevante Informationen, die unter bestimmten Bedingungen durch die Quantisierung verschwinden.

7.3.2. Ergebnisse mit hybriden akustischen Modellen

In [Stadermann u. Rigoll, 2001, 2003c, 2005b] finden sich bereits einige Ergebnisse mit den hybriden akustischen TP-Modellen nach Abschnitt 7.2.2 auf der AURORA2-Datenbasis. Speziell ist in [Stadermann u. Rigoll, 2003c] die Kombination von MFCC und RASTA-PLP Merkmalen im Client untersucht worden, während in [Stadermann u. Rigoll, 2005b] phonembasierte hybride Modelle präsentiert worden sind. Tabelle 7.3 zeigt verschiedene hybride TP-Systeme mit einem MLP als Klassifikator auf den AURORA2-Tests im Vergleich zu hybriden Systemen mit fest verbundenen Auftrittswahrscheinlichkeiten und dem TANDEM-Ansatz. Die Modelle des TANDEM-Ansatzes besitzen die gleiche Anzahl Mixturen, wie die entsprechenden Gaußmodelle aus Tabelle 7.1. Für einen RNN-Klassifikator sind die Ergebnisse in Tabelle 7.4 wiedergegeben. Details zu den verwendeten Netzen finden sich in Tabelle 3.5.

System	**Modell**	**Test A**	**Test B**	**Test C**	**Durchschn.**
AURORA2 Referenz-HMM	–	12,18%	13,73%	16,22%	13,61%
MFCC Ganzwort-HMM	fest	**9,21%**	19,21%	22,45%	15,86%
RASTA30 Ganzwort-HMM	fest	**9,24%**	**13,27%**	**11,22%**	**11,25%**
MFCC Ganzwort-HMM	TANDEM	**9,59%**	14,04%	17,82%	**13,02%**
RASTA30 Ganzwort-HMM	TANDEM	**11,25%**	**13,36%**	**12,91%**	**12,42%**
MFCC Ganzwort-HMM	TP	**8,96%**	19,40%	22,45%	15,83%
RASTA30 Ganzwort-HMM	TP	**9,29%**	**12,92%**	**11,27%**	**11,14%**
MFCC Phonem-HMM	fest	13,53%	25,16%	28,00%	21,07%
RASTA30 Phonem-HMM	fest	13,32%	19,84%	15,67%	16,40%
MFCC Phonem-HMM	TANDEM	12,93%	25,99%	22,28%	19,66%
RASTA Phonem-HMM	TANDEM	13,79%	17,34%	13,94%	15,01%
MFCC Phonem-HMM	TP	12,98%	24,61%	26,70%	20,38%
RASTA30 Phonem-HMM	TP	12,75%	18,86%	15,11%	15,66%

Tabelle 7.3.: WFR verschiedener hybrider MLP/HMM-Systeme auf dem AURORA2 Testset

Systeme mit dem RASTA-Merkmalsvektor ergeben im hybriden System die niedrigsten WFR. Auch hier zeigt die TP-Modellierung einen Gewinn gegenüber einer starren NN/HMM-Verknüpfung nach Abschnitt 5.1. Gleichauf und teilweise besser als die anderen Ansätze verhält sich hier das TANDEM-System, das auch mit MFCC-Merkmalen besser als das originale AURORA2-Referenzsystem ist. Andere Vergleichsergebnisse mit dem TANDEM-Ansatz auf der AURORA2-Datenbasis finden sich in [Ellis u. Gomez, 2001]. Beim Vergleich der Tests A, B und C, fällt auf, daß die NN bei unbekanntem Rauschen insbesondere bei MFCC-Merkmalen schlechter als die Referenz abschneiden.

System	Modell	Test A	Test B	Test C	Durchschn.
MFCC Ganzwort-HMM	TP	13,95%	20,44%	25,72%	18,90%
RASTA30 Ganzwort-HMM	TP	13,25%	17,82%	14,81%	15,36%
MFCC Phonem-HMM	TP	14,00%	21,72%	25,70%	19,43%
RASTA30 Phonem-HMM	TP	15,00%	20,57%	17,83%	17,79%

Tabelle 7.4.: WFR hybrider RNN/HMM-Systeme auf dem AURORA2-Testset

Begründet ist dies in der Tatsache, daß die unbekannten Geräusche nicht gelernt worden sind. Eine Gaußdichte kann in einem solchen Fall dank gröberer Modellierung besser verallgemeinern. Bei RASTA-Merkmalen fällt dieser Nachteil durch die bessere Geräuschunterdrückung der Merkmale nicht mehr ins Gewicht.
Abschließend läßt sich festhalten, daß die RNN bei fast allen Tests schlechter als die MLP abschneiden, obwohl die Auswertung der FFR (s. Tabelle 3.5) das Gegenteil vermuten läßt. Als Ursache für dieses Verhalten kann eine schlechtere Generalisierung der RNN in Frage kommen, da insbesondere bei fallendem SNR der Abstand zu den MLPs größer wird. Durch das Mitlernen der Reihenfolge der Trainingsmuster auch beim Hintergrundgeräusch wären hier mehr Trainingsdaten nötig, um die Generalisierung eines MLP zu erreichen. In den Tabellen 7.5 und 7.6 ist schließlich die Auswertung verteilter

System	Modell	Test A	Test B	Test C	Durchschn.
MFCC Ganzwort-HMM	fest	97,32%	99,61%	99,57%	98,69%
RASTA30 Ganzwort-HMM	fest	83,09%	95,29%	90,88%	89,53%
MFCC Ganzwort-HMM	TP	**9,34%**	16,87%	20,61%	14,61%
RASTA30 Ganzwort-HMM	TP	**9,80%**	**12,89%**	**11,96%**	**11,47%**
MFCC Phonem-HMM	fest	78,62%	96,11%	96,16%	89,12%
RASTA30 Phonem-HMM	fest	49,98%	63,25%	56,15%	56,52%
MFCC Phonem-HMM	TP	14,26%	25,87%	27,41%	21,53%
RASTA30 Phonem-HMM	TP	13,08%	17,42%	16,28%	15,46%

Tabelle 7.5.: WFR verschiedener verteilter MLP/HMM-TP-Systeme auf dem AURORA2-Testset, Bitrate 4.4 kbit/s

Systeme mit hybriden akustischen Modellen dargestellt. Die Systeme sind, abgesehen von der Quantisierung, identisch zu denen aus den Tabellen 7.3 und 7.4.
Unter allen hybriden Modellen stellt sich sowohl bei verteilten, als auch bei nicht verteilten Systemen das *RASTA30 Ganzwort-HMM TP* als das jeweils beste heraus. Die bessere Generalisierung von MLP gegenüber RNN setzt sich auch bei verteilten Erkennern fort.
Da bei einem verteilten Erkenner nach Abschnitt 7.2.2 nicht alle Auftrittswahrscheinlichkeiten über den Kanal übertragen werden, führt die Berechnung der TANDEM-

System	Modell	Test A	Test B	Test C	Durchschn.
MFCC Ganzwort-HMM	TP	13,37%	20,27%	24,28%	18,31%
RASTA30 Ganzwort-HMM	TP	13,40%	17,77%	14,89%	15,44%
MFCC Phonem-HMM	TP	14,67%	22,79%	26,41%	20,27%
RASTA30 Phonem-HMM	TP	16,01%	22,03%	18,46%	18,91%

Tabelle 7.6.: WFR verteilter hybrider RNN/HMM-TP-Systeme auf dem AURORA2-Testset, Bitrate 4.4 kbit/s

Merkmale zu numerischen Problemen, weshalb ein verteiltes TANDEM-Modell nicht ohne spezielle Maßnahmen realisierbar ist und hier nicht untersucht wird. Bei einer festen Verknüpfung zwischen NN und HMM haben die HMM-Zustände für die keine Wahrscheinlichkeiten übertragen worden sind, eine Emissionsdichte von 0. Wie Tabelle 7.5 zeigt, führt dies zu katastrophalen Erkennungsergebnissen, da bei vielen Testsätzen keine Hypothese des Dekoders bis zum Ende der Beobachtung bestehen bleibt. Bei hybriden TP-Modellen existiert dieses Problem nicht, da die Emissionsdichten nach Gl. (5.2.1) berechnet werden und somit in jedem Fall gültige Werte annehmen.
In [Stadermann u. Rigoll, 2003c] ist zusätzlich gezeigt worden, daß sich der Eingangsvektor im verteilten hybriden TP-System prinzipiell aus beliebigen Merkmalen zusammensetzen kann (in dem Fall eine Kombination aus MFCC und RASTA), je nach Einsatzort des Erkenners kann sich daraus ein Vorteil ergeben.
Einzige Bedingung zur Auslegung des Klassifikators ist, daß die Schnittstelle (die Anzahl und Bedeutung der NN-Ausgänge) zur Kanalübertragung nicht verändert wird. Auf der Serverseite können auch im verteilten System alle schon beschriebenen Vorteile der hybriden TP-Systeme eingesetzt werden (beliebige HMM-Topologie, kontextabhängige Modellierung, vergl. Abschnitt 7.4).

7.4. Experimente mit größerem Vokabular

Um den Einfluß der Quantisierung in einem verteilten Spracherkenner weiter zu untersuchen, sind die in den Kapiteln 4.8 und 5.5 beschriebenen Systeme in einer verteilten Umgebung betrieben worden. Zur Auswertung wird hier nun der 5000 Worte umfassende Test *si-05* der WSJ0-Datenbank verwendet. Auf dem Client befinden sich wiederum die Merkmalsextraktion, ein Quellen- und Kanalkodierer und (im Falle eines hybriden Systems zusätzlich) ein Klassifikator [Stadermann u. Rigoll, 2003b]. Die Datenrate, die über den Kanal übertragen wird, sowie die Quantisierungsstrategien sind identisch mit denen aus Abschnitt 7.3. Wie schon im Abschnitt 7.3.1 festgestellt, ist auch im Vergleich von Tabelle 7.7 und Tabelle 4.1 zu beobachten, daß das beste verteilte Gauß-Monophon-System mit MFCC-Merkmalen (Tabelle 7.7) besser ist, als das entsprechende nicht verteilte System, allerdings ist die Verbesserung nur mit einer Wahrscheinlichkeit von PrV = 57% signifikant. Das Ergebnis des besten verteilten Triphon-Systems aus Tabelle 7.7 ist nur mit PrV = 53% signifikant schlechter als das entsprechende aus Tabelle 4.1. Zwischen

System	Mixturen	WFR
Monophone kont.Gauß	10	15,28%
Monophone kont.Gauß	12	14,78%
Triphone, kont.Gauß, TBC	6	12,54%
Triphone, kont.Gauß, TBC	12	13,88%

Tabelle 7.7.: WFR verteilter Gauß-Systeme mit dem *si-05*-Testset, Bigramm-Sprachmodell, Viterbi-Dekoder, Bitrate: 4,4 kbit/s

verteilten und nicht verteilten Gauß-Systemen ergibt sich also kein signifikanter Unterschied.
Tabelle 7.8 zeigt die WFR mit verteilten hybriden TP-Modellen auf den WSJ-Testdaten. Zur genaueren Untersuchung sind noch einmal 2 Fälle unterschieden: Ein Set von HMM ist auf der Serverseite unter Verwendung des quantisierten Netzes trainiert worden (Zusatz *quant*), das andere HMM-Set ist nicht an den verteilten Erkenner angepaßt, sondern unverändert aus den Systemen aus Tabelle 5.2 übernommen. Die verteilten Systeme sind mit einer Wahrscheinlichkeit von PrV = 90% (Mittelwert aus Mono47 und Tri10534) signifikant schlechter gegenüber denen aus Tabelle 5.2. Dennoch ist auch bei den verteilten hybriden Systemen das Monophon-TP-Modell schon deutlich besser (19% relativ, PrV = 100%) als das beste, kontextabhängige Gauß-Modell. Zusammenfassend können hybride TP-Modelle bei allen unter Abschnitt 7.3.2 erwähnten Vorteilen, auch in verteilten Spracherkennern mit größerem Vokabular ihren Qualitätsvorsprung gegenüber Gauß-Modellen halten.

MLP System	HMM-System	WFR
MLP273-1000-47	Mono47 quant	10,20%
MLP273-1000-47	Mono47	10,69%
MLP273-1000-47	Tri10534 quant	9,49%
MLP273-1000-47	Tri10534	9,02%

Tabelle 7.8.: WFR verteilter MLP/TP-HMM-Systemen mit dem *si-05*-Testset, Bigramm-Sprachmodell, Viterbi-Dekoder, Bitrate: 4,4 kbit/s

8. Fazit

8.1. Zusammenfassung

Das akustische Modell eines automatischen Spracherkenners basiert auf Methoden der statistischen Mustererkennung, um eine parametrische Repräsentation der gesprochenen Äußerung zu ermöglichen. Zusammen mit einer Vorverarbeitung zur Datenreduktion, einem Wörterbuch zur orthografisch korrekten Darstellung und einem Sprachmodell zur Eingrenzung wahrscheinlicher Worte ist das akustische Modell das Kernelement des Dekoders zur Konvertierung gesprochener Sprache in geschriebenen Text. Die vorangegangenen Kapitel beschreiben die Elemente eines Spracherkenners mit besonderem Augenmerk auf dem hybriden akustischen Modell bestehend aus einem statischen Klassifikator und Hidden-Markov-Modellen. Als Beispiele für geeignete statische Klassifikatoren werden neuronale Netze mit und ohne Rückkopplung, sowie Support-Vektor-Maschinen ausführlich vorgestellt. Diese Klassifikatoren müssen überwacht trainiert werden und sind in der Lage, Auftrittswahrscheinlichkeiten für die zu unterscheidenden Symbole zu berechnen. Zur Erkennung zeitvariabler Daten (Sprachsignale) bietet sich eine Kombination des Klassifikators mit HMM an. Von den verschiedenen vorgestellten Möglichkeiten, wie einer festen NN/HMM-Verknüpfung oder dem TANDEM-Ansatz, hat sich die Verknüpfung über verbundene, gewichtete Auftrittswahrscheinlichkeiten (TP) als die erfolgreichste Strategie herausgestellt. Die Wortfehlerrate von TP-Systemen beim sprecherunabhängigen WSJ0-Test mit einem Vokabularumfang von 5000 Worten ist signifikant niedriger als die WFR von Gauß-HMM oder anderen hybriden Ansätzen. Mit dem TP-Ansatz sind kontextabhängige Modelle und beliebige HMM-Topologien ohne Veränderung des Klassifikators möglich, die Ergebnisse belegen den zu erwartenden Gewinn dieser Erweiterungen. Beim Vergleich der neuronalen Netze zeigt das MLP die besten Ergebnisse, das RNN kommt bei etwas geringerer Leistung allerdings mit weniger Parametern aus. Zur Verbesserung des RNN sind neben der Phonem-, bzw. HMM-Zustandsklassifikation Zusatzaufgaben parallel trainiert worden. Es hat sich als vorteilhaft herausgestellt, wenn zusätzlich das Geschlecht des Sprechers klassifiziert wird, da diese Aufgabe einfach zu lernen und unabhängig von der Hauptaufgabe ist. Die Verwendung von SVM kann bisher nur bei sehr kleinem Vokabular (11 Zahlwörter) auf dem AURORA2-Test evaluiert werden, da die Klassifikation mit mehreren SVM bislang sehr viel Rechenzeit erfordert. Dennoch lassen sich einige positive Tendenzen dieses Systems besonders bezüglich der Robustheit gegenüber Geräuschen erkennen.
Neben den sprecherunabhängigen Systemen werden weiterhin Möglichkeiten präsentiert, die hybriden akustischen Modelle mit verbundenen Auftrittswahrscheinlichkeiten an einen einzelnen Sprecher zu adaptieren. Erwähnenswert ist hierbei, daß sowohl Parameter des Klassifikators (untersucht worden sind in dieser Arbeit neuronale Netze), als

auch Parameter der HMM unabhängig voneinander und mit zusätzlichem Gewinn adaptiert werden können.
Zusätzliche Anwendungsmöglichkeiten für die beschriebenen hybriden akustischen Modelle bietet die verteilte Spracherkennung, bei der die Komponenten örtlich verteilt arbeiten und über einen Kanal verbunden sind. Besonders die Flexibilität bei der Veränderung einzelner Komponenten kann mit Gauß'schen akustischen Modellen nicht erreicht werden. Dieser Vorteil ist allerdings nur mit einer gegenüber dem bisherigen Ansatz erhöhten Rechenleistung auf dem Client (wegen der Eingliederung des Klassifikators) zu erreichen. Die vorliegende Arbeit zeigt, wie Clients basierend auf hybriden TP-Modellen mit unterschiedlichen Merkmalsvektoren (MFCC und RASTA-PLP) an den gleichen Server gekoppelt werden können. Bei bekanntem Geräusch kann ein Client mit MFCC-Merkmalsextraktion mit dem AURORA2-Test das beste Ergebnis erreichen, während unter veränderten Umgebungsbedingungen ein Client mit RASTA-PLP-Merkmalen vorteilhafter ist. Auch bei der AURORA2-Datenbasis und verteilter Erkennung ist insgesamt ein deutlicher Gewinn der hybriden TP-Modelle gegenüber Gauß-HMM-Systemen zu verzeichnen.

8.2. Ausblick

Trotz der umfangreichen Betrachtungen der besprochenen Algorithmen bleiben stets offene Fragen und Vorschläge zur Leistungssteigerung. Die möglichen Verbesserungen des Klassifikators sind kaum erschöpfend zu beschreiben. Insbesondere bei SVM müssen zum Einen bessere Trainingsverfahren umgesetzt werden, die die Anzahl der Support-Vektoren minimieren. Zum Anderen ist die aus Implementierungsgründen in dieser Arbeit verwendete Kombination mehrerer SVM in der Form *eins-gegen-alle* nicht optimal. Weiterhin ist das Training von Klassifikator und HMM-Parametern zweigeteilt. Wünschenswert wäre ein einstufiger Trainingsprozeß, bei dem das Klassifikatortraining ohne eine zusätzliche Segmentierung auskommt. Da die Fensterfehlerrate als Qualitätskriterium des Klassifikators nicht immer zuverlässige Aussagen über die Qualität des gesamten Erkenners zuläßt, sind andere Kriterien denkbar, wie z.B. die in dieser Arbeit vorgestellte PFR. Eine Verbindung zwischen PFR und dem Optimierungskriterium des Klassifikatortrainings ist allerdings bisher nicht gefunden worden.
Die Methode der hybriden Modellierung von Zeitsignalen unter Benutzung einer Klassifikator/HMM-Kombination mit verbundenen Auftrittswahrscheinlichkeiten ist nicht nur für die akustische Modellierung eines Sprachsignals geeignet. Sofern eine geordnete Folge von Merkmalsvektoren gleicher Dimension erzeugt werden kann, ist dieser Ansatz z.B. auch zur Handschrifterkennung, zur Emotionserkennung oder zur Auswertung von Besprechungen einsetzbar. Die Schwierigkeit ist – neben der Erzeugung einer geeigneten Merkmalsvektorfolge – die Wahl und Auslegung des Klassifikators, sowie (in geringerem Maße) die Wahl der HMM-Topologie. Zusammenfassend bietet die in dieser Arbeit beschriebene hybride akustische Modellierung die Flexibilität und das Potenzial, sowohl als bessere Alternative zu den benannten klassischen Ansätzen zu fungieren, als auch für neue Aufgaben der Mustererkennung gerüstet zu sein.

A. Verwendete Formelzeichen und Abkürzungen

A.1. Formelzeichen

τ	Zeitvariable
$s(\tau), s'(\tau)$	Audiosignale
f	Frequenz
ω	Kreisfrequenz
Ω	transformierte Frequenz auf der Barkskala
$S(f), S(z)$	Fourier- bzw. z-Spektrum des Audiosignals $s(\tau)$
T_F	Dauer eines Fensters
$w(\tau)$	Fensterfunktion
T_V	Verschiebung zwischen zwei Fenstern
t	diskreter Zeitschritt von einem Fenster zum nächsten, $t \cdot T_V$
c_n	Cepstralkoeffizient mit Index n
c_S, c_U	Cepstralkoeffizienten des reinen Sprachsignals und des Kanals
c_X	Cepstralkoeffizient des resultierenden Signals
$\bar{c_S}, \bar{c_X}$	Mittelwerte des Sprachsignals und des resultierenden Signals
$\tilde{c_X}$	Cepstralkoeffizient des kanalbereinigten,resultierenden Signals
$\mathfrak{J}$	Parameter bei der RASTA-PLP-Berechnung
$H(z)$	Filterfunktion
e	Energie eines Fensters
$\vec{f}$	Merkmalsvektor, Einzelelement f_n
$\vec{f}_\Delta, \vec{f}_{\Delta\Delta}$	dynamische Merkmalsvektoren
N	Dimension eines Merkmalsvektors
$\vec{x}$	Eingangsvektor in den Klassifikator, zusammengesetzt aus einem oder mehreren Merkmalsvektoren, Einzelelement x_l
$\mathbf{W}$	Gewichtsmatrix eines NN zwischen Eingang und Ausgang (RNN) bzw. Eingang und versteckter Schicht (MLP)
$\mathbf{V}$	Gewichtsmatrix eines NN zwischen Rückkopplungsschicht und Ausgang (RNN) bzw. versteckter Schicht und Ausgang (MLP)
$\vec{z}$	Rückkopplungsvektor im RNN, bzw. Ausgangsvektor der versteckten Zwischenschicht im MLP, Einzelelemen: z_k
$\vec{\zeta}$	Ergebnisvektor des Produktes $\mathbf{W}^T\vec{x}$, Einzelelement ζ_k
$\vec{\xi}$	Ergebnisvektor des Produktes $\mathbf{V}^T\vec{z}$, Einzelelement ξ_i
$\vec{s}$	Support-Vektor

$\vec{y}$	Ausgangsvektor des Klassifikators mit Auftrittswahrscheinlichkeiten, Einzelelement y_j
$\vec{y}'$	Zielvektor für das Training von NN oder SVM, Einzelelement y'_j
L	Anzahl der Klassifikatoreingänge
J	Anzahl der Klassifikatorausgänge = Anzahl der zu unterscheidenden Klassen
K	Anzahl der Rückkopplungsknoten (RNN) bzw. der versteckten Knoten (MLP)
E	Optimierungsfunktion für NN bzw. SVM
ρ_j	Klassensymbol/Repräsentant für die Klasse mit dem Index j
$F_h(\zeta_k)$	Nichtlineare Funktion der Rückkopplungsschicht (RNN), bzw. der versteckten Schicht (MLP)
$F_o(\xi_j)$	Nichtlineare Funktion der Ausgangsschicht eines Klassifikators
β	Lernrate
$Pr(\rho_j)$	Wahrscheinlichkeit für das Auftreten des Symbols ρ_j
m_{nu}	einzelnes Modell als Repräsentant für ein Phonem oder ein Wort
M_n	Sequenz von Modellen m_{nu} als Repräsentant für ein Wort oder einen Satz
$\mathbf{A}$	quadratische Matrix der Übergangswahrscheinlichkeiten a_{il}
$p(x)$	Wahrscheinlichkeitsdichtefunktion der Größe x
X	zeitliche Sequenz von Merkmalsvektoren
Q	zeitliche Sequenz von HMM-Zuständen
q_i	HMM-Zustand mit Index i
c_{ij}	Gewichtungsfaktor der Mixtur j im HMM-Zustand q_i
$\vec{\kappa}_j$	Codebuchvektor eines Vektorquantisierers
$\vec{\mu}_j$	Mittelwertvektor einer Gaußfunktion, Einzelelement $\mu_{n,j}$
$\mathbf{\Sigma_j}$	Kovarianzmatrix einer Gaußfunktion
$\vec{\sigma}_j^2$	Varianzvektor der Diagonalelemente der Kovarianzmatrix einer Gaußfunktion, Einzelelement $\sigma_{n,j}^2$
$b_i(t)$	Ausgabewahrscheinlichkeitsdichte des HMM-Zustandes q_i
$b_{ij}(t)$	Mixtur mit dem Index j der Wahrscheinlichkeitsdichte $b_i(t)$
λ	Trainierbarer Parametersatz eines HMM
$\mathfrak{L}$	Likelihood, bedingte Wahrscheinlichkeitsdichte von Beobachtungen
$\alpha_i(t), \beta_i(t)$	Vorwärts- bzw. Rückwärtswahrscheinlichkeit(sdichte)
$\xi_{ij}(t)$	Wahrscheinlichkeit für das Beobachten der Zustände q_i und q_j zu den Zeitschritten $t+1$ und t bei Kenntnis einer gesamten Beobachtung
K	zeitliche Sequenz von Gaußdichten/-mixturen
$k_{q_i}(t)$	Mixtur der Wahrscheinlichkeitsdichte des HMM-Zustandes q_i zum Zeitpunkt t
$\zeta_{ij}(t)$	Verbundwahrscheinlichkeit für das Auftreten von Zustand q_i und Mixtur j unter Kenntnis einer gesamten Beobachtung

R	Anzahl der Trainingssätze im Trainingskorpus
B	Vokabulargröße des Spracherkenners
A, B	Parameter zur Einstellung der nachgeschalteten Sigmoid-Funktion zur Konvertierung des SVM-Ausgangs in eine Wahrscheinlichkeit
$\mathrm{var}(z_k)$	Varianz des Neurons z_k
η	Schwellwert für die Anzahl der zu adaptierenden Neuronen
κ_{ij}	Transformierte Gewichtungskoeffizient, Hilfsgröße zur HMM-Adaption mit Gradientenmaximierung
$q_V(t)$	HMM-Zustand zum Zeitpunkt t, ermittelt aus einer Viterbi-Segmentierung
ν_{ij}	Hilfsgröße in den MAP-Adaptionsgleichungen
$\mathbf{C}, \vec{\mu}_C$	Kovarianzmatrix $\mathbf{C}$ aller Mixturkoeffizienten aller Sprecher und entsprechender Mittelwertvektor $\vec{\mu}_C$
$\mathbf{E}$	Matrix der Eigenvoices
$\vec{v}$	Vektor mit Parametern zur Eigenvoiceadaption
BR	Bitrate
$\tilde{J}$	Anzahl an Wahrscheinlichkeitswerten, die über den Kanal übertragen werden
b_{n_p}	Anzahl in Bit mit der ein Wahrscheinlichkeitswert quantisiert wird

A.2. Abkürzungen

DSR	*engl.* Distributed Speech Recognition
GSM	*engl.* Global System for Mobile communications
HMM	Hidden-Markov-Modell(e), *engl.* Hidden Markov Model
MAP	*engl.* Maximum A Posteriori
ML	*engl.* Maximum Likelihood
MLED	*engl.* Maximum Likelihood Eigenvoice Decomposition
MLLR	*engl.* Maximum Likelihood Linear Regression
MLP	Multi-Layer-Perzeptron, *engl.* Multi Layer Perceptron
MMK	Mensch-Maschine-Kommunikation
NLU	*engl.* Natural Language Understanding
NN	Neuronale(s) Netz(e), *engl.* Neural Network
PDA	*engl.* Personal Digital Assistant
PLP	*engl.* Perceptual Linear Prediction
RNN	Rückgekoppeltes neuronales Netz, *engl.* Recurrent Neural Network
SI	*engl.* speaker independent
SNR	Signal-zu-Rausch-Verhältnis, *engl.* Signal-to-Noise Ratio
SVM	Support-Vektor-Maschine, *engl.* Support Vector Machine
TBC	*engl.* Tree-Based Clustering
TP	verbundene Auftrittswahrscheinlichkeiten, *engl.* Tied-Posteriors
VQ	Vektorquantisierer, *engl.* Vector Quantizer
WSJ	*engl.* Wallstreet Journal Datenbasis

B. Sprach-Datenbasen

B.1. Phonemvorrat

Die in dieser Arbeit verwendete Definition eines Phonemvorrates für amerikanisches Englisch entstammt dem Aussprachewörterbuch für die WSJ-Datenbasis. Dieses Wörterbuch ist vom Laboratoire d'Informatique pour la Mécanique et les Sciences de l'Ingénieur (LIMSI) für diese Datenbasis zusammengestellt worden [Gauvain u. a., 1995]. Sie unterscheidet 45 Phoneme, die in Tabelle B.1 zusammen mit einem Beispielwort (das Phonem ist jeweils unterstrichen) wiedergegeben sind. In Klammern ist die Darstellung der Phoneme in der Schreibweise des CMU-Lexikons angegeben.

Phonem	**Beispiel**	**Phonem**	**Beispiel**	**Phonem**	**Beispiel**
a (aa)	b<u>o</u>b	f (f)	<u>f</u>ine	O (oy)	b<u>oy</u>
@ (ae)	b<u>a</u>t	g (g)	<u>g</u>ore	o (ow)	b<u>oa</u>t
^ (ah)	b<u>u</u>t	h (hh)	<u>h</u>at	p (p)	<u>p</u>et
c (ao)	ab<u>oar</u>d	I (ih)	b<u>i</u>t	r (r)	<u>r</u>ent
W (aw)	b<u>ou</u>t	| (ix)	act<u>i</u>ng	s (s)	<u>s</u>at
x (ax)	<u>a</u>board	i (iy)	b<u>ee</u>t	S (sh)	<u>sh</u>in
X (ex)	ac<u>er</u>	J (jh)	<u>g</u>in	t (t)	<u>t</u>en
Y (ay)	b<u>i</u>te	k (k)	<u>c</u>ore	T (th)	<u>th</u>in
B (b)	<u>b</u>et	l (l)	<u>l</u>et	u (uw)	b<u>oo</u>t
C (ch)	<u>ch</u>in	L (el)	abl<u>e</u>	U (uh)	b<u>oo</u>k
d (d)	<u>d</u>ebt	m (m)	<u>m</u>et	v (v)	<u>v</u>at
D (dh)	<u>th</u>at	M (em)	bott<u>om</u>	w (w)	<u>w</u>et
E (eh)	b<u>e</u>t	G (ng)	sin<u>g</u>	y (y)	<u>y</u>et
R (er)	b<u>ir</u>d	n (n)	<u>n</u>et	z (z)	<u>z</u>oo
e (ey)	b<u>ai</u>t	N (en)	clinto<u>n</u>	Z (zh)	a<u>zu</u>re

Tabelle B.1.: Phonemvorrat des verwendeten LIMSI-Aussprachelexikons für amerikanisches Englisch

B.2. Die Wall-Street-Journal Datenbasis

Die *Wall-Street-Journal*-Datenbasis (WSJ), mit der die Ergebnisse der Kapitel 4 bis 7 erstellt worden sind, stammt in ihrer ersten Fassung aus dem Jahr 1992 (WSJ0). Sie ist eine vom amerikanischen Verteidigungsminsterium (DARPA) mitfinanzierte Sammlung

von gelesenen Texten aus der gleichnamigen Zeitung. Ziel des Korpus ist die Entwicklung von Systemen zur Erkennung kontinuierlich gesprochener Sprache mit großem Vokabular auf einer international vergleichbaren Grundlage.

Nach [Paul u. Baker, 1992] enthält die Version von 1992 folgende Trainings- und Testszenarios:

- *sd-3* - Trainingsdaten für ein sprecherabhängiges System mit 3 Sprechern und jeweils 2400 Sätzen pro Sprecher (insgesamt 7200 Sätze, entspricht etwa 4,8 Stunden pro Sprecher)
- *sd-12/si-12* - Trainingsdaten für ein sprecherabhängiges System mit 12 Sprechern und jeweils 600 gesprochenen Sätzen (insgesamt 7200 Sätze, entspricht ca. 1,2 Stunden/Sprecher), diese Daten können unter der zweiten Bezeichnung auch für ein sprecherunabhängiges System verwendet werden
- *si-84* - Trainingsdaten für ein sprecherunabhängiges System mit 84 Sprechern und jeweils 50 oder 100 Sätzen/Sprecher (7200 Sätze, entspricht etwa 15 Stunden für alle Sprecher)

Die Vokabulargröße der WSJ0-Trainingsdaten umfaßt 5000 Worte. Testergebnisse können auf insgesamt 6 verschiedenen Sets ermittelt werden, zusätzlich stehen zum Optimieren der trainierten Modelle 6 korrespondierende Entwicklungs-Testsets zur Verfügung. Die einzelnen Tests sind

- *sd-05* Sprecherabhängiger Test, Vokabulargröße 5000 Worte, Training auf sd-3, insgesamt 330 Sätze
- *sd-20* Sprecherabhängiger Test, Vokabulargröße 20000 Worte, Training auf sd-3, insgesamt 330 Sätze
- *sd-05a/sd-20a* Sprecherabhängiger Test, Vokabulargröße 5000 bzw. 20000 Worte, Training auf sd-12, insgesamt 330 Sätze
- *si-05/si-20* Sprecherunabhängiger Test, Vokabulargröße 5000 bzw. 20000 Worte, Training auf si-84, insgesamt 330 Sätze

Zusätzlich gehören zur Datenbasis auch Sprachmodelle: Mitgeliefert werden statistische Modelle trainiert auf 35 Millionen Worten des WSJ der Jahrgänge 1987-1989, jeweils als Bigramm und Trigramm für Vokabulargrößen von 5000, 20000 und 64000 Worten.
1993 ist eine neue Version der WSJ-Datenbasis unter der Bezeichnung WSJ1 erschienen. Bei der WSJ1 gliedern sich die Testsets in die Basistests (*Hubs*)

- H1 - Vokabulargröße 64000 Worte
- H2 - Vokabulargröße 5000 Worte

sowie in die zu den Hubs gehörenden Spezialtests (*Spokes*):

- S1 - Adaption des Sprachmodells

- S2 - Erkennung von themenfremden Nachrichten (nicht aus der WSJ)
- S3 - Sprecheradaption
- S4 - Inkrementelle Sprecheradaption
- S5 - Mikrofonunabhängigkeit
- S6 - Verwendung eines anderen (bekannten) Mikrofons
- S7 - Hintergrundgeräusche
- S8 - Definiertes Rauschen
- S9 - spontane (diktierte) Sprache

Im Rahmen dieser Arbeit wurde der Spezialtest S3 für eine Vokabulargröße von 5000 Worten (H2) verwendet (s. Abschnitt 6.4). Einzelheiten zu den WSJ1-Trainingsdaten und zu den weiteren Tests sind [Kubala u. a., 1994] zu entnehmen.

B.3. Die AURORA2-Datenbasis

Ausgehend von der *TI-Digits*-Datenbasis [Leonard, 1984] besteht die *AURORA2*-Datenbasis [Hirsch u. Pearce, 2000] aus gesprochenen englischsprachigen (amerikanisches Englisch) Ziffernketten. Die Länge einer Ziffernkette reicht dabei von 1 bis 7 Worten. Das Vokabular umfaßt die Worte *one*, *two*, ..., *nine*, *zero*, *oh*. Dem Trainingset der TI-Digits sind 8440 Äußerungen von 55 männlichen und 55 weiblichen Sprechern entnommen. Um den Anforderungen von (mobiler) Telefonsprache gerecht zu werden, ist die Abtastfrequenz von 20kHz auf 8kHz reduziert und das Sprachsignal mit einer G.712 Frequenzcharakteristik [ITU recommendation G.712, 1996] gefiltert worden. Zusätzlich existiert ein zweites Trainingsset (Multi-condition Training) mit den gleichen Sprachdaten, diesmal allerdings neben einem unbehandelten Teil mit künstlich hinzuaddierten Geräuschen bei SNRs[1] zwischen 20dB und 5dB (in 5dB Schritten). Die Geräusche sind z.B. aus einem Restaurant oder aus einer U-Bahn; eine ausführliche Darstellung aller Geräusche (inklusive Langzeitspektren) findet sich in [Hirsch u. Pearce, 2000].
Die AURORA2-Datenbasis definiert 3 Tests mit insgesamt 4004 Äußerungen von 52 männlichen und 52 weiblichen Sprechern:

- Test A
  1001 Äußerungen mit den Geräuschen aus dem Multi-condition Trainingset diesmal bei SNRs zwischen -5dB und 20dB (in Schritten von 5dB) plus einem unbehandelten Teil (*clean*). Das Testergebnis ist die mittlere WFR der Tests zwischen 0dB und 20dB.

[1] *engl.*: Signal-to-noise ratio

- Test B
  1001 Äußerungen mit neuen Geräuschen bei SNRs zwischen -5dB und 20dB (in Schritten von 5dB) plus einem unbehandelten Teil (*clean*). Das Testergebnis ist die mittlere WFR der Tests zwischen 0dB und 20dB.

- Test C
  1001 Äußerungen mit bekannten und neuen Geräuschen bei SNRs zwischen -5dB und 20dB (in Schritten von 5dB) plus einem unbehandelten Teil (*clean*). Im Gegensatz zu den anderen Tests sind hier alle Daten mit einer MIRS-Charakteristik [Hirsch u. Pearce, 2000] gefiltert worden. Das Testergebnis ist die mittlere WFR der Tests zwischen 0dB und 20dB.

Zusätzlich zu den Sprachdaten enthält die AURORA2-Datenbasis Referenzergebnisse, die mit kontinuierlichen Gauß-HMM unter HTK [Young u. a., 2000] erstellt worden sind. In den Ergebnistabellen (Abschnitt 7.3.1) sind diese Zahlen unter der Bezeichnung *AURORA2 Referenz-HMM* aufgelistet.

C. Systemaufbau

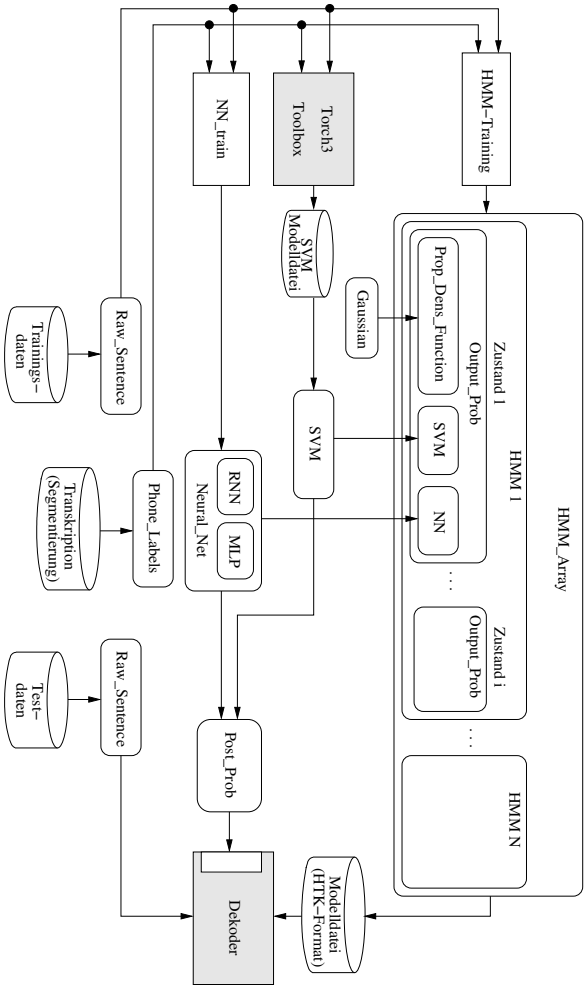


Abbildung C.0.1.: Module und Funktionen zum Training und Test akustischer Modelle

Bild C.0.1 zeigt das Zusammenspiel der wesentlichen Module und Objekte beim Training und der Erkennung akustischer Modelle, wie es im Rahmen dieser Arbeit unter Benutzung der Programmiersprache C++ realisiert worden ist. Schraffierte Boxen kennzeichnen externe Programme, Boxen mit runden Ecken bezeichnen Objekte (Klassen) zum Aufbewahren und Manipulieren von Daten bzw. Parametern. Externe Datenquellen und -senken sind durch Zylinder symbolisiert.

D. Herleitung des EM-Algorithmus für HMM

Ausgangspunkt ist die Maximierung der Größe L aus Gl 4.4.1, die sich wie folgt darstellen läßt:

$$\begin{aligned} L &= \int_U \log\left(p(X|\hat{\lambda})\right) p(U|X,\lambda) dU \\ &= \mathrm{E}\left[\log\left(p(X|\hat{\lambda})\right)|X,\lambda\right] \\ &= \underbrace{\mathrm{E}\left[\log\left(p(X,U|\hat{\lambda})\right)|X,\lambda\right]}_{\mathfrak{Q}(\hat{\lambda},\lambda)} - \underbrace{\mathrm{E}\left[\log\left(p(U|X,\hat{\lambda})\right)|X,\lambda\right]}_{\mathfrak{H}(\hat{\lambda},\lambda)} \end{aligned} \tag{D.0.1}$$

Die Größe $\mathfrak{H}(\hat{\lambda},\lambda)$ läßt sich mit Hilfe der Jensen-Ungleichung nach oben begrenzen [Schukat-Talamazzini, 1995], so daß immer gilt: $\mathfrak{H}(\hat{\lambda},\lambda) \leq \mathfrak{H}(\lambda,\lambda)$. Das bedeutet, daß für neue, verbesserte Parameter $\hat{\lambda}$ die Größe $\mathfrak{H}(.)$ stets abnimmt. Der EM-Algorithmus kann sich also auf die Maximierung der Kullback-Leibler-Statistiken $\mathfrak{Q}(\lambda,\hat{\lambda})$ beschränken, es gilt also:

$$\mathfrak{Q}(\hat{\lambda},\lambda) \geq \mathfrak{Q}(\lambda,\lambda) \Rightarrow L(\hat{\lambda}) \geq L(\lambda) \tag{D.0.2}$$

Bei Verwendung von Wahrscheinlichkeitsdichten mit mehreren Mixturen, gibt es, wie in Abschnitt 4.4 beschrieben, neben der Beobachtung X die Abfolge der Zustände Q (im Modell charakterisiert durch die Übergangswahrscheinlichkeiten $\mathbf{A}$) und die Abfolge der Mixturen K (im Modell charakterisiert durch die Ausgabedichten $b_{ij}(\vec{x})$) als nicht-beobachtbare Variablen, so daß folgender Ausdruck zu maximieren ist:

$$\mathfrak{Q}(\hat{\lambda},\lambda) = \mathrm{E}\left[\log\left(p(X,K,Q|\hat{\lambda})\right)|X,\lambda\right] \tag{D.0.3}$$

Damit ist der *Expectation*-Schritt des EM-Algorithmus vollzogen. Die HMM-Struktur ergibt mit Gl. (4.4.13) einen konkreten Ausdruck für $\log\left(p(X,K,Q|\hat{\lambda})\right)$. Bei der Bildung der einzelnen Gradienten bezüglich der gesuchten Parameter sind die Nebenbedingungen aus den Gleichungen (4.2.6) und (4.3.2) einzuhalten, so daß Lagrange-Multiplikatoren bei der Bildung der Gradienten eingeführt werden. Eine ausführliche Darstellung des *Maximization*-Schrittes findet sich in [Bilmes, 1998]. Die Lösung dieser Maximierung sind die Gleichungen (4.4.14) bis (4.4.16) Der vollständige Algorithmus führt also zunächst den E-Schritt mit frei gewählten Startwerten für die Parameter λ durch, berechnet dann im M-Schritt neue, verbesserte Parameter $\hat{\lambda}$ und wiederholt dann beide Schritte so lange, bis ein Abbruchkriterium erreicht ist.

E. Zusammenfassung von Parametern kontextabhängiger Modelle

Bei der Erstellung kontextabhängiger Modelle wird allgemein von bereits trainierten kontextunabhängigen Modellen ausgegangen, die entsprechend der Kontextabhängigkeit kopiert werden. Die Übergangsmatrizen **A** aller neuen kontextabhängigen Modelle eines ursprünglichen Monophones werden in einem ersten Schritt zusammengefaßt (s. Bild E.0.1). Zur weiteren Datenreduzierung, insbesondere bei Gauß'schen Ausgabedichten,

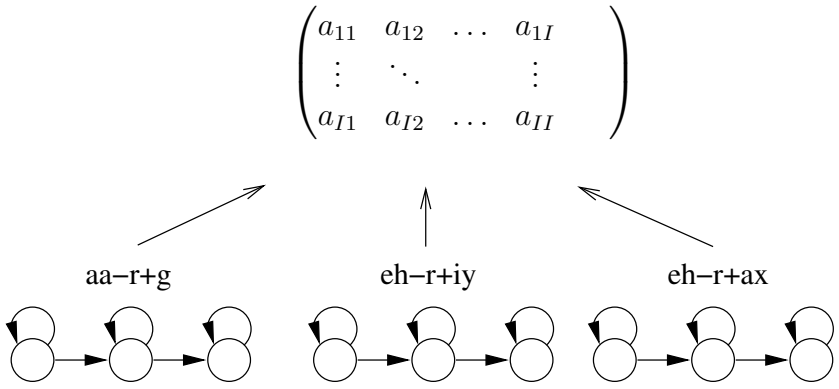


Abbildung E.0.1.: Zusammenfassung der Übergangswahrscheinlichkeiten

können einzelne Zustände unterschiedlicher Modelle zusammengefaßt werden. Hierzu kommen mathematische *Clusteringverfahren* zum Einsatz, die entweder kleine Gebiete sukzessive Zusammenfassen (Bottom-Up-Clustering) oder ein großes Gebiet schrittweise unterteilen (Top-Down-Clustering). Zum Zusammenfassen von Zuständen ist ein Abstandsmaß, wie die euklidische Distanz, erforderlich, das z.B. zwei Zustände mit kleinstem Abstand findet. Die Bestimmung des Abstandes erfolgt durch Betrachtung der Parameter der Zustandsdichten b_i, die mit den Trainingsdaten geschätzt worden sind. Abbruchkriterium für diese Art des Clusterings kann eine bestimmte Anzahl von Clustern oder ein Schwellwert für den Abstand zwischen den Clustern sein. Problematisch ist bei diesem Verfahren die Behandlung von kontextabhängigen Modellen, die nicht im Trainingsmaterial enthalten sind, da für diese Modelle keine Parameter geschätzt werden konnten und somit das Abstandsmaß nicht definiert ist.
Beim Unterteilen von Clustern kann dieses Problem durch Einführung von Expertenregeln bzw. -fragen umgangen werden [Bahl u. a., 1991]. In Bild E.0.2 sind beispielhaft einige dieser Regeln für das Clustering von Phonemen vorgestellt. Ausgangspunkt ist ein

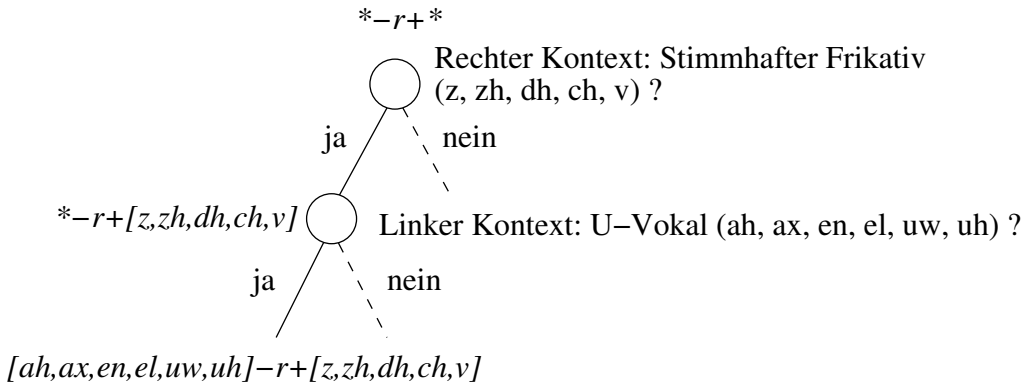


Abbildung E.0.2.: Zusammenfassen von Zuständen anhand von Expertenregeln

System von kontextabhängigen Modellen, bei denen zunächst alle Modelle die Parameter der entsprechenden Monophone verwenden. Anschließend wird aufgrund der einzelnen Regeln versucht, die Modelle zu unterteilen, wobei die Reihenfolge und Anzahl der Regeln vom Algorithmus bestimmt werden. Die Auswahl der einzelnen Regeln geschieht durch Betrachtung der Ausgabedichten der einzelnen Zustände vor und nach einer Unterteilung. Resultat ist ein Baum (Bild E.0.2), mit dem auch unbekannten Modellen eine gut geschätzte Ausgabedichte zugeordnet werden kann. In seiner ursprünglichen Form kann dieses Verfahren nur Gaußförmige Ausgabedichten mit nur einer Mixtur verarbeiten [Young u. a., 2000], Erweiterungen und Details zum baumförmigen Zusammenfassen von Zuständen finden sich in [Willett, 2000].
Da bei hybriden akustischen Modellen die Berechnung der Produktionswahrscheinlichkeitsdichte nicht ohne Daten erfolgen kann (da keine parametrische Beschreibung der Dichten vorliegt), kann das beschriebene baumförmige Zusammenfassen von Zuständen nicht auf diese Modelle übertragen werden. Anstelle einer Neuberechnung der Produktionswahrscheinlichkeitsdichten für jeden Schritt des Clusteringalgorithmus wird in dieser Arbeit ein bereits zusammengefaßtes Set von Gauß-HMM in hybride Modelle umgewandelt und anschließend basierend auf dieser Zusammenfassung neu trainiert.

F. Adaptionsergebnisse im Detail

F.1. Adaption des neuronalen Netzes

Basis NN: MLP273-1000-47 (vergl. Tabelle 5.2 und Tabelle 3.1)
Zusammenfassung in Tabelle 6.2

Sprecher	**WFR adapt. MLP**	**WFR SI**	**Δ rel.**	**#adaptierte Neuronen**
4OA	6,51%	5,47%	+19,0%	203
4OB	5,97%	7,46%	-20,0%	176
4OC	8,98%	10,22%	-12,0%	193
4OD	9,43%	10,38%	-9,0%	135
4OE	14,71%	16,22%	-9,0%	207
4OF	26,69%	30,06%	-11,0%	164
4OG	6,31%	5,41%	+17,0%	208
4OH	20,10%	22,14%	-9,0%	203
4OI	10,89%	11,11%	+2,0%	162
4OJ	28,91%	34,64%	-17,0%	215
Durchschnitt	13,85%	15,31%	-9,5%	186,6
4ND	30,69%	36,23%	-15,0%	164
4NE	28,10%	37,47%	-25,0%	171
4NF	23,32%	35,88%	-35,0%	166
4NH	23,20%	34,25%	-32,0%	191
4NI	24,92%	20,97%	+19,0%	159
4NJ	12,20%	22,87%	-47,0%	167
4NK	15,73%	21,98%	-28,0%	188
4NL	12,38%	17,40%	-29,0%	171
4NM	33,28%	36,94%	-10,0%	200
4NN	31,93%	36,71%	-13,0%	143
Durchschnitt	23,85%	30,07%	-22,0%	172

Tabelle F.1.: Detailliertes WSJ S3-C2/S3-P0 Adaptionsergebnis (Wortfehlerrate) nach Adaption des MLP

Basis RNN: RNN39-400-139 (vergl. Tabelle 5.4 und Tabelle 3.2)
Zusammenfassung in Tabelle 6.3

Sprecher	**WFR adapt. RNN**	**WFR SI**	**Δ rel.**	**#adaptierte Neuronen**
4OA	6,25%	6,25%	0,0%	121
4OB	8,21%	7,71%	+6,0%	110
4OC	9,73%	11,47%	-15,0%	128
4OD	11,56%	12,03%	-4,0%	84
4OE	14,71%	16,52%	-11,0%	121
4OF	29,75%	32,21%	-8,0%	100
4OG	9,01%	9,01%	-3,0%	113
4OH	21,88%	24,68%	-11,0%	106
4OI	14,22%	12,67%	+12,0%	106
4OJ	37,24%	36,98%	+1,0%	116
Durchschnitt	16,26%	16,95%	-4,0%	110,5
4ND	35,72%	41,26%	-13,0%	107
4NE	38,13%	42,74%	-11,0%	115
4NF	29,40%	36,40%	-19,0%	114
4NH	37,57%	46,13%	-19,0%	114
4NI	23,10%	23,40%	-1,0%	101
4NJ	21,73%	29,35%	-26,0%	105
4NK	19,51%	29,91%	-35,0%	104
4NL	17,52%	19,00%	-8,0%	108
4NM	38,85%	46,02%	-16,0%	109
4NN	35,02%	34,60%	-1,0%	97
Durchschnitt	29,66%	34,88%	-15,0%	107,4

Tabelle F.2.: Detailliertes WSJ S3-C2/S3-P0 Adaptionsergebnis (Wortfehlerrate) nach Adaption des RNN

F.2. Adaption der HMM-Gewichte

Basis NN: MLP273-1000-47 (vergl. Tabelle 5.2 und Tabelle 3.1)
Zusammenfassung in den Tabellen 6.4, 6.5 und 6.6

Sprecher	**WFR adapt. HMM grad.**	**WFR adapt. HMM Eigenvoice**	**WFR adapt. HMM MAP**	**WFR SI**
4OA	4,43%	6,25%	5.73%	5,47%
4OB	7,46%	9,20%	7,46%	7,46%
4OC	10,22%	10,47%	10,22%	10,22%
4OD	12,26%	9,67%	9,43%	10,38%
4OE	14,71%	15,92%	14,71%	16,22%
4OF	29,75%	33,44%	29,75%	30,06%
4OG	5,41%	5,11%	6,01%	5,41%
4OH	22,65%	22,65%	20,61%	22,14%
4OI	12,00%	12,67%	12,44%	11,11%
4OJ	35,94%	35,68%	33,85%	34,64%
Durchschn.	15,48%	16,11%	15,02%	15,31%
4ND	33,58%	30,31%	25,03%	36,23%
4NE	28,63%	28,29%	25,73%	37,47%
4NF	30,96%	30,79%	22,02%	35,88%
4NH	26,38%	21,55%	14,36%	34,25%
4NI	20,82%	22,42%	18,84%	20,97%
4NJ	19,95%	19,16%	16,01%	22,87%
4NK	16,64%	15,47%	10,01%	21,98%
4NL	14,09%	14,83%	11,03%	17,40%
4NM	33,12%	32,64%	22,45%	36,94%
4NN	36,43%	38,26%	32,63%	36,71%
Durchschn.	26,06%	25,37%	19,81%	30,07%

Tabelle F.3.: Detailliertes WSJ S3-C2/S3-P0 Adaptionsergebnis (Wortfehlerrate) nach Adaption der HMM-Gewichte

F.3. Adaption der HMM-Gewichte nach NN-Adaption

Basis NN: MLP273-1000-47 (vergl. Tabelle 5.2 und Tabelle 3.1)
Zusammenfassung in den Tabellen 6.7, 6.8 und 6.9

Sprecher	**WFR adapt. HMM grad.**	**WFR adapt. HMM Eigenvoice**	**WFR adapt. HMM MAP**	**WFR SI**
4OA	7,03%	5,99%	6,51%	5,47%
4OB	5,72%	6,72%	5,97%	7,46%
4OC	8,98%	9,23%	10,72%	10,22%
4OD	8,02%	10,38%	9,43%	10,38%
4OE	15,02%	15,62%	15,02%	16,22%
4OF	26,69%	29,45%	26,69%	30,06%
4OG	5,71%	7,51%	6,31%	5,41%
4OH	17,81%	20,61%	18,58%	22,14%
4OI	11,11%	12,89%	11,56%	11,11%
4OJ	28,91%	29,69%	28,91%	34,64%
Durch-schnitt	13,50%	14,81%	13,97%	15,31%
4ND	24,15%	29,43%	23,77%	36,23%
4NE	28,10%	25,73%	24,93%	37,47%
4NF	23,32%	23,58%	18,52%	35,88%
4NH	21,69%	20,72%	14,92%	34,25%
4NI	20,21%	20,67%	16,57%	20,97%
4NJ	12,20%	10,42%	11,44%	22,87%
4NK	13,00%	12,74%	10,66%	21,98%
4NL	11,76%	12,01%	11,64%	17,40%
4NM	23,89%	24,68%	21,34%	36,94%
4NN	30,80%	34,04%	30,94%	36,71%
Durch-schnitt	20,91%	21,40%	18,47%	30,07%

Tabelle F.4.: Detailliertes WSJ S3-P0 Adaptionsergebnis (Wortfehlerrate) nach Adaption des MLP und der HMM-Gewichte

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[Alexander 1986]
ALEXANDER, S. T.: *Adaptive signal processing.* Springer, 1986

[Atal 1974]
ATAL, B. S.: Effectiveness of linear prediction characteristics of the speech wave for automatic speaker identification and verification. In: *Journal of the Acoustical Society of America (JASA)* 5 (1974), Nr. 6, S. 1304–1312

[Bahl u. a. 1991]
BAHL, L. R. ; SOUZA, P. V. ; GOPALAKRISHNAN, P. S. ; NAHAMOO, D. ; PICHENY, M. A.: Decision Trees for Phonological Rules in Continuous Speech. In: *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP).* Toronto, Canada, Mai 1991, S. 185–188

[Barras u. a. 2001]
BARRAS, C. ; LAMEL, L. ; GAUVAIN, J.: AUTOMATIC TRANSCRIPTION OF COMPRESSED BROADCAST AUDIO. In: *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP).* Salt Lake City, Utah, USA, 2001

[Baum 1972]
BAUM, L. E.: An Inequality and Associated Maximization Technique in Statistical Estimation of Probabilistic Functions of Markov Processes. In: *Inequalities* (1972), Nr. 3, S. 1–8

[Bilmes 1998]
BILMES, Jeff A.: A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models / International Computer Science Institute. Berkeley CA, United States, 1998 (TR-97-021). – Forschungsbericht

[Bisani u. Ney 2004]
BISANI, M. ; NEY, Herrmann: Bootstrap estimates for confidence intervals in ASR performance evaluation. In: *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP).* Montreal, Canada, 2004

[Blum u. Li 1991]
BLUM, E. K. ; LI, L. K.: Approximation theory and feedforward networks. In: *Neural Networks* 4 (1991), Nr. 5, S. 511–515

[Botterweck 2000]
BOTTERWECK, Henrik: Very Fast Adaptation for Large Vocabulary Continuous Speech Recognition using Eigenvoices. In: *6th Int. Conference on Spoken Language Processing (ICSLP)*. Beijing, China, Oktober 2000

[Bourlard u. Morgan 1994]
BOURLARD, Herve ; MORGAN, Nelson: *Connectionist Speech Recognition: A Hybrid Approach*. Kluwer Academic Publishers, 1994

[Burges 1998]
BURGES, C. J. C.: *A tutorial on support vector machines for pattern recognition*. Bell Laboratories, Lucent Technologies, 1998

[Caruana 1997]
CARUANA, Rich: Multitask Learning. In: *Machine Learning* (1997), Nr. 28, S. 41–75

[David u. Benkner 1996]
DAVID, K. ; BENKNER, T.: *Digitale Mobilfunksysteme*. Teubner, 1996

[Deller u. a. 1993]
DELLER, John R. ; PROAKIS, John G. ; HANSEN, John H. L.: *Discrete Time Processing of Speech Signals*. Macmillan Publishing Company, 1993

[Duan u. a. 2003]
DUAN, Kaibo ; KEERTHI, S. S. ; CHU, Wei ; SHEVADE, Shirish K. ; POO, Aun N.: Multi-Category Classification by Soft-Max Combination of Binary Classifiers. In: *4th International Workshop on Multiple Classifier Systems*. Surrey, United Kingdom, Juni 2003

[Duda u. Hart 1973]
DUDA, R. O. ; HART, P. E.: *Pattern classification and scene analysis*. Wiley, 1973. – 138–143 S

[Ellis u. Gomez 2001]
ELLIS, Daniel P. ; GOMEZ, Manuel J. R.: Investigations into Tandem Acoustic Modeling for the Aurora Task. In: *European Conference on Speech Communication and Technology*. Aalborg, Denmark, September 2001

[Ellis u. a. 2001]
ELLIS, Daniel P. ; SINGH, Rita ; SIVADAS, Sunil: Tandem Acoustic Modeling in Large-Vocabulary Recognition. In: *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Salt Lake City , Utah, USA, Mai 2001

[Engelbrecht u. a. 1999]
ENGELBRECHT, A. ; FLETCHER, L. ; CLOETE, I.: Variance Analysis of Sensitivity Information for Pruning Feedforward Neural Networks. In: *IEEE International Joint Conference on Neural Networks*. Washington DC, USA, 1999

[Eppinger u. Herter 1993]
EPPINGER, Bernd ; HERTER, Eberhard: *Sprachverarbeitung.* Carl Hanser Verlag München Wien, 1993

[ETSI standard document 2000]
Speech Processing, Transmission and Quality aspects (STQ); Distributed speech recognition; Front-end feature extraction algorithm; Compression algorithms. In: *ETSI ES 201 108 v1.1.1 (2000-02)*, 2000

[ETSI standard document 2003]
Speech Processing, Transmission and Quality aspects (STQ); Distributed speech recognition; Front-end feature extraction algorithm; Compression algorithms. In: *ETSI ES 201 108 v1.1.3 (2003-09)*, 2003

[Evermann u. a. 2005]
EVERMANN, G. ; CHAN, H.Y. ; GALES, M.J.F. ; JIA, B. ; MRVA, D. ; WOODLAND, P.C. ; YU, K.: Training LVCSR Systems on thousands of hours of data. In: *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP).* Philadelphia, PA, USA, März 2005, S. I–209–I–212

[Fahlman 1988]
FAHLMAN, Scott E.: An Emirical Study of Learning Speed in Back-Propagation Networks / Carnegie-Mellon University. 1988 (CMU-CS-88-162). – Forschungsbericht

[Fritsch u. a. 1997]
FRITSCH, Jürgen ; FINKE, M. ; WAIBEL, Alex: Context–Dependent Hybrid HME / Speech Recognition using Polyphone Clustering Decision Trees. In: *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1997, S. 1759–1762

[Furui 1986]
FURUI, S.: Speaker Independent Isolated Word Recognizer Using Dynamic Features of Speech Spectrum. In: *IEEE Transactions on Acoustic, Speech and Signal Processing* 34 (1986), Nr. 1, S. 52–59

[Gales 1998]
GALES, Mark J. F.: Maximum Likelihood Linear Transformations for HMM-Based Speech Recognition. In: *Computer Speech and Language* 12 (1998), S. 75–98

[Ganapathiraju u. a. 1998]
GANAPATHIRAJU, Aravind ; HAMAKER, Jonathan ; PICONE, Joseph: Support Vector Machines for Speech Recognition. In: *5th Int. Conference on Spoken Language Processing (ICSLP).* Sydney, Australia, November 1998, S. 2923–2926

[Ganapathiraju u. a. 2003]
GANAPATHIRAJU, Aravind ; HAMAKER, Jonathan ; PICONE, Joseph: Advances in Hybrid SVM/HMM Speech Recognition. In: *GSPx / International Signal Processing Conference.* Dallas, Texas, USA, April 2003

[Gauvain u. a. 1995]
GAUVAIN, Jean-Luc ; LAMEL, Lori F. ; ADDA-DECKER, Martine: Developments in Continuous Speech Dictation using the ARPA WSJ Task. In: *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Detroit, USA, Mai 1995, S. 65–68

[Gauvain u. Lee 1991]
GAUVAIN, Jean-Luc ; LEE, Chin-Hui: Bayesian Learning of Gaussian Mixture Densities for Hidden Markov Models. In: *Proc. DARPA Speech and Natural Language*, Morgan Kaufmann, Februar 1991

[Gauvain u. Lee 1992]
GAUVAIN, Jean-Luc ; LEE, Chin-Hui: Bayesian Learning for Hidden Markov Model with Gaussian Mixture State Observation Densities. In: *Speech Communication* 11 (1992), Juni, Nr. 2-3

[Gauvain u. Lee 1994]
GAUVAIN, Jean-Luc ; LEE, Chin-Hui: Maximum a-posteriori estimation for multivariate Gaussian mixture observations of Markov chains. In: *IEEE Transactions on Speech and Audio Processing* 2 (1994), S. 291–298

[Gillick u. Cox 1989]
GILLICK, L. ; COX, S. J.: Some statistical issues in the comparison of speech recognition algorithms. In: *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Glasgow, Scotland, 1989

[Hastie u. Tibshirani 1998]
HASTIE, Trevor ; TIBSHIRANI, Robert: Classification by Pairwise Coupling. In: JORDAN, M. I. (Hrsg.) ; KEARNS, M. J. (Hrsg.) ; SOLLA, S. A. (Hrsg.): *Advances in Neural Information Processing Systems*, MIT Press, 1998

[Hengen u. a. 2004]
HENGEN, Heiko ; FEID, Michael ; PANDIT, Madhukar: Überwacht lernende Klassifikationsverfahren im Überblick, Teil 2. In: *at - Automatisieriungstechnik* 52 (2004), Nr. 4, S. A9–A16

[Hermansky 1990]
HERMANSKY, Hynek: Perceptual linear predicitive (PLP) analysis of speech. In: *Journal of the Accoustical Society of America* 87 (1990), Nr. 4, S. 1738–1752

[Hermansky u. a. 2000]
HERMANSKY, Hynek ; ELLIS, Daniel P. W. ; SHARMA, Sangita: Tandem connectionist feature extraction for conventional HMM systems. In: *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Istanbul, Turkey, 2000

[Hermansky u. Morgan 1994]
HERMANSKY, Hynek ; MORGAN, Nelson: RASTA Processing of Speech. In: *IEEE Transactions on Speech and Audio Processing* 2 (1994), Nr. 4, S. 578–589

[Hild u. Waibel 1993]
HILD, Hermann ; WAIBEL, Alex: Connected Letter Recognition with a Multi-State Time Delay Neural Network. In: HANSON, Stephen J. (Hrsg.) ; COWAN, Jack D. (Hrsg.) ; GILES, C. L. (Hrsg.): *Advances in Neural Information Processing Systems* Bd. 5, Morgan Kaufmann, San Mateo, CA, USA, 1993, S. 712–719

[Hirsch u. Pearce 2000]
HIRSCH, H. G. ; PEARCE, D.: The AURORA experimental framework for the performance evaluation of speech recognition systems under noisy conditions. In: *ISCA ITRW ASR2000*, 2000

[Hirsch u. a. 1991]
HIRSCH, Hans-Günther ; MEYER, P. ; RUEHL, H.: Improved speech recognition using high-pass filtering of subband envelopes. In: *European Conference on Speech Communication and Technology*. Genova, Switzerland, 1991

[Huo u. Chan 1995]
HUO, Qiang ; CHAN, Chorkin: Bayesian Adaptive Learning of the Parameters of Hidden Markov Model for Speech Recognition. In: *IEEE Transactions on Speech and Audio Processing* 3 (1995), Nr. 5, S. 334–345

[Huo u. Lee 1997]
HUO, Qiang ; LEE, Chin-Hui: On-Line Adaptive Learning of the Continuous Density Hidden Markov Model Based on Approximate Recursive Bayes Estimate. In: *IEEE Transactions on Speech and Audio Processing* 2 (1997), Nr. 5, S. 161–172

[Igel u. Hüsken 2000]
IGEL, Christian ; HÜSKEN, Michael: Improving the Rprop Learning Algorithm. In: *Proceedings of the Second International Symposium on Neural Computation NC'2000*, ICSC Academic Press, 2000, S. 115–121

[ITU recommendation G.712 1996]
Transmission performance characteristics of pulse code modulation channels. In: *ITU recommendation G.712*, 1996

[Jelinek 1976]
JELINEK, Fred: Continuous Recognition by Statistical Methods. In: *Proceedings of the IEEE* 64 (1976), Nr. 4, S. 532–555

[Joost u. Schiffmann 1998]
JOOST, Merten ; SCHIFFMANN, Wolfram: Speeding up Backpropagation Algorithms by using Cross-Entropy combined with Pattern Normalization. In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems (IJUKFS)* 6 (1998), Nr. 2, S. 117–126

[Jordan 1986]
JORDAN, Michael I.: Attractor dynamics and parallelism in a connectionist sequen-

tial machine. In: *Proc. of the Eighth Conference of the Cognitive Science Society*. Englewood Cliffs, NJ, USA : Erlbaum, 1986, S. 531–546

[Killer u. a. 2003]
KILLER, Mirjam ; STÜKER, Sebastian ; SCHULTZ, Tanja: Grapheme Based Speech Recognition. In: *European Conference on Speech Communication and Technology*. Geneva, Switzerland, September 2003, S. 3141–3144

[Kubala u. a. 1994]
KUBALA, F. ; BELLEGARDA, J.R. ; COHEN, J. ; PALLETT, D.S. ; PAUL, D.B. ; PHILLIPS, M. ; RAJASEKARAN, R. ; RICHARDSON, F. ; RILEY, M. ; ROSENFELD, R. ; ROTH, R. ; WEINTRAUB, M.: The Hub and Spoke Paradigm for CSR Evaluation. In: *Proc. of the ARPA Spoken Language Technology Workshop*. Plainsboro, New Jersey : Morgan Kaufmann, März 1994, S. 9–14

[Kuhn u. a. 1999]
KUHN, R. ; NGUYEN, P. ; JUNQUA, J.-C. ; BOMAN, R. ; NIEDZIELSKI, N. ; FINCKE, S. ; FIELD, K. ; CONTOLINI, M.: Fast Speaker Adaptation using A Priori Knowledge. In: *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Phoenix, USA, März 1999

[Kuhn u. a. 1998]
KUHN, R. ; NGUYEN, P. ; JUNQUA, J.-C. ; GOLDWASSER, L. ; NIEDZIELSKI, N. ; FINCKE, S. ; FIELD, K. ; CONTOLINI, M.: Eigenvoices for Speaker Adaptation. In: *5th International Conference on Spoken Language Processsing (ICSLP)*. Sydney, Australia, Dezember 1998

[Kuhn u. a. 2000]
KUHN, Roland ; JUNQUA, Jean-Claude ; NGUYEN, Patrick ; NIEDZIELSKI, Nancy: Rapid Speaker Adaptation in Eigenvoice Space. In: *IEEE Transactions on Speech and Audio Processing* 8 (2000), November, Nr. 6, S. 695–707

[Lang 1994]
LANG, Manfred K.: Towards User Adequate Human-Computer-Interaction. In: B. HORVAT, Z. K. (Hrsg.): *Modern Modes of Man-Machine-Communication*. Maribor, Slowenien, 1994, S. 1/1–1/9

[Lang u. Stahl 1994]
LANG, Manfred K. ; STAHL, Holger: Spracherkennung für einen ergonomischen Mensch-Maschine-Dialog. In: *mikroelektronik* 8 (1994), Nr. 2, S. 78–82

[Launay u. a. 2002]
LAUNAY, Benoit ; SIOHAN, Olivier ; SURENDRAN, Arun ; LEE, Chin-Hui: Towards knowledge-based features for HMM based large vocabulary automatic speech recognition. In: *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Orlando, Florida, USA, 2002

[Leggetter u. Woodland 1995]
LEGGETTER, Christian J. ; WOODLAND, Phil C.: Maximum likelihood linear regression for speaker adaptation of continuous density hidden Markov models. In: *Computer Speech and Language* 9 (1995), S. 171–185

[Leonard 1984]
LEONARD, R.G.: A database for speaker independent digit recognition. In: *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. San Diego, California, USA, 1984

[Lin u. a. 2003]
LIN, Hsuan-Tien ; LIN, Chih-Jen ; WENG, Ruby Chiu-Hsing: A note on Platt's probabilistic outputs for support vector machines / Computer Science and Information Engineering, National Taiwan University. Version: Mai 2003. `http://www.csie.ntu.edu.tw/~cjlin/papers.html` (1). – Forschungsbericht. – Elektronische Ressource

[Linde u. a. 1980]
LINDE, Y. ; BUZO, A. ; GRAY, R.: An Algorithm for Vector Quantizer Design. In: *IEEE Transactions on Communications* 28 (1980), S. 84–95

[Morgado 2004]
MORGADO, Miguel: *Optimizing neural network's topologies*, Lehrstuhl für Mensch-Maschine-Kommunikation, Technische Universität München, Diplomarbeit, Oktober 2004

[Neto u. a. 1995]
NETO, J. ; ALMEIDA, L. ; HOCHBERG, M. ; MARTINS, C. ; NUNES, L. ; RENALS, S. ; ROBINSON, A.: Speaker-Adaptation for Hybrid HMM-ANN Continuous Speech Recognition System. In: *European Conference on Speech Communication and Technology*. Madrid, Spain, September 1995, S. 2171–2174

[Neukirchen 1999]
NEUKIRCHEN, Christoph: *Integration neuronaler Vektorquantisierer in ein Hidden-Markov-Modell-basiertes System zur automatischen Spracherkennung*, Faculty of Electrical Engineering, Gerhard-Mercator-University Duisburg, Diss., 1999

[Nillson 1971]
NILLSON, N. J.: *Problem Solving Methods of Artificial Intelligence*. McGraw-Hill, 1971

[Parveen u. Green 2003]
PARVEEN, Shahla ; GREEN, Phil: Multitask Learning in Connectionist Robust ASR using Recurrent Neural Networks. In: *European Conference on Speech Communication and Technology*. Geneva, Switzerland, September 2003, S. 1813–1816

[Paul u. Baker 1992]
PAUL, Douglas B. ; BAKER, Janet M.: The Design for the Wall Street Journal-based CSR Corpus. In: *International Conference on Spoken Language Processing (ICSLP)*. Banff, Canada, Oktober 1992, S. 899–902

[Picone 1993]
PICONE, Joseph W.: Signal Modeling Techniques in Speech Recognition. In: *Proceedings of the IEEE* 81 (1993), Nr. 9, S. 1215–1247

[Platt 2000]
PLATT, John C.: Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In: SMOLA, A.J. (Hrsg.) ; BARTLETT, P. (Hrsg.) ; SCHOELKOPF, B. (Hrsg.) ; SCHUURMANS, D. (Hrsg.): *Advances in Large Margin Classifiers*, MIT Press, 2000, S. 61–74

[Reichl 1996]
REICHL, Wolfgang: *Beiträge zur statistischen Modellierung und effizienten Dekodierung in der automatischen Spracherkennung*, Fakultät für Elektrotechnik und Informationstechnik, Technische Universität München, Diss., Januar 1996

[Reiter u. a. 2005]
REITER, Stephan ; SCHREIBER, Sascha ; RIGOLL, Gerhard: Multimodal Meeting Analysis by Segmentation and Classification of Meeting Events based on a Higher Level Semantic Approach. In: *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Philadelphia, PA, USA, März 2005, S. II–161–II–164

[Riedmiller u. Braun 1993]
RIEDMILLER, M. ; BRAUN, H.: A Direct Adaptive Method for Faster Backpropagation Learning: The RPROP Algorithm. In: *IEEE International Conference on Neural Networks*, 1993

[Rigoll 1990]
RIGOLL, Gerhard: Large Vocabulary Hidden Markov Model Based Speech Recognition. In: *European Transactions on Telecommunications and Related Technologies* 1 (1990), Nr. 1, S. 37–42

[Rigoll 1994a]
RIGOLL, Gerhard: Maximum Mutual Information Neural Networks for Hybrid Connectionist-HMM Speech Recognition Systems. In: *IEEE Transactions on Speech and Audio Processing, Special Issue on Neural Networks for Speech* 2 (1994), Januar, Nr. 1, S. 175–184

[Rigoll 1994b]
RIGOLL, Gerhard: Mutual Information Neural Networks: A New Connectionist Approach for Dynamic Speech Recognition Tasks. In: *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Adelaide, April 1994, 645–648

[Rigoll 1994c]
RIGOLL, Gerhard: *Neuronale Netze - Eine Einführung für Ingenieure, Informatiker und Naturwissenschaftler*. Expert Verlag, 1994 (Kontakt und Studium)

[Robinson 1994]
ROBINSON, A. J.: An Application of Recurrent Nets to Phone Probability Estimation. In: *IEEE Transactions on Neural Networks* 5 (1994), März, Nr. 2, S. 298–305

[Rottland 2000]
ROTTLAND, Jörg: *Ein hybrider Ansatz zur automatischen Spracherkennung und Sprecheradaption für große Wortschätze*, Faculty of Electrical Engineering, Gerhard-Mercator-University Duisburg, Diss., Februar 2000

[Rottland u. a. 1998]
ROTTLAND, Jörg ; NEUKIRCHEN, Christoph ; RIGOLL, Gerhard: Speaker Adaptation for Hybrid MMI/Connectionist Speech Recognition Systems. In: *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Seattle, Mai 1998, 465–468

[Rottland u. Rigoll 2000]
ROTTLAND, Jörg ; RIGOLL, Gerhard: Tied Posteriors: An Approach for Effective Introduction of Context Dependency in Hybrid NN/HMM LVCSR. In: *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Istanbul, Turkey, 2000

[Ruske 1988]
RUSKE, Günther: *Automatische Spracherkennung*. R. Oldenbourg Verlag, 1988

[Salomon 2001]
SALOMON, Jesper: *Support Vector Machines for Phoneme Classification*, School of Artificial Intelligence, Division of Informatics, University of Ediburgh, Diplomarbeit, 2001

[Salomon u. a. 2002]
SALOMON, Jesper ; KING, Simon ; OSBORNE, Miles: Framewise Phone Classification using Support Vector Machines. In: *7th Int. Conference on Spoken Language Processing (ICSLP-Interspeech)*. Denver, Colorado, USA, September 2002

[Santini u. Bimbo 1995]
SANTINI, S. ; BIMBO, A. D.: Recurrent Neural Networks Can Be Trained to Be Maximum A Posteriori Probability Classifiers. In: *Neural Networks* 8 (1995), Nr. 1, S. 25–29

[Schalkoff 1994]
SCHALKOFF, Robert J.: *Artificial Neural Networks*. McGraw-Hill, 1994

[Schukat-Talamazzini 1995]
SCHUKAT-TALAMAZZINI, E. G.: *Automatische Spracherkennung*. Vieweg, 1995

[Schuller u. a. 2005]
SCHULLER, Björn ; JIMENEZ VILLAR, Raquel ; RIGOLL, Gerhard ; LANG, Manfred: Meta-Classifiers in Acoustic and Linguistic Feature Fusion-Based Affect Recognition.

In: *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Philadelphia, PA, USA, März 2005, S. 325–328

[Schölkopf 2001]
SCHÖLKOPF, Bernhard: Tutorial: SVM and Kernel methods. In: *Proc. of Neural Information Processing Systems (NIPS)*. Vancouver, Canada, 2001

[Senior 1994]
SENIOR, Andrew W.: *Off-line Cursive Handwriting Recognition using Recurrent Neural Networks*, University of Cambridge, Diss., September 1994

[Shire 2001]
SHIRE, Michael: Relating Frame Accuracy with Word Error in Hybrid ANN-HMM ASR. In: *European Conference on Speech Communication and Technology*. Aalborg, Danmark, September 2001

[Sivadas u. Hermansky 2002]
SIVADAS, Sunil ; HERMANSKY, Hynek: Hierarchical Tandem Feature Extraction. In: *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Orlando, Florida, USA, Mai 2002

[Sommer 2004]
SOMMER, Elmar: *Sprecheradaption mit Eigenvoices*, Lehrstuhl für Mensch-Maschine-Kommunikation, Technische Universität München, Diplomarbeit, November 2004

[Stadermann u. a. 2005]
STADERMANN, Jan ; KOSKA, Wolfram ; RIGOLL, Gerhard: Multi-task Learning Strategies for a Recurrent Neural Net in a Hybrid Tied-Posteriors Acoustic Model. In: *9th European Conference on Speech Communication and Technology (Interspeech)*. Lisboa, Portugal, September 2005

[Stadermann u. a. 2001]
STADERMANN, Jan ; MEERMEIER, Ralf ; RIGOLL, Gerhard: Distributed Speech Recognition using Traditional and Hybrid Modeling Techniques. In: *European Conference on Speech Communication and Technology*. Aalborg, Denmark, September 2001

[Stadermann u. Rigoll 2001]
STADERMANN, Jan ; RIGOLL, Gerhard: Comparison of Standard and Hybrid Modeling Techniques for Distributed Speech Recognition. In: *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. Madonna di Campiglio Trento, Italy, Dezember 2001

[Stadermann u. Rigoll 2003a]
STADERMANN, Jan ; RIGOLL, Gerhard: Comparing NN Paradigms in Hybrid NN/HMM Speech Recognition using Tied Posteriors. In: *IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*. St. Thomas, U.S. Virgin Islands, November 2003

[Stadermann u. Rigoll 2003b]
STADERMANN, Jan ; RIGOLL, Gerhard: Distributed Speech Recognition on the WSJ task. In: *European Conference on Speech Communication and Technology.* Geneva, Suisse, September 2003

[Stadermann u. Rigoll 2003c]
STADERMANN, Jan ; RIGOLL, Gerhard: Flexible Feature Extraction and HMM Design for a Hybrid Distributed Speech Recognition System in Noisy Environments. In: *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP).* Hongkong, China, April 2003

[Stadermann u. Rigoll 2004]
STADERMANN, Jan ; RIGOLL, Gerhard: A Hybrid SVM/HMM Acoustic Modeling Approach to Automatic Speech Recognition. In: *8th Int. Conference on Spoken Language Processing (ICSLP-Interspeech).* Jeju Island, Korea, Oktober 2004

[Stadermann u. Rigoll 2005a]
STADERMANN, Jan ; RIGOLL, Gerhard: Two-Stage Speaker Adaptation of Hybrid Tied-Posterior Acoustic Models. In: *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP).* Philadelphia, PA, USA, März 2005, S. I–977–I–980

[Stadermann u. Rigoll 2005b]
STADERMANN, Jan ; RIGOLL, Gerhard: Verteilte Spracherkennung mit hybriden akustischen Modellen. In: *31. Deutsche Jahrestagung für Akustik, DAGA05.* München, Deutschland, März 2005

[Ström 1996]
STRÖM, Nikko: Speaker-Adaptation by Modeling the Speaker Variation in a Continuous Speech Recognition System. In: *4th Int. Conference on Spoken Language Processing (ICSLP).* Philadelphia, USA, Oktober 1996

[Syed u. a. 1999]
SYED, Nadeem A. ; LIU, Huan ; SUNG, Kah K.: Incremental Learning with Support Vector Machines. In: *Proc. Int. Joint Conf. on Artificial Intelligence (IJCAI-99)*, 1999

[Thomae u. a. 2003]
THOMAE, Matthias ; FABIAN, Tibor ; LIEB, Robert ; RUSKE, Günther: A One-Stage Decoder for Interpretation of Natural Speech. In: *Proceedings of the IEEE International Conference on Natural Language Processing and Knowledge Engineering (NLP-KE 2003).* Beijing, China, Oktober 2003, S. 56–64

[Vapnik 1995]
VAPNIK, Vladimir: *The Nature of Statistical Learning Theory.* New York : Springer Verlag, 1995

[Vapnik 1998]
VAPNIK, Vladimir: *Statistical Learning Theory.* New York : John Wiley and Sons, Inc., 1998

[Wallhoff u. a. 2000]
WALLHOFF, Frank ; WILLETT, Daniel ; RIGOLL, Gerhard: Frame Discriminative and Confidence-Driven Adaptation for LVCSR. In: *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Istanbul, Turkey, Juni 2000, 1835–1838

[Wallhoff u. a. 2001]
WALLHOFF, Frank ; WILLETT, Daniel ; RIGOLL, Gerhard: Scaled Likelihood Linear Regression for Hidden Markov Model Adaptation. In: *European Conference on Speech Communication and Technology*. Aalborg, Denmark, September 2001

[Wendt u. a. 2001]
WENDT, Sascha ; FINK, Gernot A. ; KUMMERT, Franz: Forward Masking for Increased Robustness in Automatic Speech Recognition. In: *European Conference on Speech Communication and Technology*. Aalborg, Denmark, September 2001

[Westphal 1997]
WESTPHAL, Martin: The use of Cepstral Means in Conversational Speech Recognition. In: *European Conference on Speech Communication and Technology*. Rhodes, Greece, September 1997

[Westwood 1999]
WESTWOOD, Robert: *Speaker Adaptation Using Eigenvoices*, Department of Engineering, Cambridge University, Diplomarbeit, August 1999

[Willett 2000]
WILLETT, Daniel: *Beitraege zur statistischen Modellierung und effizienten Dekodierung in der automatischen Spracherkennung*, Faculty of Electrical Engineering, Gerhard-Mercator-University Duisburg, Diss., November 2000. `http://www.ub.uni-duisburg.de/diss/diss0126/`. – Elektronische Ressource

[Willett u. a. 1998]
WILLETT, Daniel ; NEUKIRCHEN, Christoph ; RIGOLL, Gerhard: DUCODE - Der Stackdekoder / Faculty of Electrical Engineering - Computer Science, Gerhard-Mercator-University Duisburg. Version: November 1998. `http://www.fb9-ti.uni-duisburg.de/publ/98/stack.ps.gz` (1). – Forschungsbericht. – Elektronische Ressource

[Williams u. Zipser 1990]
WILLIAMS, Ronald J. ; ZIPSER, David: Gradient-Based Learning Algorithms for Recurrent Connectionist Networks / Northeastern University. Boston, United States, April 1990 (NU-CCS-90-09). – Forschungsbericht

[Woodland u. a. 1995]
WOODLAND, P.C. ; LEGGETTER, C.J. ; ODELL, J.J. ; VALTCHEV, V. ; YOUNG, S.J.: The 1994 HTK Large Vocabulary Speech Recognition System. In: *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. Detroit, Michigan, 1995, S. 73–76

[Young u. a. 2000]
YOUNG, Steve ; KERSHAW, Dan ; ODELL, Julian ; OLLASON, Dave ; VALTCHEV, Valtcho ; WOODLAND, Phil: *The HTK Book*. HTK, Version 3.0, 2000

[Yuk u. Flanagan 1999]
YUK, D. ; FLANAGAN, J.: Telephone speech recognition using neural networks and hidden Markov models. In: *IEEE Int. Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1999

[Zhou u. Austin 1998]
ZHOU, P. ; AUSTIN, J.: Learning Criteria for Training Neural Network Classifiers. In: *Neural Computing and Applications* 7 (1998), S. 334–342

[Zhu u. a. 2004]
ZHU, Qifeng ; CHEN, Barry ; MORGAN, Nelson ; STOLCKE, Andreas: On using MLP features in LVCSR. In: *8th Int. Conference on Spoken Language Processing (ICSLP-Interspeech)*. Jeju Island, Korea, Oktober 2004

[Zwicker u. Terhardt 1980]
ZWICKER, Eduard ; TERHARDT, Ernst: Analytical expressions for critical-band rate and critical bandwidth as a function of frequency. In: *Journal of the Acoustical Society of America* 68 (1980), Nr. 5, S. 1523–1525