Kapitel 4

Modellierung

Bei der Frage nach einem geeigneten Paradigma zur Modellierung der Handschrift ließen sich theoretisch alle bisher genutzten Prinzipien der Mustererkennung untersuchen. Dies würde einerseits sowohl regelbasierte/heuristische Methoden berühren, wie auch Verfahren, mit denen sich bestimmte Eigenschaften aus Beispielen erlernen lassen, wie z. B. neuronale Netze oder Hidden Markov Modelle.
Abgesehen von eher akademischen Problemstellungen der Mustererkennung scheint sich zunehmend die Erkenntnis durchzusetzen, dass sich die sog. lernenden Methoden - insbesondere bei komplexen Mustererkennungsproblemen - im Vergleich zu regelbasierten Ansätzen durch eine deutlich höhere Robustheit und Generalisierungsfähigkeit auszeichnen.
Als ’komplex’ ist im Zusammenhang mit einer automatischen Verarbeitung durchaus die menschliche Sprache, sowohl in gesprochener als auch in geschriebener Form einzustufen. Diese speziellen Mustererkennungsprobleme weisen darüber hinaus noch eine besondere Eigenschaft auf: Es handelt sich dabei um dynamische Muster, bei denen zunächst unbekannt ist, wie groß die Gesamtlänge des beobachteten Musters ausfällt, oder wann genau welche Merkmale auftreten und wie stark diese dann ausgeprägt sind. Verschleifungseffekte mit einer einhergehenden zeitlichen Unschärfe der einzelnen Klassengrenzen erschweren die Situation zudem.
Hidden Markov Modelle scheinen genau für solche Problemstellungen ein probates Paradigma darzustellen, da sie über wichtige Eigenschaften verfügen, wie:

- automatische Lernfähigkeit anhand vorgegebener Trainingsbeispiele,
- hervorragende Eigenschaften zur Modellierung dynamischer Muster und
- Möglichkeiten zur integrierten Erkennung und Segmentierung.

Diese Vorteile von Hidden Markov Modellen ermöglichen über die oben genannten Kerneinsatzgebiete weitere interessante Anwendungsmöglichkeiten, wie z. B. im Bereich der

Dokumentenverarbeitung [Bra00, Bra99b, Bra99a], der Unterschriftenverifikation [Rig98b], der automatischen Indexierung von Videosequenzen [Eic97a], der Videosequenz-Erkennung [Rig98a, Eic98, Eic97b] oder aber der Erkennung handskizzierter Piktogramme [Mü98, Mü99], die für eine inhaltsorientierte Bilddatenbankabfrage verwendet werden können [1].
Als genereller Ansatz für die Online-Handschrifterkennung bieten sich im besonderen HMM auf Grund der genannten Vorteile und Vielseitigkeiten an. Innerhalb dieses Rahmens eröffnen sich jedoch wiederum zahlreiche Realisierungsmöglichkeiten. Die nachfolgenden Abschnitte sollen zunächst eine Antwort darauf liefern, welche grundlegende Modellierungsform im Bereich der Online-Handschrifterkennung vorteilhaft ist.

4.1 Vergleich zwischen diskreten und kontinuierlichen Modellen

Neben anderen Faktoren hat die Wahl der Modellierungsform einen entscheidenden Einfluß auf die Erkennungsgenauigkeit des Systems. In der Spracherkennung scheinen sich nach einigen z. T. widersprüchlichen Ergebnissen verstärkt kontinuierliche Modelle durchzusetzen. So wurde in einigen Arbeiten dokumentiert [Bah81, Bro87], dass diskrete Spracherkennungssysteme im Vergleich zu kontinuierlichen Systemen höhere Erkennungsraten zeigen, während in [Rab85, VNG87] gegensätzliche Ergebnisse belegt werden. Die teilweise stark abweichenden Ergebnisse in den referenzierten Veröffentlichungen zeigen, dass eine Untersuchung der Frage nach der geeigneten Modellierungsform individuell erfolgen sollte. Abb. 4.1 verdeutlicht die wesentlichen Unterschiede zwischen diskreter und kontinuierlicher Modellierung. Bei kontinuierlichen HMM werden die Ausgabeverteilungen in parametrischer Form ausgeführt. Unter der Annahme einer Gauß-förmigen Verteilung der Merkmale kommen i. d. R. auch Gauß'sche Mischverteilungen zum Einsatz.

Bei diskreten Systemen wird der (im Beispiel zweidimensionale) Merkmalsraum zunächst mittels eines Vektorquantisierers (VQ) in Partitionen eingeteilt. Die Merkmalsvektoren $\vec{x}$ werden dazu anhand eines Abstandsmaßes dem nächstliegenden Codebuchvektor des Vektorquantisierers zugeordnet. Ein Codebuch der Größe N_C führt folglich zu N_C verschiedenen Partitionen (Voronoi-Zellen) des Merkmalsraumes, die bei der Verarbeitung i-dimensionaler Merkmalsvektoren i-dimensionale konvexe Polytope bilden. Die diskrete Ausgabeverteilung $b_s(n)$ des Zustandes s modelliert damit die Auftrittswahrscheinlichkeit der in der n-ten VQ-Partition zusammengefassten Merkmalsvektoren. Damit sind diskrete Modelle unabhängig von Annahmen bezüglich der Merkmalsverteilung. Als Standard-Lösung werden in diskreten Systemen zur Partitionierung des Merkmalsraumes unüberwachte, selbstorganisierende,

[1] Siehe dazu auch http://www.fb9-ti.uni-duisburg.de/rotdemo.html

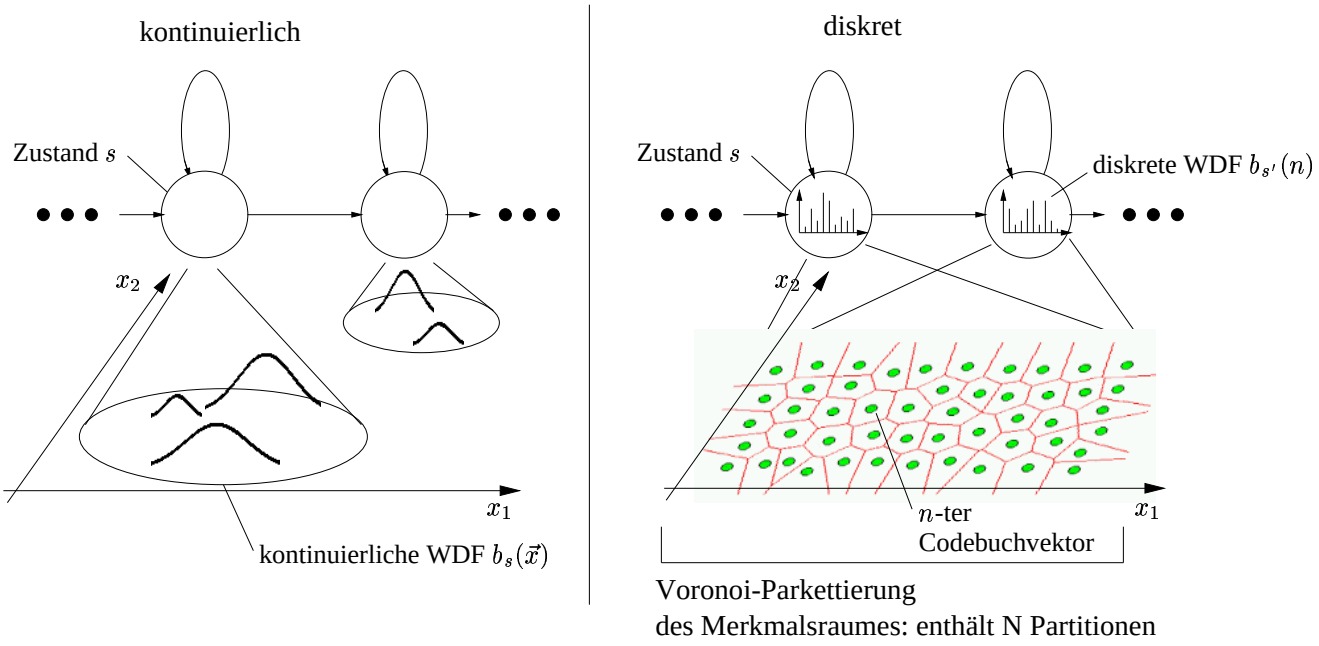

Abbildung 4.1: Prinzipielle Darstellung kontinuierlicher und diskreter HMM

ggf. hierarchische Vektorquantisierer [Lee89] eingesetzt. Als weiterer Vorteil diskreter Modelle gilt die effiziente Berechnung der Ausgabewahrscheinlichkeiten, was insbesondere bei der Erkennung zu deutlichen Geschwindigkeitsvorteilen führen kann. Bei gegebenem VQ-Index n läßt sich die Berechnung von $p(n|s)$ auf einen einfachen Speicherzugriff auf die als Vektor abgelegten Wahrscheinlichkeitsdichtefunktionen realisieren.
Beiden Modellierungsformen ist allerdings die generelle Vorgehensweise bei der Modellbildung gemein. Ausgehend von allgemeinen Modellprototypen, mit denen die Modellstruktur definiert wird, werden aus einigen vorsegmentierten Zeichenbeispielen initiale Verteilungen und Übergangswahrscheinlichkeiten ermittelt. Experimentell konnte als optimale Struktur das Links-Rechts-Modell mit 12 Zuständen pro Graphem (bzw. Zeichen) bestimmt werden. Ein Links-Rechts-Modell ist mit Selbsttransitionen, sowie mit Übergängen zum Nachfolgezustand ausgestattet.
Zur Berechnung initialer Verteilungen werden in einer ersten Iteration die vorsegmentierten Beobachtungssequenzen eines Graphems in 12 gleich lange Abschnitte unterteilt. Diese 12 gleich langen aufeinander folgenden Abschnitte werden den 12 in einem Modell enthaltenen aufeinander folgenden Zuständen zugeordnet. Für diese Initialisierungsstufe werden die exemplarisch im Anhang B, in Abb. B.1 gezeigten Trainingsbeispiele verwendet. Anhand dieser zunächst willkürlichen Zuordnung lassen sich erste Verteilungsparameter berechnen. Diese Verteilungsparameter dienen in einer nächsten Iteration dazu, mit einem sog. Viterbi-Alignment die vorhandenen Trainingsdaten den Zuständen zuzuordnen, woraus sich die Parameter mit jeder Iteration verfeinert schätzen lassen. Abb. 4.2 zeigt die Zustandssegmentie-

rung einer Trainingssequenz in der 1. Iteration der Initialisierung und nach erfolgtem Training. Es zeigt sich dabei, dass sich trotz willkürlicher Zuordnung gleich langer Sequenzen zwecks Initialisierung, das Alignment im Laufe des Trainings teilweise deutlich verschieben kann. Es bilden sich Zustände für verschiedene, unterschiedlich lange Merkmalssequenzen heraus.

Im Anschluß an das Viterbi-Training werden die so initialisierten Modelle mit dem Baum-

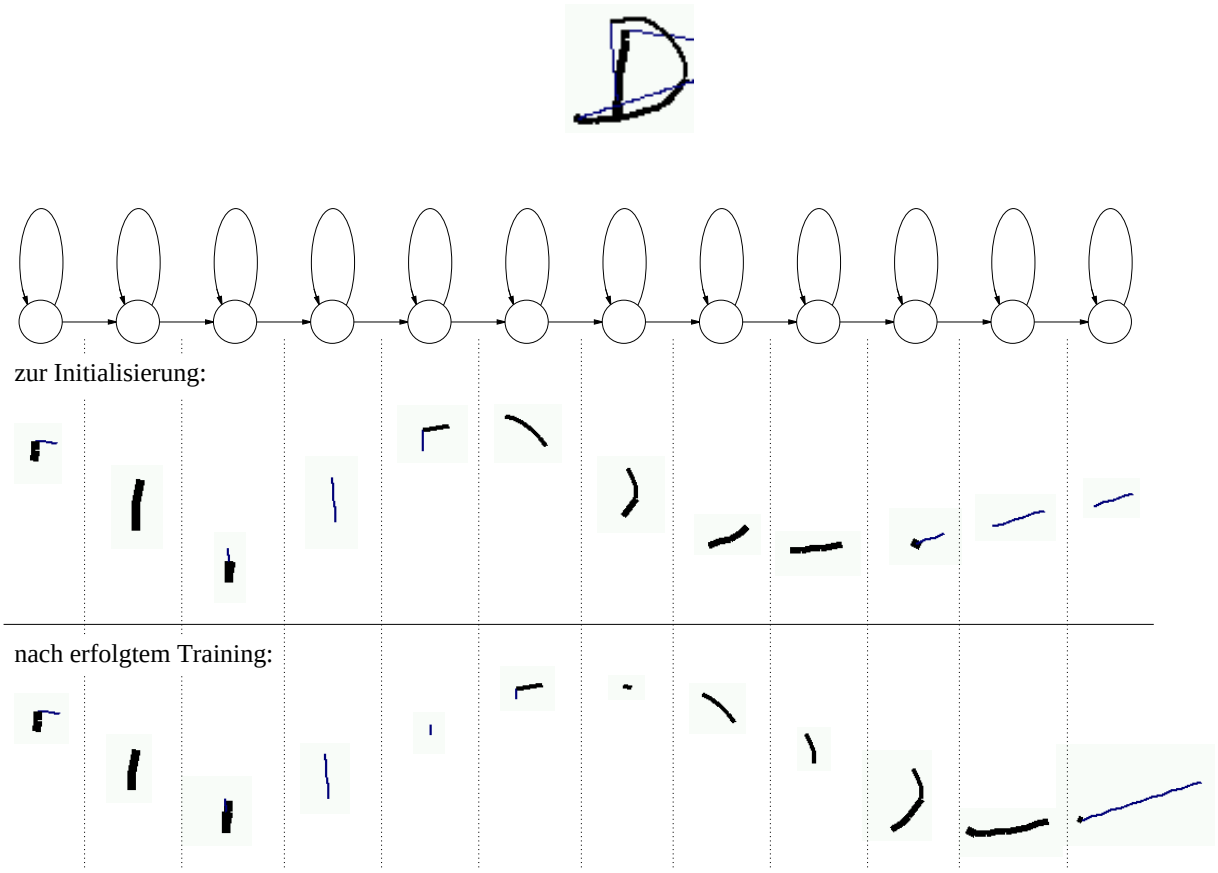

Abbildung 4.2: Alignment nach Initialisierung und nach vollständigem Training

Welch-Algorithmus weiter optimiert. Diese Optimierung läßt sich in zwei Stufen einteilen. Die erste Stufe basiert wiederum auf den schon für das Viterbi-Training vorsegmentierten Daten. In der zweiten Stufe werden dann ganze Wörter oder Sätze für das Training herangezogen. Beispiele dazu sind in Abb. B.2 gezeigt. Im Gegensatz zu dem Viterbi-Training werden die Trainingsmuster den Zuständen nicht mehr in deterministischer Weise zugeordnet, sondern mit gewissen Wahrscheinlichkeitswerten.

Grundsätzlich steckt hinter dem Gedanken der Parameter-Optimierung die Idee, die Wahrscheinlichkeit zu maximieren, dass die gegebene Trainingsmenge X von den vorhandenen Modellen λ generiert wurde. Dieses - auch als Likelihood bekannte - Gütemaß $p(X|\lambda)$ läßt sich bedauerlicherweise weder analytisch, noch direkt maximieren. Statt dessen wurde in

[Bau69] vorgeschlagen, die Kullback-Leibler Distanz

$$Q(\lambda, \lambda^*) = \sum_S p(S|X, \lambda) \log p(X, S|\lambda^*) \tag{4.1}$$

zwischen einem Ausgangsmodell λ und einem optimierten Modell λ^* zu maximieren. Die Optimierungsgleichungen der Modellparameter (sowohl der Ausgabeverteilungen wie auch der Übergangswahrscheinlichkeiten) lassen sich direkt aus Glg. 4.1 ableiten. Wird ein Parametersatz λ^* berechnet, der Glg. 4.1 maximiert, läßt sich weiterhin zeigen, dass

$$P(X|\lambda^*) \geq P(X|\lambda) \tag{4.2}$$

gilt womit sich in iterativer Weise die HMM hinsichtlich deren Likelihood optimieren lassen. Weitergehende Grundlagen zur Theorie der Hidden Markov Modelle können [ST95] entnommen werden. Eine sehr gute und verständliche Zusammenfassung ist in [Rab89] zu finden. Im weiteren sollen grundlegende Aspekte zu diesem Thema lediglich im Bedarfsfall betrachtet werden.

4.2 Verwendung neuronaler Netze zur Merkmalsquantisierung

Der Vergleich der Eigenschaften kontinuierlicher und diskreter HMM zeigt einerseits, dass es vorteilhaft ist, auf bestimmte Verteilungsannahmen der Merkmale zu verzichten. Andererseits stellt sich durch die Vektorquantisierung immer auch ein Quantisierungsfehler ein. Ein weiterer Schwachpunkt der bekannten diskreten Standardverfahren ist, dass der selbstorganisierende Vektorquantisierer ohne Klasseninformation des Merkmalsvektors trainiert wird. Damit stellt der Vektorquantisierer eine weitere unabhängige, verlustbehaftete Verarbeitungsstufe im Gesamtsystem dar. Die wesentlichen Nachteile diskreter Standard-Systeme sind in den folgenden Punkten zusammenzufassen:

- Die Merkmalskompression durch vorangehende Vektorquantisierung ist stets verlustbehaftet.
- Die Berechnung des Codebuches, welches später den Merkmalsraum partitioniert, wird bei den gängigen Standardverfahren (wie z. B. k-means) in selbstorganisierender Weise durchgeführt. D. h. es wird keine Klasseninformationen ausgenutzt.

Um die Schwachpunkte selbstorganisierender Vektorquantisierer zumindest teilweise zu umgehen, bieten sich neuronale Netze an. Eine umfassende Beschreibung zu gängigen

neuronalen Netzen und den zugrunde liegenden Paradigmen ist in [Zel94] gegeben. Die speziell für die Handschrifterkennung adaptierten Neuronalen Netztypen erlauben es, das Wissen über die Klassenzugehörigkeit von Merkmalsvektoren zu integrieren. Eine derartige Verknüpfung neuronaler Netze (NN) mit Hidden Markov Modellen (HMM) führt zu einer speziellen Variante hybrider NN/HMM-Systeme [Rig94]. Verschiedene neuronale Paradigmen wurden bereits zur Verwendung in hybriden Spracherkennungssystemen untersucht [Neu99, Rot00]. Vergleichbare Untersuchungen zur Online-Handschrifterkennung existierten bisher nicht [Rig96a, Rig98c]. Während in [Neu99] die synchrone Optimierung mehrerer Codebücher für Perzeptron-Strukturen dokumentiert ist, sollen die nachfolgenden Abschnitte insbesondere das sequentielle wie auch das multiple Training von nN- (nächster-Nachbar-) Vektorquantisierern und die damit erzielten Ergebnisse beschreiben [Kos98a, Rig99, Rig98d] .

4.2.1 Die Transinformation als Gütemaß für das Training neuronaler Netze

Bei eingehender Betrachtung läßt sich bei Verwendung eines selbstorganisierenden Vektorquantisierers die Berechnung der Wahrscheinlichkeit für einen Merkmalsvektor $\vec{x}$ in einem HMM-Zustand s eines diskreten Systems aus der Wahrscheinlichkeit der VQ-Partition y_n im Zustand s ableiten:

$$p(\vec{x}|s) = p(y_n|s) \tag{4.3}$$

Das Codebuch des diskreten Systems besteht dabei aus N_C Einträgen y_n mit $n = 1, \cdots, N_C$. Der Vektorquantisierer ordnet einem Merkmalsvektor $\vec{x}$ denjenigen Index y_n des VQ-Prototypen zu, dessen Abstand zu dem Merkmalsvektor am geringsten ist (s. Abb. 4.1)
Im diskreten, wie im hybriden System wird durch den Vektorquantisierer bei Präsentation des Merkmalsvektors $\vec{x}$ ein VQ-Prototypindex y_n erzeugt: $\vec{x} \xrightarrow{VQ} y_n$. Der statistische Zusammenhang zwischen dem Merkmalsvektor $\vec{x}$ und dem VQ-Index y_n kann durch die bedingte Wahrscheinlichkeit $p(y_n|\vec{x})$ beschrieben werden. Mit Hilfe des Satzes von Bayes

$$p(\vec{x}, y_n) = p(\vec{x}) \cdot p(y_n|\vec{x}) = p(y_n) \cdot p(\vec{x}|y_n) \tag{4.4}$$

kann die Wahrscheinlichkeit $p(\vec{x})$ berechnet werden:

$$p(\vec{x}) = \frac{p(\vec{x}|y_n)}{p(y_n|\vec{x})} \cdot p(y_n). \tag{4.5}$$

Glg. (4.5) beschreibt somit den VQ-Prozeß, der unabhängig vom HMM-Zustand s ist. Fügt man nun die Bedingung eines bestimmten Zustandes s ein, ergibt sich für die bedingte Auftrittswahrscheinlichkeit:

$$p(\vec{x}|s) = \frac{p(\vec{x}|y_n)}{p(y_n|\vec{x})} \cdot p(y_n|s). \tag{4.6}$$

Mit Hilfe des nach $p(\vec{x}|y_n)$ umgestellten Satzes von Bayes (Glg. (4.4)) kann Glg. (4.6) schließlich umgeformt werden zu

$$p(\vec{x}|s) = \frac{p(\vec{x})}{p(y_n)} \cdot p(y_n|s). \tag{4.7}$$

Unter der Annahme, dass der Quotient $p(\vec{x})/p(y_n)$ nun bekannt ist läßt sich die Wahrscheinlichkeit des Merkmalsvektors $p(\vec{x})$ aus der Wahrscheinlichkeit $p(y_n)$ des VQ-Index y_n berechnen. Zur Verdeutlichung sei an dieser Stelle angemerkt, dass nach Glg. (4.3) in einem diskreten System die Identität der Wahrscheinlichkeiten $p(\vec{x})$ und $p(y_n)$ in Glg. (4.7) angenommen wird. Dies wiederum entspricht der - zweifellos idealisierten - Annahme einer verlustlosen Vektorquantisierung.

Ausgehend von Gleichung (4.7) kann nun ein Algorithmus abgeleitet werden, der zu einem optimierten Vektorquantisierer führt, wobei das Gesamtsystem (VQ *und* HMMs) die Vorteile diskreter und kontinuierlicher Modelle vereinen. Analog zur ansonsten üblichen Minimierung eines Fehlermaßes basiert das Training dieser Vektorquantisierer auf der Maximierung eines informationstheoretischen Gütemaßes. In der Gesamtheit ergibt sich somit ein hybrides HMM/NN-System.

Die Parameterschätzung von HMM-Systemen basiert zumeist auf dem sog. *Maximum Likelihood* (ML) Kriterium. Die Likelihood $P(X|S)$ eines Systems wird anhand seiner aktuellen Parameter λ (Ausgabeverteilungen in den HMM-Zuständen) über alle K Abtastwerte der Trainingsmenge aufsummiert:

$$p(X|S) = \sum_{k=1}^{K} \log p(\vec{x}(k)|s(k)). \tag{4.8}$$

Zur Berechnung bedient man sich aus numerischen Erwägungen i. d. R. der logarithmierten Form. Die erforderliche Zuordnung der einzelnen Merkmalsvektoren $\vec{x}(k)$ zu den entsprechenden HMM-Zuständen $s(k)$ kann z. B. automatisch mittels des *Viterbi-Alignments* bestimmt werden. Aus einem Parametersatz λ kann nun durch Anwendung von Gradientenverfahren in iterativer Weise ein optimierter Parametersatz $\hat{\lambda}$ berechnet werden:

$$\hat{\lambda} = \arg\max_{\lambda} \left\{ \frac{1}{K} \sum_{k=1}^{K} \log p(\vec{x}(k)|s(k)) \right\}. \tag{4.9}$$

Verwendet man weiterhin die statistischen Zusammenhänge des Vektorquantisierers aus Glg. (4.7) in Glg. (4.9), so können selbst die Parameter des Vektorquantisierers (Codebuchvektoren) λ_{VQ} in Übereinstimmung mit dem ML-Kriterium optimiert werden. Üblicherweise werden die Komponenten der Codebuchvektoren als *Gewichte* bezeichnet. Wird Glg. (4.7) in Glg. (4.9) eingesetzt, ergibt sich damit folgender Ausdruck für die Optimierung der Codebuchvektoren:

$$\hat{\lambda}_{VQ} = \arg\max_{\lambda_{VQ}} \frac{1}{K} \left\{ \sum_{k=1}^{K} \log p(\vec{x}(k)) - \sum_{k=1}^{K} \log p(y_n(k)) + \sum_{k=1}^{K} \log p(y_n(k)|s(k)) \right\}. \tag{4.10}$$

Als *Informationsgehalt* eines Zeichens bezeichnet man den negativen Logarithmus der Auftrittswahrscheinlichkeit dieses Zeichens. Die einzelnen Terme aus Glg. (4.10) sind die Erwartungswerte der entsprechenden Informationsgehalte und sind demnach auch bekannt unter dem Begriff der *Entropie* H. Damit läßt sich obige Gleichung umschreiben in

$$\hat{\lambda}_{VQ} = \arg\max_{\lambda_{VQ}}\{-H(X) + H(Y) - H(Y|S)\}. \tag{4.11}$$

Da die Merkmalsvektoren X auf der Eingangsseite des neuronalen Netzes anliegen, ist die Entropie $H(X)$ unabhängig von dem Codebuch des Vektorquantisierers, sodass folglich die Differenz $H(Y) - H(Y|S)$ zu maximieren ist. Das zu maximierende Gütemaß des neuronalen Vektorquantisierers ist damit die Transinformation I und lautet

$$I(Y,S) = H(Y) - H(Y|S) = H(S) - H(S|Y). \tag{4.12}$$

Aus Glg. (4.12) und anhand des informationstheoretischen Kanalmodells (Abb. 4.3) wird deutlich, dass die Maximierung von $I(S,Y)$ ebenso einer Maximierung von $H(S)-H(S|Y)$ entspricht. Da der Wert für $H(S)$ durch das Viterbi-Alignment der Trainingsdaten vorgegeben ist, reduziert sich die Maximierung der Transinformation $I(S,Y)$ deshalb auf die Minimierung der *Äquivokation* $H(S|Y)$.

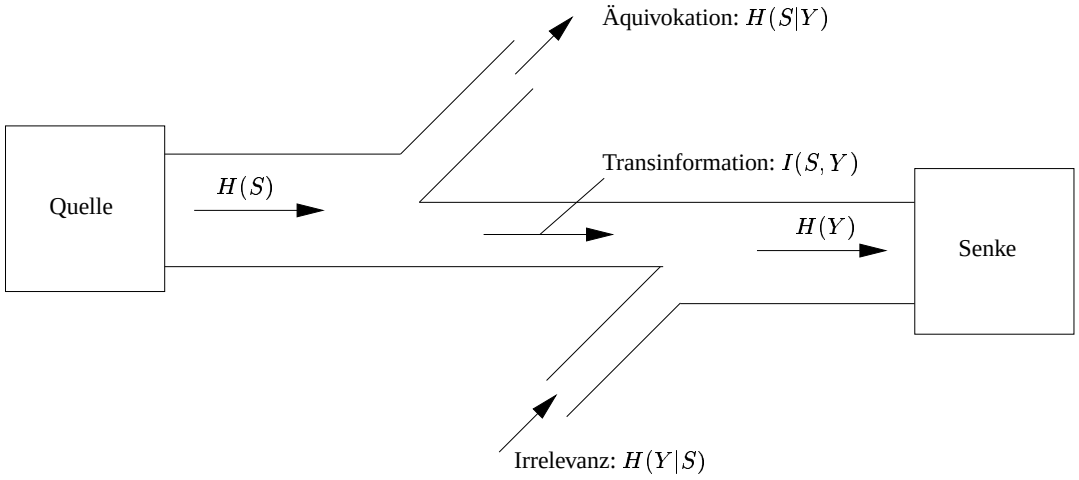

Abbildung 4.3: Informationstheoretisches Kanalmodell (Berger'sches Diagramm)

4.2.2 Automatische Handschrifterkennung - ein Nachrichtenübertragungssystem

Das Kanalmodell erlaubt darüber hinaus eine interessante systemorientierte Betrachtung der zuvor beschriebenen Theorie, die am Beispiel der Handschrifterkennung nachfolgend erläutert wird. Die wesentlichen Komponenten des Kanalmodells sind die Nachrichtenquelle, die Nachrichtensenke (Empfänger), und der Übertragungskanal, welcher Quelle und

Senke miteinander verbindet. Im Falle der Schrifterkennung repräsentiert die Quelle eine bestimmte Zeichen- bzw. Zustandsfolge S (basierend auf dem zu schreibenden Text), die von einem Schreiber generiert wird. Im Falle der Online-Handschrifterkennung werden diese geometrischen Signale dann durch eine Reihe von Transformationen in die VQ-Folge Y umgeformt. Im wesentlichen handelt es sich hierbei um die Vorverarbeitungsschritte, die Merkmalsextraktion und die Vektorquantisierung. Die Funktionen der Schrifterzeugung und der anschließenden Transformationen lassen sich entsprechend des Kanalmodells in Abb. 4.3 dem Übertragungskanal selbst zuordnen. Die Senke umfasst schließlich die Funktion des Erkenners, der aufgrund der empfangenen vektorquantisierten Daten mit Hilfe der Hidden Markov Modelle die Erkennung durchführt und die VQ-Daten wieder in eine Textform transformiert.

Ein Beobachter am Kanaleingang sieht demnach eine Nachrichtenquelle mit einem mittleren Informationsgehalt $H(S)$ (vergl. Abb. 4.3). Davon geht - bedingt durch verschiedene Verarbeitungsstufen - der Anteil $H(S|Y)$ verloren. Die bedingte Entropie $H(S|Y)$ ist der mittlere Informationsgehalt je Zeichen, den das Zeichen, bzw. der Zustand s noch zusätzlich liefern würde, nachdem y empfangen wurde. Da der Empfänger jedoch nur Zugriff auf die Zeichen Y hat, ist - wie bereits oben erwähnt - $H(S|Y)$ der Verlustanteil, der in erster Linie durch den Quantisierungsfehler entsteht. Da die übrigen Vorverarbeitungsstufen ohnehin fest vorgegeben sind, reduziert sich die Optimierung auf die Minimierung dieses Informationsverlustes. Die Transinformation $I(S,Y)$ ist der verbleibende Teil der Quelleninformation, der über den Kanal übertragen wird und den Empfänger erreicht. Ggf. wird der Transinformation bei der Übertragung noch eine gewisse Fehlinformation $H(Y|S)$ hinzugefügt (z. B. durch Kanalstörungen, Rauschen), die in dem mittleren Informationsgehalt $H(Y)$ resultiert. Der Fehlinformationsanteil $H(Y|S)$ stammt also nicht aus der Nachrichtenquelle, sondern ist der mittlere Informationsanteil, den ein Zeichen y noch liefern würde, nachdem ein gesendetes Zeichen s bekannt ist. Dem Beobachter am Kanalausgang erscheint schließlich eine Nachrichtenquelle mit der Entropie $H(Y)$. Die Aufgabe des Erkenners ist es nun, aufgrund der empfangenen Daten Y einen (möglichst) fehlerfreien Rückschluß auf die gesendeten Daten (Zustands- bzw. Graphemfolge S) zu ziehen.

Realisierung verteilter Architekturen

Das hier vorgestellte Verfahren ermöglicht die Realisierung komplexer Mensch-Maschine-Schnittstellen (Handschrift- oder auch Spracherkennung) auf Plattformen mit geringer Rechenleistung und relativ schmalbandiger Netz-Anbindung (Mobiltelefone, PDA) [Rig00]. Dies läßt sich durch eine Übertragung der relevanten Merkmale in hoch-komprimierter Form und einer Erkennung dieser komprimierten Daten auf CPU-Servern realisieren, die z. B. ein Mobilfunkanbieter bereithält. Das Erkennungsergebnis wird anschließend wieder zum Endgerät zurückgesendet und kann dort je nach gewünschter Anwendung als Befehl

interpretiert oder als Text-Passage weiter verarbeitet werden.
Wenn die rechenaufwendige Erkennung auf dem Endgerät nicht möglich ist, bietet sich die Auslagerung des eigentlichen Erkennungsprozesses an. Dies ließe sich über die Komprimierung der Merkmalsvektoren auf dem Endgerät und die Übertragung dieser Merkmale an einen leistungsfähigen Rechner ermöglichen, auf dem der eigentliche Erkennungsprozeß stattfindet. Zum Zweck der Merkmalskompression können Vektorquantisierer eingesetzt werden, die einen vieldimensionalen kontinuierlichen Merkmalsvektor auf eindimensionale diskrete Werte abbilden und so eine sehr hohe Kompressionsrate erzielen. Dies wiederum ermöglicht selbst die Nutzung schmalbandigster Übertragungskanäle. Abb. 4.4 gibt einen

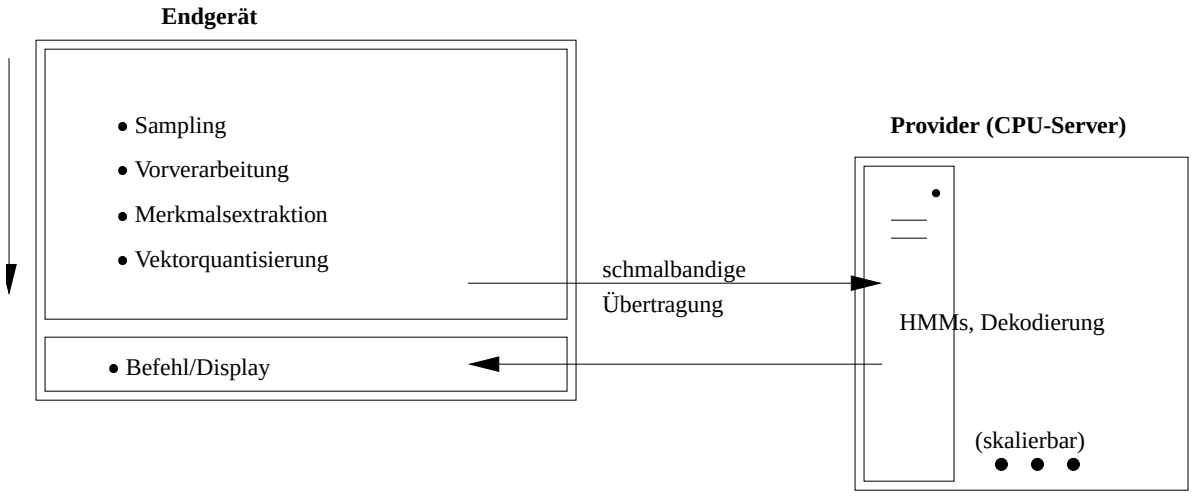

Abbildung 4.4: Übersicht eines verteilten Erkennungssystems

Überblick über eine mögliche verteilte Systemarchitektur. Auf der Benutzerseite werden dabei sämtliche vorbereitende Operationen durchgeführt, wie die Signalaufnahme, Vorverarbeitung, Merkmalsextraktion und Vektorquantisierung. Die so gewonnenen komprimierten Merkmale werden anschließend mit geringer Bandbreite an ein skalierbares System von CPU-Servern gesendet, auf denen der eigentliche, rechenaufwendige Erkennungsprozeß stattfindet. Das Erkennungsergebnis kann anschließend zum Benutzer übertragen werden, auf dessen Endgerät das Ergebnis dann als Textpassage oder als Befehl verwendet werden kann.
Ein wesentlicher Aspekt des hier beschriebenen hybriden Ansatzes ist der Zusammenhang zwischen der Vektorquantisierung und der Erkennung mittels Hidden Markov Modellen, da dies genau die Schnittstelle zwischen Endgerät und Erkenner in Abb. 4.4 (bzw. zwischen Sender und Empfänger in Abb. 4.3) darstellt. Dieser Zusammenhang soll im nachfolgenden Abschnitt beschrieben werden.

4.3 Hybrides NN/HMM Handschrifterkennungssystem

Im Detail läßt der hier vorgestellte theoretische Rahmen zahlreiche Realisierungsformen zu. So ist es durch entsprechende Modifikationen möglich, einen MMI-Vektorquantisierer mit diversen neuronalen Paradigmen zu realisieren [Neu99]. Hierbei kommen neben dem *nächster-Nachbar*-VQ (nN-VQ) auch Strukturen wie das MLP (*Multi-Layer-Perzeptron*), RBF- (Radiale Basisfunktionen-) Netze oder rekurrente Netze in Frage. Gemeinsam ist den verschiedenen Paradigmen eine generelle Arbeitsweise. Es wird ein Merkmalsvektor $\vec{x}(k)$ - ggf. zusammen mit einem gewissen zeitlichen Kontext $\vec{x}(k-k'),\cdots,\vec{x}(k),\cdots\vec{x}(k+k')$ an den Eingang des Netzes angelegt. Ausgangsseitig erzeugt das Netz mittels einer *winner-takes-all*-Schicht (WTA) einen diskreten VQ-Label y_n mit

$$y_n = y(\vec{x}) \tag{4.13}$$

und

$$n = \arg\max_i f_i(\vec{x}). \tag{4.14}$$

$f_i(\vec{x})$ ist dabei die interne Aktivierung des Ausgangsknotens i. Des weiteren ist es möglich einen Merkmalsvektor $\vec{x}$ in Z verschiedene Untervektoren $\vec{x}^{(z)}$ zu unterteilen. Bei dieser sog. Multi-Codebuchtechnik, wie sie in Abschnitt 4.3.2 beschrieben wird, können dann die einzelnen Codebücher simultan nach dem MMI-Prinzip trainiert werden, sodass das Training in einer maximierten Verbund-Transinformation $I(Y^{(1)},\cdots,Y^{(z)},S)$ resultiert.
Ähnlich viele Freiheitsgrade hinsichtlich des System-Designs ergeben sich bei der Auswahl der Klassen. Während bisher davon ausgegangen wurde, dass das Viterbi-Alignment die Trainingsdaten entsprechend den HMM-Zuständen s segmentiert, besteht ebenso die Möglichkeit die Klassenzugehörigkeiten entsprechend den Graphemgrenzen (Mono-, Bi- oder Trigrapheme) oder aber auch entsprechend der Wortgrenzen festzulegen. In der Praxis könnten so z. B. wortbasierte HMMs eingesetzt werden, die speziell bei Erkennungssystemen mit kleinen Wortschätzen Vorteile bieten. Außerdem ist hier zu unterstreichen, dass die Codebuchgröße unabhängig von der Anzahl der Klassen (Zustände, Grapheme, etc.) gewählt werden kann.
Ein allgemeines Problem bei Gradientenabstiegsverfahren stellen mögliche lokale Nebenminima, sowie flache Plateaus im Funktionsverlauf dar, die bei der Parameteroptimierung zu suboptimalen Lösungen führen können. Als Lösung bieten sich dazu aus der Neuroinformatik bekannte Verfahren an, wie z. B. *Momentum*, oder *RProp* (resilient propagation) [Zel94]. Bei den durchgeführten Experimenten zeigten speziell die RProp-Verfahren eine schnelle und gute Konvergenz.

4.3.1 MMI-Training einzelner Codebücher

Die grundlegende Idee des Trainings des neuronalen Netzes, bzw. die Optimierung der Gewichtungsvektoren, ist die Minimierung einer Bewertungsfunktion (Äquivokation $H(S|Y)$) mittels Gradientenabstieg [Neu99]. Die bedingte Entropie $H(S|Y)$ ist definiert mit

$$H(S|Y) = -\sum_{l=1}^{L}\sum_{n=1}^{N} p(s_l, y_n) \cdot \log p(s_l|y_n) \tag{4.15}$$

und läßt sich als Funktion von $p(s, y)$ umschreiben:

$$H(S|Y) = -\sum_{l=1}^{L}\sum_{n=1}^{N} p(s_l, y_n) \cdot \log \frac{p(s_l, y_n)}{\sum_{m=1}^{L} p(s_m, y_n)}. \tag{4.16}$$

Aus den Zustands-weise segmentierten Trainingsdaten läßt sich wiederum die Verbundwahrscheinlichkeit $p(s_l, y_n)$ abschätzen

$$p(s_l, y_n) \approx \frac{1}{K} \cdot \sum_{k=1}^{K} \delta_{s(k),s_l} \cdot o_n(k), \tag{4.17}$$

mit dem Kronecker-Delta δ. Da die WTA-Funktion aus Glg. (4.14), angewandt auf die Aktivierung der Ausgangsknoten nicht stetig differenzierbar ist, wird eine Soft-Max Funktion o eingeführt, welche die Maximumsuche unter den Ausgangsaktivierungen $f_n(\vec{x})$ aus Glg. (4.14) annähert durch

$$o_n(\vec{x}) = \frac{e^{C \cdot f_n(\vec{x})}}{\sum_{i=1}^{N} e^{C \cdot f_i(\vec{x})}}. \tag{4.18}$$

Bei hinreichend großer Wahl der Konstante C, kann nun die WTA-Funktion durch die Soft-Max-Funktion approximiert werden:

$$\lim_{C\to\infty} o_n(\vec{x}) = \begin{cases} 1 & \text{wenn } f_n(\vec{x}) > f_i(\vec{x}) \quad \forall \ 1 \le i \le N \ , \ n \ne j \\ 0 & \text{sonst} \end{cases} \tag{4.19}$$

Für die Einstellung eines bestimmten VQ-Parameters λ_{VQ} wird nachfolgend die partielle Ableitung $\partial H(S|Y)/\partial\lambda_{VQ}$ betrachtet. Unter Anwendung der verallgemeinerten Kettenregel ergibt sich damit

$$\frac{\partial H(S|Y)}{\partial \lambda_{VQ}} = \sum_{l=1}^{L}\sum_{n=1}^{N} \frac{\partial H(S|Y)}{\partial p(s_l, y_n)} \cdot \sum_{k=1}^{K} \frac{\partial p(s_l, y_n)}{\partial o_n(\vec{x}(k))} \cdot \sum_{i=1}^{N} \frac{\partial o_n(\vec{x}(k))}{\partial f_i(\vec{x}(k))} \cdot \frac{\partial f_i(\vec{x}(k))}{\partial \lambda_{VQ}}. \tag{4.20}$$

Nach der Bildung der partiellen Differentiale und einigen Umformungen ergibt sich mit der Einführung einer Hilfsfunktion h_i mit

$$h_i(\vec{x}(k)) = -\frac{C}{K} \cdot o_n(\vec{x}(k)) \cdot \left(\log p(s(k)|y_n) - \sum_{n=1}^{N} \log p(s(k)|y_n) \cdot o_n(\vec{x}(k)) \right), \tag{4.21}$$

der folgende Ausdruck für die Ableitung der Äquivokation:

$$\frac{\partial H(S|Y)}{\partial \lambda_{VQ}} = \sum_{k=1}^{K}\sum_{i=1}^{N} \frac{\partial f_i(\vec{x}(k))}{\partial \lambda_{VQ}} \cdot h_i(\vec{x}(k)). \tag{4.22}$$

Glg. (4.22) stellt nun eine allgemeingültige Berechnungsvorschrift für die Adjustierung des VQ-Parameters λ_{VQ} dar, die unabhängig von der Struktur des neuronalen Netzes verwendet werden kann. Unter Einbeziehung des Lernparameters β, mit welchem sich die Stabilität und die Konvergenz des Trainings kontrollieren lässt, kann mit Hilfe von Glg. 4.22 das Gewicht λ_{VQ} um $\Delta\lambda_{VQ}$ korrigiert werden:

$$\Delta\lambda_{VQ} = -\beta\frac{\partial H(S|Y)}{\partial\lambda_{VQ}} \tag{4.23}$$

Die bedingten Wahrscheinlichkeiten $p(s(k)|y_n)$ sind wiederum aus der gegebenen Trainingsdatenbasis zu schätzen.
Unter Verwendung eines nN-Vektorquantisierers ist für die Aktivierung f_i der Euklid'sche Abstand des J-dimensionalen Merkmalsvektors $\vec{x}$ zum nächsten VQ-Prototypenvektor $\vec{\lambda}_i$ einzusetzen:

$$f_i(\vec{x}) = \|\vec{x} - \vec{\lambda}_i\|_2^2 = \sum_{j=1}^{J}(x_j - \lambda_{VQ_{i,j}})^2. \tag{4.24}$$

4.3.2 Simultanes MMI-Training multipler Codebücher

Die Quantisierung eines reellwertigen, vieldimensionalen Vektors, auf einen skalaren, ganzzahligen Wert ist in der Regel ein mehr oder minder stark verlustbehafteter Prozeß. Diese Aussage gilt natürlich ebenso für MMI-trainierte VQ. Auch hier ist der Informationsverlust nicht vollständig aufzufangen. Neben dem MMI-Training besteht eine weitere einfache Möglichkeit den quantisierungsbedingten Verlust zu minimieren, indem das Codebuch vergrößert wird. Diese Möglichkeit der Verlustminimierung ist allerdings nur bis zu einem gewissen Grade möglich, da ab einer bestimmten Codebuchgröße die Parameter nicht mehr zuverlässig geschätzt werden können. Dieses *sparse-data*-Problem betrifft zum einen die VQ-Parameter, insbesondere aber die diskreten Wahrscheinlichkeitsdichtefunktionen (WDF) der HMM-Zustände, da diese WDF bei einer Codebuchgröße von J allgemein auch J zu schätzende Parameter umfassen. Bei einer vorgegebenen finiten Trainingsmenge muß also stets die Trainierbarkeit der Paramater gewährleistet bleiben. Ein günstiger Kompromiß zwischen Auflösungsvermögen des VQ und Trainierbarkeit sämtlicher Parameter kann nur auf experimentelle Weise gefunden werden.
Wird die Tatsache berücksichtigt, dass verschiedene Teile eines Merkmalsvektors in vielen Fällen aus verschiedenen Merkmalsextraktionsverfahren gewonnen werden, ergibt sich eine weitere Option zur Verbesserung der Auflösung der Vektorquatisierung. Diese wird durch eine Aufteilung des Merkmalsvektors $\vec{x}$ in Z verschiedene Untervektoren $\vec{x}^{(z)}$ erreicht, wobei jeder Untervektor $\vec{x}^{(z)}$ von einem bestimmten Vektorquantisierer $VQ^{(z)}$ mit moderater Codebuchgröße $J^{(z)}$ komprimiert wird. Damit läßt sich zum einen die Parameteranzahl der einzelnen WDF auf ein sinnvolles Maß begrenzen ($\sum_{z=1}^{Z} J^{(z)}$), zum anderen aber noch immer

eine hinreichend gute Auflösung mit theoretisch bis zu $\prod_{z=1}^{Z} J^{(z)}$ verschiedenen Partitionskombinationen erzielen. Mit den aus Kapitel 3 gewonnenen Erkenntnissen bietet sich eine Aufteilung in Trajektorien- und Bitmap-Merkmalen an. Eine Hinzunahme weiterer Merkmalsströme scheint aus derzeitiger Sicht wenig gewinnbringend, womit die folgenden Betrachtungen für $Z = 2$ gelten. Eine Verallgemeinerung auf weitere Merkmalsströme ist in analoger Weise möglich.
Idealerweise sollte die Multi-Stream-Technik so angewendet werden, dass die Aufteilung zu einer statistischen Unabhängigkeit der Teilvektoren $\vec{x}^{(z)}$ untereinander führt. Eine solche Annahme jedoch global zu treffen ist sicherlich nicht sehr realistisch. Bei einer sinnvollen Aufteilung in Untervektoren kann hingegen als Näherung die Annahme der lokalen statistischen Unabhängigkeit getroffen werden, wie sie auch bei der HMM-Parameterschätzung für Multi-Stream-Modelle genutzt wird. D. h., dass zumindest die Ausschnitte eines Streams als statistisch unabhängig angesehen werden, die einem bestimmten Zustand s zugeordnet werden. Die bedingte Auftrittswahrscheinlichkeit wird dann als Produkt der bedingten Einzelwahrscheinlichkeiten berechnet:

$$p(y(\vec{x})|s) = p(y^{(1)}(\vec{x}^{(1)}), \cdots, y^{(Z)}(\vec{x}^{(Z)})|s) = p(y^{(1)}(\vec{x}^{(1)})|s) \cdot p(y^{(2)}(\vec{x}^{(2)})|s). \tag{4.25}$$

Im Idealfall - bei globaler statistischer Unabhängigkeit - können somit die beiden VQ einzeln, d. h. unabhängig voneinander trainiert werden. Die Gesamt-Transinformation I_{ges} ist dann die Summe der Transinformationswerte der Einzel-VQ:

$$I_{ges} = I(Y^{(1)}, S) + I(Y^{(2)}, S). \tag{4.26}$$

Diese globale statistische Unabhängigkeit ist jedoch in starkem Maße idealisiert: in der Praxis sind stets Korrelationen zwischen verschiedenen Merkmalen festzustellen. Dies ist selbst bei Trajektorien- und Bitmap-Merkmalen der Fall. Um optimale Ergebnisse zu erzielen, sollte die Optimierung des einen VQ unter Berücksichtigung der übrigen VQ erfolgen. Dazu kann zunächst ausgehend vom Kanalmodell und Glg. (4.12), übertragen auf die Multi-Stream-Technik, die Gesamt-Transinformation I_{ges} als Gütemaß wie folgt formuliert werden:

$$I_{ges} = I(Y^{(1)}, Y^{(2)}, S) = H(Y^{(1)}, Y^{(2)}) - H(Y^{(1)}, Y^{(2)}|S). \tag{4.27}$$

Basierend auf der Annahme einer lokalen statistischen Unabhängigkeit (s. Glg. (4.25)), kann die Berechnung des zweiten Terms aus Glg. (4.27) auf die Summation der bedingten Einzelentropien zurückgeführt werden:

$$I_{ges} = H(Y^{(1)}, Y^{(2)}) - (H(Y^{(1)}|S) + H(Y^{(2)}|S)). \tag{4.28}$$

Erweitert man den Ausdruck um $\mp \sum_{z=1}^{2} H(Y^{(z)})$, ergibt sich

$$\begin{aligned} I_{ges} = \quad & -\left(\underbrace{\left\{ (H(Y^{(1)}) + H(Y^{(2)})) \right\} - H(Y^{(1)}, Y^{(2)})}_{I(Y^{(1)}, Y^{(2)})} \right) \\ & + \underbrace{H(Y^{(1)}) - H(Y^{(1)}|S)}_{I(Y^{(1)}, S)} + \underbrace{H(Y^{(2)}) - H(Y^{(2)}|S)}_{I(Y^{(2)}, S)}, \end{aligned} \tag{4.29}$$

was schließlich zu der verwendeten Bewertungsfunktion I_{ges} führt:

$$I_{ges} = -I(Y^{(1)}, Y^{(2)}) + I(Y^{(1)}, S) + I(Y^{(2)}, S) \tag{4.30}$$

Verglichen mit der idealisierten Funktion für I_{ges} aus Glg. (4.26), wird die Transinformationssumme der einzelnen Ströme um den Summanden $-I(Y^{(1)}, Y^{(2)})$ korrigiert. Dieser Korrekturterm beschreibt nun genau die Korrelation der einzelnen diskreten Merkmalsströme untereinander und kann daher auch als Maß für die Redundanz der Merkmalströme interpretiert werden. Für die Praxis bedeutet dies, dass die Aufteilung der Untervektoren und deren statistische Unabhängigkeit nicht mehr als kritisch für die spätere Erkennungsleistung gelten muß. Dies wird dadurch erreicht, dass neben der Maximierung der Transinformation einzelner Merkmalsströme die Dekorrelation der Merkmale als zusätzliches Bewertungskriterium in den Trainingsprozeß einfließt.

Die Maximierung von $I(Y^{(z)}, S)$ der einzelnen VQ erfolgt bei simultanem Training analog zu dem separaten MMI-Training aus Glg. (4.21) und Glg. (4.22). Entsprechend Glg. (4.30) ist zur Maximierung des Gesamtausdrucks noch die Minimierung der statistischen VQ-Abhängigkeiten nötig. Mit folgender Gleichung kann die Transinformation der statistischen VQ-Abhängigkeiten ausgedrückt werden:

$$I(Y^{(1)}, Y^{(2)}) = \sum_{j_1=1}^{J_1} \sum_{j_2=1}^{J_2} p(y_{j1}, y_{j2}) \cdot \log \frac{\bar{p}(y_{j1}, y_{j2})}{p(y_{j1}) \cdot p(y_{j2})} \tag{4.31}$$

Entsprechend der Jensen-Ungleichung [ST95] läßt sich die Transinformation der VQ-Abhängigkeiten $I(Y^{(1)}, Y^{(2)})$ aus den WDF der Modelle $\bar{p}$ und den wahren WDF p berechnen, welche approximativ aus den Trainingsdaten ermittelt werden. Ähnlich wie im vorangegangenen Abschnitt kann nun unter Anwendung der Kettenregel die partielle Ableitung von $I(Y^{(1)}, Y^{(2)})$ nach einem bestimmten VQ-Gewicht λ_{VQ} des z-ten VQ bestimmt werden:

$$\frac{\partial I(Y^{(1)}, Y^{(2)})}{\partial \lambda_{VQ}} = \sum_{k=1}^{K} \sum_{i=1}^{J_z} \frac{\partial f_i(\vec{x}(k))}{\partial \lambda_{VQ}} \cdot h_i(\vec{x}(k)). \tag{4.32}$$

Mit der Aktivierungsfunktion f_i entsprechend Glg. (4.24) und der Hilfsfunktion h_i, mit

$$\begin{aligned} h_i^{(z)}(\vec{x}(k)) = \quad & \tfrac{C}{T} \cdot o_n^{(z)}(\vec{x}(k)) \cdot \\ & \cdot \sum_{j=1}^{J_z} \left(\tfrac{\partial I(Y^{(1)}, Y^{(2)})}{\partial p(y_1(k), \cdots, y_j^{(z)}, \cdots, y^{(Z)}(k))} + \tfrac{\partial I(Y^{(1)}, Y^{(2)})}{\partial p(s(k), y_j^{(z)})} \right) \cdot (\delta_{i,j} - o_j^{(z)}(\vec{x}(k))) \end{aligned} \tag{4.33}$$

läßt sich die Berechnungsvorschrift entsprechend Glg. (4.32) für die Einstellung der Gewichte angeben.

4.4 Ergebnisse

Die Ergebnisse des Trainings eines Multi-Stream-Systems mit zwei Merkmalsströmen $Y^{(1)}$ und $Y^{(2)}$ sind mit dem entsprechenden Transinformationsverlauf in Abb. 4.5 dargestellt. In der Regel wird ein solches Training in drei Stufen durchgeführt. Die erste Stufe dient der Initialisierung des Netzes. Diese Initialisierung wird hier bei Verwendung Euklid'scher Abstandsmaße als Aktivierungsfunktionen mit einem k-means Algorithmus durchgeführt.
In der zweiten Stufe setzt dann auf diesem k-means-VQ das MMI-Training für einzelne Codebücher auf, wie es in Abschnitt 4.3.1 beschrieben wurde. In Abb. 4.5 entspricht dies den Iterationen 1 bis 10. In dieser zweiten Trainingsstufe (separates Training) ist der Anstieg der Transinformation deutlich für jedes der eingesetzten Codebücher zu sehen.
Anschließend wird in der dritten Stufe ein simultanes Training der beiden Codebücher durchgeführt (Iterationen 11-20). In dieser Stufe schließlich, wird als Bewertungsfunktion die Gesamt-Transinformation $I_{ges} = I(Y^{(1)}, Y^{(2)}, S)$ verwendet. Bei diesem simultanen Training wird zwecks Maximierung von I_{ges} eine leichte Steigerung von $I(Y^{(1)}, S)$ erreicht. Die Transinformation des zweiten Codebuches $I(Y^{(2)}, S)$ wird hingegen sogar verringert. Dies geschieht zu Gunsten der Dekorrelation der einzelnen Merkmale. Die Korrelation ist abzulesen an dem Verlauf der Transinformation der VQ-Abhängigkeiten $I(Y^{(1)}, Y^{(2)})$. Insgesamt ist jedoch ein deutlicher Anstieg der Gesamt-Transinformation I_{ges} zu erkennen.

Prinzipiell könnte die zweite Stufe dieses Verfahrens (separates Training) durch eine verlängerte dritte Trainingsstufe (simultanes Training) ersetzt werden, was durchaus zu ähnlichen Ergebnissen führt. Da die dritte Stufe jedoch relativ rechenzeitaufwendig ist stellt sich die geschilderte Variante mit k-means, separatem und simultanem Training als sehr praktikabel dar.

Tab. 4.1 zeigt zunächst die Ergebnisse eines schreiberabhängigen Systems mit einem Vokabular bestehend aus 30000 Wörtern. Die hier gezeigten Ergebnisse wurden mit einem diskreten System erzielt. Die einzelnen Zeilen zeigen die Ergebnisse jedes Schreibers ('ank', 'jmr' und 'vdm') mit den entsprechenden Durchschnittsergebnissen für verschiedene Merkmals-

	ank	jmr	vdm	ø
BM:	95.2 %	84.5 %	95.2 %	91.6 %
$[\alpha, \tilde{p}]$:	90.3 %	85.5 %	70.7 %	82.2 %
alle:	93.0 %	77.9 %	83.4 %	84.8 %

Tabelle 4.1: Erkennungsergebnis - diskretes System, schreiberabhängig, 30K Lexikon

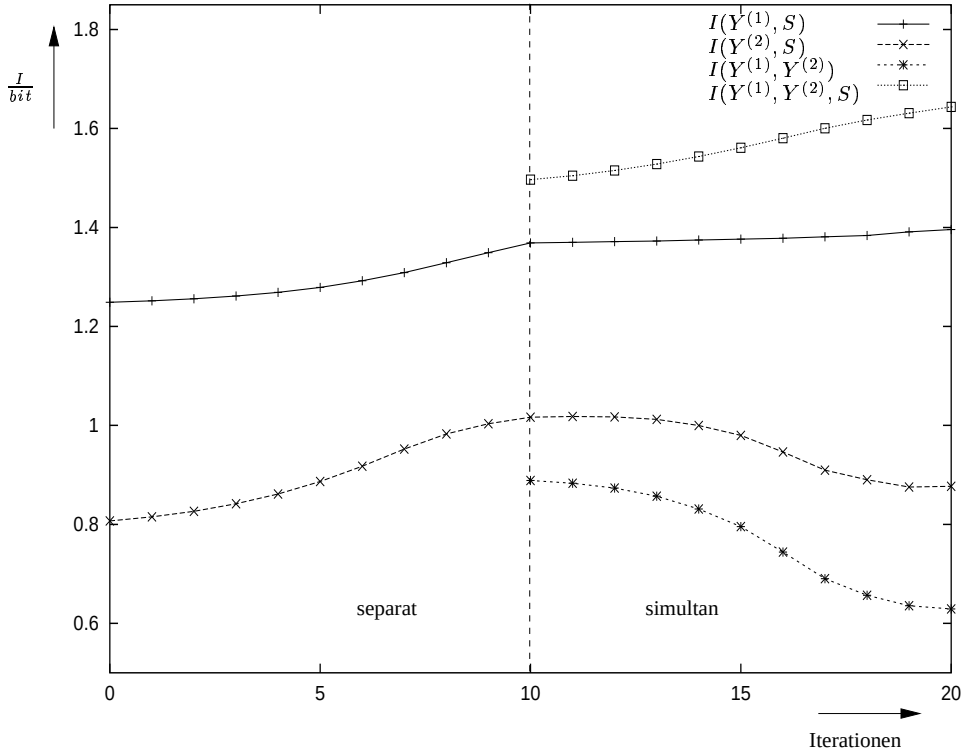

Abbildung 4.5: Verlauf der Transinformation für separates und simultanes MMI-Training

kombinationen. Die Kettencodierung (Zeile 2) weist hierbei deutlich schlechtere Resultate auf, als die gleitende Bitmap (Zeile 1). Erstaunlich in dem schreiberabhängigen Fall ist, dass die Merkmalskombination auch eher kontraproduktiv wirkt. Bei den folgenden Diskussionen soll daher der Fokus auf der Bitmap und auf den kombinierten Merkmalen liegen.

Tab. 4.2 zeigt die Ergebnisse der untersuchten Paradigmen. Zeile 1 und 3 zeigen die Ergebnisse für die Bitmap als Einzelmerkmal. Die übrigen Zeilen stellen die Ergebnisse kombinierter Merkmale dar. Während mit dem diskreten System unter Verwendung der Bitmap als Einzelmerkmal noch 91,6 % Korrektheit erzielt wurden (Tab. 4.1), konnten mit kontinuierlichen Modellen bereits 92,0 % erzielt werden. Das hybride System hingegen lieferte bei gleichen Bedingungen durchschnittlich 95,2 %. Die Merkmalskombination wirkt sich bei kontinuierlichen Systemen mit 94,6 %, wie hybriden Systemen mit 94,3 % wiederum uneinheitlich aus. Bei dem kontinuierlichen System liegt der Vorteil darin begründet, dass sämtliche Merkmale in einem Merkmalsstrom zusammengefasst werden, und so statistische Unabhängigkeitsannahmen weniger kritisch sind. Dieser Nachteil kann schließlich mit dem simultanen MMI Training von Bitmap und α-Merkmal aufgeholt werden. Es ergibt sich schließlich eine Erkennungsrate von 96,8 %, was einer relativen Fehlerreduktion verglichen mit dem diskreten Basissystem (Tab. 4.1, Zeile 3) um 79 % entspricht. Diese Experimente werden anschließend unter Verwendung eines Lexikons mit 200000 Wörtern wiederholt (s. Tab. 4.3). Die Ergebnisse des kontinuierlichen Systems und des hybriden Systems liegen zunächst wieder relativ dicht beieinander. Es zeigt sich jedoch auch hier eine Überlegenheit

des simultanen MMI-Trainings mit 91,3 % Korrektheit gegenüber der kontinuierlichen Modellierung oder dem separaten MMI-Training. Im Vergleich zu dem kontinuierlichen System mit 87,7 % bedeutet dies eine relative Fehlerreduktion um immerhin 29 %. Die Ergebnisse des schreiberabhängigen Systems sind bereits optimiert bezüglich Codebuchgröße bzw. Anzahl der Mischverteilungen und bezüglich der Kontextgröße.
Diese Erfahrungswerte werden nun wiederum eingesetzt um das schreiberunabhängige System zu evaluieren. Mit dieser näherungsweise optimalen Parameteranzahl wird für das diskrete WI-System bei Verwendung eines 2000 Wörter umfassenden Lexikon unter Nutzung kombinierter Merkmale eine Erkennungsrate von 86,9 % erreicht. Wird das α-Codebuch konstant gehalten und nur das Bitmap-Codebuch MMI-trainiert, verbessert sich das Ergebnis auf 88,9 %. Bei einem separaten MMI-Training beider Codebücher läßt sich dieser Wert auf 89,7 % steigern. Eine letztmalige Steigerung ist mit dem simultanen MMI-Training beider Codebücher (dritte Stufe) möglich. Hierbei wird schließlich eine Erkennungsrate von 90,6 % möglich.

4.5 Kapitelzusammenfassung

Wie in den vorangegangenen Abschnitten gezeigt wurde, minimiert ein MMI- (*maximum mutual information*-) Training des Vektorquantisierers die Äquivokation, die ihrerseits im wesentlichen von den Informationsverlusten der Vektorquantisierung herrührt. Gleichzeitig wird das Gesamtsystem (VQ und HMMs) aber geschlossen nach dem ML-Verfahren trainiert, wobei in das Training des Vektorquantisierers zusätzlich die Klasseninformation der einzelnen Trainingsvektoren einfließt. Das MMI-Training ist sowohl für verschiedene neuronale Paradigmen mit verschiedenen Aktivierungsfunktionen anwendbar, wie auch für das simultane Training mehrerer VQ in einem Multi-Stream-System, bei dem verschiedene Merkmalstypen über verschiedene Codebücher in unabhängigen Merkmalströmen modelliert werden.
Abschließend läßt sich zusammenfassen, dass mit dem hier beschriebenen Verfahren die entscheidenden Nachteile eines diskreten Systems verglichen mit einem kontinuierlichen Sys-

	ank	jmr	vdm	ø	
BM:	95.7 %	88.2 %	92.2 %	92.0 %	kontinuierlich
alle:	96.8 %	91.4 %	95.7 %	94.6 %	
BM:	96.8 %	87.7 %	95.7 %	95.2 %	MMI (separat)
alle:	96.2 %	89.8 %	96.8 %	94.3 %	
alle:	97.3 %	94.1 %	98.9 %	96.8 %	MMI (simultan)

Tabelle 4.2: Erkennungsergebnis - schreiberabhängig, 30K Lexikon

	ank	jmr	vdm	ø	
alle:	94.1 %	77.5 %	91.4 %	87.7 %	kontin.
BM:	91.4 %	79.3 %	94.2 %	88.3 %	MMI (separat)
alle:	97.3 %	83.4 %	93.1 %	91.3 %	MMI (simultan)

Tabelle 4.3: Erkennungsergebnis - schreiberabhängig, 200K Lexikon

tem (s. Abschnitt 4.1) eliminiert werden konnten. Neben den sehr guten Erkennungsraten ergeben sich mit der Verwendung hybrider Modelle einige weitere wesentliche Vorteile:

- Es bietet sich mit dem hybriden Ansatz die Möglichkeit zur Realisierung verteilter Systeme. Bei geringer Bitrate können die quantisierten Merkmale ohne Verluste der Erkennungsgenauigkeit zum Erkenner übertragen werden.
- Die für das MMI-Training zu definierenden Klassen können Zustände, Grapheme (Bi- und Trigrapheme) oder - im Fall von Erkennungssystemen mit kleinem Vokabular - Worte sein.
- Die Unabhängigkeit von der Klassendefinition führt dazu, dass die Anzahl der Ausgangsknoten des VQ unabhängig von der Klassenanzahl (Anzahl der Zustände, Grapheme oder Worte) ist. Die Parameteranzahl der Codebücher und HMM-Verteilungen läßt sich also unabhängig von den gewählten Klassen optimieren.
- Es bedarf keiner Annahme über die Art der Verteilung der verwendeten Merkmale. Die Merkmale können so auch nicht Gauß-verteilt oder sogar diskreter Natur sein.
- Die Berechnung der Wahrscheinlichkeiten aus den diskreten HMM-Verteilungen läßt sich durch *table look-up* effizient gestalten.

Die überlegene Erkennungsleistung mit den weiteren genannten Vorteilen legt daher die Anwendung hybrider Modelle, d. h. die Kombination MMI-trainierter neuronaler Netze für die Vektorquantisierung mit HMM nahe.