Hidden Markov Models

Nikolas Dörfler

21.11.2003

Hauptseminar Machine Learning

1 Einleitung

Nicht alle Vorgänge laufen stehts in einer festen deterministischen Reihenfolge ab und sind somit relativ einfach zu beschreiben. Meist folgt ein System zwar grundlegenden Regeln, aber es verhält sich nicht deterministisch. Auch gibt es Vorgänge die nicht direkt beobachtet werden können sondern von denen wir nur Signale erfassen und auswerten können. Diese müssen dann ausreichen, um Rückschlüsse auf das wahre Geschehen zu ziehen. Zur Beschreibung solcher im Verborgenen ablaufenden nichtdeterministischer Vorgänge bieten sich statistische Modellierungen wie die der Hidden Markov Models an. Hidden Markov Models werden heute bei verschiedensten Problemstellungen, wie z.B. Spracherkennung, Handschriftenerkennung, in der Biologie zur Klassifzierung von Proteinsequenzen und zur Modellierung von Wirtschaftlichen Vorgängen eingesetzt. Im Folgenden sollen die Theorie und grundlegenden Problemstellungen dargestellt werden. Als Anwendungsbeispiel wird kurz auf Spracherkennung unter Verwendung von Hidden Markov Models eingegangen.

2 Markov Models

Wenn wir ein System beobachten, wollen wir normalerweise wissen in welchem Zustand es sich zum Zeitpunkt der Beobachtung befindet. Beispielsweise könnte uns interessieren wie das Wetter heute ist. Interessant wäre auch, zu wissen wie das Wetter am nächsten Tag sein wird, oder zumindest, wie es warscheinlich sein wird. Ein einfaches Modell hierzu könnte folgendermaßen aussehen: Es gibt drei Zustände, Sonne, Wolken, und Regen. Die Abfolge dieser Zustände ist natürlich nicht fest vorgegeben sondern weitgehend

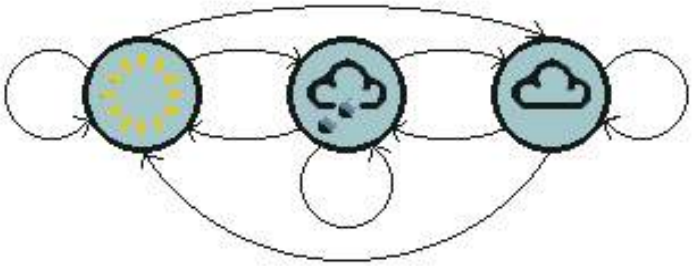

Abbildung 1: Beispiel: Modell des Wetters

zufÄllig. Zum Beispiel folgt Regen mit einer gewissen Wahrscheinlichkeit auf Sonne. Ein solches Modell nennt man ein Markov Modell.

Definition: Ein Markov Modell ist ein System von n Zuständen. Zu diskreten Zeitpunkten t geht das System mit Übergangswahrscheinlichkeiten

$$a_{ij} = P(\omega_j(t+1)|\omega_i(t)) \tag{1}$$

vom Zustand i in den Zustand j über. Diese Übergangswahrscheinlichkeiten kann man in einer Zustandsübergangsmatrix A zusammenfassen.

Für unser Wettermodell gilt beispielsweise:

$$A = \begin{pmatrix} 0,5 & 0,25 & 0,25 \\ 0,375 & 0,125 & 0,375 \\ 0,125 & 0,625 & 0,375 \end{pmatrix}$$

Außerdem gilt die folgende Normalisierungsbedingung:

$$\sum_{j=1}^{N} a_{ij} = 1 \text{ für alle Zustände i} \tag{2}$$

Mit diesem System kann man jetzt die Wahrscheinlichkeit, mit der ein bestimmter Zustand vorliegt, ausrechnen. Allerding muss natürlich noch bekannt sein wie diese Warscheinlichkeiten zum Zeitpunkt der Initialisierung waren. Dies kann man im Vektor π zusammenfassen. Ein Markov Modell wird also durch das Tupel (A,π) vollständig beschrieben.

3 Hidden Markov Models

Oft ist es jedoch gar nicht möglich ein System direkt zu beobachten sondern man kann lediglich sehen, welche Auswirkungen es auf seine Umwelt hat. Beispielsweise könnte man sich vorstellen, dass wir versuchen das Wetter zu beobachten indem wir die Feuchtigkeit eines Holzstücks das im Freien liegt feststellen und versuchen daraus auf das Wetter zurückzuschließen. Wenn wir uns beispielsweise in einem geschlossenen Haus befinden und uns jemand das Holz bringt dann können wir das Wetter selbst nicht mehr beobachten sondern nur noch seine Auswirkungen auf das Holz. Man kann dieses System jetzt als Hidden Markov Model beschreiben.

Definition: Ein Hidden Markov Model besteht aus n Zuständen die jedoch nicht direkt beobachtet werden können. Jeder dieser Zustände $\omega(t)$ emittiert zu jedem Zeitpunkt t ein zufällig ausgewähltes sichtbares Symbol oder auch sichtbarer Zustand genannt. Das gesammte System generiert somit beim durchlaufen von t verborgenen Zuständen die Sequenz $V^T = \{v(1), v(2), ..., v(T)\}$

Für die Übergangswahrscheinlichkeiten gilt:

$$a_{ij} = P(\omega_j(t+1)|\omega_i(t)) \tag{3}$$

Die Wahrscheinlichkeit für die Emission eines bestimmten Symbols v zum Zeitpunkt t wenn sich das System im Zustand ω(t) befindet ist

$$b_j(v(t)) = P(v(t)|\omega_j(t)) \tag{4}$$

Die b_js kann man in der Zustand-Symbolmatrix B zusammenfassen.

Es gibt auch hier einen Vektor π der die initiale Wahrscheinlichkeitsverteilung angibt. Ein Hidden Markov Model ist daher Über (A,B,π) definiert. Außerdem gilt (2) für die a_{ij}s und analog für die b_js (Normalisierungsbedingungen):

$$\sum_k b_j(k) = 1 \text{ für alle sichtbaren Zustände k} \tag{5}$$

Für unser Wetterbeispiel könnte dies dann so aussehen: Wir haben die Zustände Sonne, Wolken und Regen sowie Beobachtungen staubtrocken, trocken, feucht und nass mit den folgenden Wahrscheinlichkeitsmatrizen.

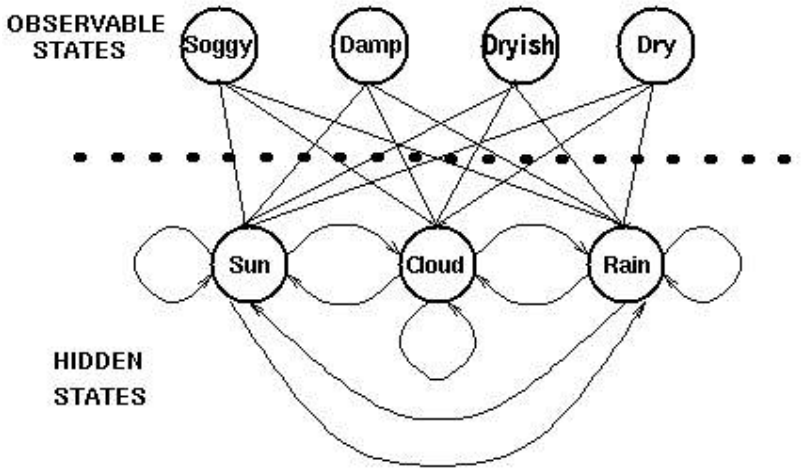

Abbildung 2: Beispiel: Hiddem Markov Modell des Wetters

Zustandsübergangsmatrix

$$A = \begin{pmatrix} 0,5 & 0,25 & 0,25 \\ 0,375 & 0,125 & 0,375 \\ 0,125 & 0,625 & 0,375 \end{pmatrix}$$

Emissionswahrscheinlichkeitsmatrix

$$B = \begin{pmatrix} 0,6 & 0,2 & 0,15 & 0,05 \\ 0,25 & 0,25 & 0,25 & 0,25 \\ 0,05 & 0,1 & 0,35 & 0,5 \end{pmatrix}$$

Initialverteilung

$$\pi = (1, 0, 0)$$

Dies bedeutet dann z.B: im Zustand Sonne ist das Holz mit P= 0,6 trocken. Es gibt auch Modelle bei denen statt Symbolen mit diskreten Wahrscheinlichkeiten kontinuierliche Verteilungen angenommen werden. Darauf soll hier jedoch nicht näher eingegangen werden da dies den Rahmen dieser Arbeit sprengen würde.

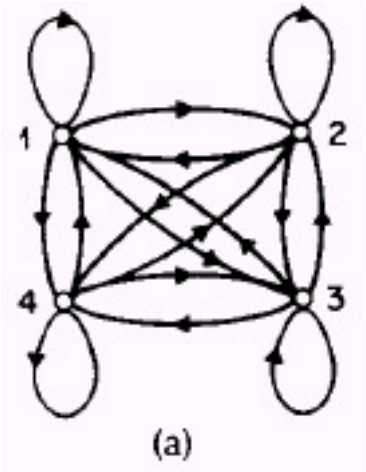

Abbildung 3: Ergodisches Hidden Markov Model

3.1 Verschiedene Typen von HMMs

3.1.1 Ergodische Modelle

Bei einem ergodischen HMM können alle Zustände von jedem Zustand aus innerhalb von einem Schritt mit Wahrscheinlichkeit > 0 erreicht werden. Die Zustandsübergangsmatrix darf daher an keiner Stelle 0 sein und ist von der Form:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

3.1.2 Links-Rechts Modelle

Bei einem Links-Rechts Modell geht das System mit jedem Schritt entweder zum gleichen Zustand oder zu einem bisher noch nicht besuchten über. Daraus folgt :

$$a_{ij} = 0 \text{ für } j < i$$

Zusätzlich kann man noch fordern:

$$a_{ij} = 0 \text{ für } j > i + \Delta$$

Die Übergangsmatrix hat dann Bandenform

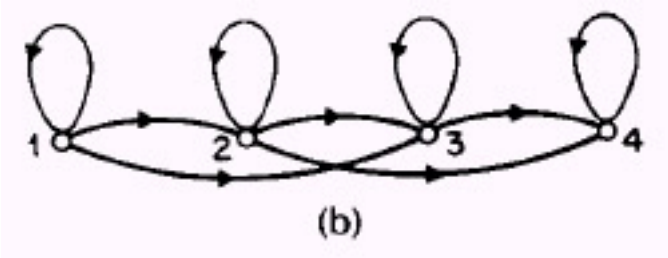

Abbildung 4: Links-Rechts Hidden Markov Model

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{pmatrix}$$

3.2 Zentrale Problemstellungen

Im Zusammenhang mit Hidden Markov Models stellen sich die folgenden 3 Probleme:

Evaluation(Auswertung):

Gegeben ist ein HMM M = (A,B,π)und eine Beobachtungssequenz V^T: Berechne die Wahrscheinlichkeit dass V^T von M erzeugt wurde.

Decoding (Entschlüsseln):

Gegeben ist ein HMM M = (A,B,π)und eine Beobachtungssequenz V^T: Berechne die Folge von verborgenen Zuständen die V^T mit der höchsten Wahrscheinlichkeit erzeugt hat.

Learning (Lernen):

Für ein Hidden Markov Model M ist die Zahl der sichtbaren und unsichtbaren Zustände bekannt, sowie ein oder mehrere Beobachtungssequenzen (Trainigssequenzen):
Berechne die Parameter a_{ij} und b_j(k).

3.3 Lösungen

3.3.1 Evaluation

Um nun herauszufinden mit welcher Wahrscheinlichkeit das System M eine beobachtete Sequenz V^T von sichtbaren Zuständen generiert hat könnte man folgendes Lösungsverfahren anwenden: Man berechnet für alle möglichen Sequenzen die Wahrscheinlichkeit des Durchlaufens vom ersten zum letzten Zustand und gleichzeitiger Emission der sichtbaren Sequenz. Diese Wahrscheinlichkeiten addiert ergeben die Gesamtwahrscheinlichkeit P(V^T).

$$P(V^T) = \sum_{r=1}^{r_{max}} P(V^T|\omega_r^T)P(\omega_r^T) \tag{6}$$

Für alle möglichen Sequenzen $\omega_r^T = \omega(1), \omega(2),..., \omega(\mathrm{T})$

$$P(V^T) = \sum_{r=1}^{N}\prod_{t=1}^{T} P(v(t)|\omega(t))P(\omega(t)|\omega(t-1)) \tag{7}$$

N ist die Anzahl der unsichtbaren Zustände, T ist die Länge der Beobachtungssequenz

Das Problem bei diesem Ansatz ist allerdings, das die Berechnung von der Komplexität $o(N^T T)$ ist und daher in der Praxis nicht verwendet werden kann.

3.3.2 Forward algorithm

Mit diesem Algorithmus kann obiges Problem rekursiv bzw. mit Dynamischer Programmierung gelöst werden. Wir definieren dazu zunächst :

$$\alpha_i(t) = \begin{cases} \pi_i b_i(v(0)) & t = 0 \\ \sum_{j=1}^{N} \alpha_j(t-1)a_{ij}b_j(v(t)) & sonst \end{cases} \tag{8}$$

α ist die Wahrscheinlichkeit, dass sich das Modell gerade im Zustand i befindet, und bereits die ersten t Elemente von V^T produziert hat.

Wir initialisieren α also mit $\pi_i b_i(v(0))$, der Wahrscheinlichkeit für die Emission des ersten Zeichens im jeweiligen Zustand i. Dann berechnen wir für alle Zustände i und für alle Zeiten t die α-Werte. Diese ergeben sich als Summe der Werte für t-1 mal der Übergangswahrscheinlichkeiten mal der Emissionswahrscheinlichkeit. (siehe auch Abb. 5) Die Gesamtwahrscheinlichkeit,dass das System M eine beobachtete Sequenz V^T von sichtbaren Zuständen generiert hat, ergibt sich dann als Summe der α_i(T).

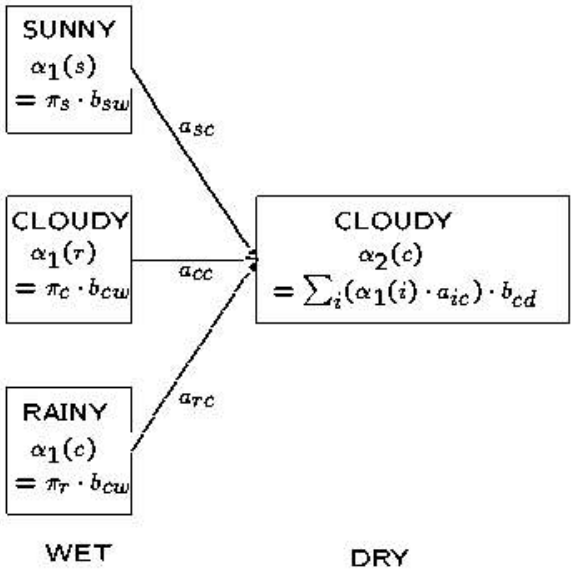

Abbildung 5: Forward Algorithm

Dieser Algorithmus hat die Komplexität o(N^2T) und ist damit deutlich günstiger.

3.3.3 Backward algorithm

Man kann das gleiche Problem auch anders herum lösen und für alle Zustände die Wahrscheinlichkeit berechnen, dass sich das Modell im Zustand i befindet und noch die letzten T-t Elemente der Beobachtungssequenz erzeugen wird:

$$\beta_i(t) = \begin{cases} 1 & \text{für } t = T \\ \sum_{j=1}^{N} \beta_j(t+1) a_{ij} b_j(v(t+1)) & sonst \end{cases} \tag{9}$$

3.3.4 Decoding / Viterbi-algorithmus

Beim Decoding-Problem geht es darum für ein bestimmtes HMM (A,B,π) die Sequenz verborgener Zustände zu finden, die mit der größten Wahrscheinlichkeit die beobachtete Sequenz V^T erzeugt hat. Man könnte dies wieder dadurch lösen, dass man alle möglichen Sequenzen durchgeht und die maximale Wahrscheinlichkeit heraussucht. Dieses Vorgehen hat jedoch

(vgl. Evaluation) exponentielle Komplexität und ist deshalb zu vermeiden. Ein wesentlich schnelleres Verfahren bietet des Viterbialgorithmus. Es handelt sich hierbei um eine Abwandlung des Forward-algorithmus mit zusätzlicher Maximierung der Wahrscheinlichkeit. Wir definieren hierzu zunächst die Funktion $\delta_i(t)$:

$$\delta_i(t) = \max P(\omega(1), ..., \omega(t) = i, v(1), ..., v(t)|M) \text{ über alle Pfade} \quad (10)$$

$\delta_i(t)$ ist also die größte Wahrscheinlichkeit V^T generiert zu haben wenn man sich im Zustand i zur Zeit t befindet. Man kann auch sagen $\delta_i(t)$ ist die Wahrscheinlichkeit über den besten Pfad der Länge t.

Rekursiv kann man dies so schreiben:

$$\delta_i(t) = max_{1 \leq j \leq N}[\delta_j(t-1)a_{ij}]b_j(v(t)) \quad (11)$$

Das heisst, man sucht für jeden Schritt den Partiellen Pfad mit der höchsten Wahrscheinlichkeit heraus, der bis zum Zustand i geht. Da wir aber an der Abfolge der Zustände interessiert sind müssen wir zusätzlich den letzten Zustand in diesem Pfad in $\psi_i(t)$ speichern.(vgl. hierzu Abb.6) Die Berechnung läuft dann in drei Schritten ab:

Initialisierung:

$$\delta_i(0) = \pi(i)b_i(v(0)) \quad (12)$$

Berechnung der Sequenz:

$$\delta_i(t) = max_{1 \leq j \leq N}[\delta_j(t-1)a_{ij}]b_j(v(t)) \quad (13)$$

$$\psi_i(t) = \arg max_{1 \leq j \leq N}[\delta_j(t-1)a_{ij}] \quad (14)$$

Terminierung:

$$\omega(T) = \arg max_{1 \leq j \leq N}[\delta_j(t-1)] \quad (15)$$

Sequenzrückverfolgung:

$$\omega(t) = \psi_{\omega(t+1)}(t+1) \text{ für } 1 \leq t \leq T-1 \quad (16)$$

Dieser Algorithmus bezieht den gesamten Kontext mit in die Berechnung ein und erzeugt eine global optimierte Lösung. Dies wirkt sich in der Praxis positiv aus wenn man z.B. stark rauschende Signalquellen hat.

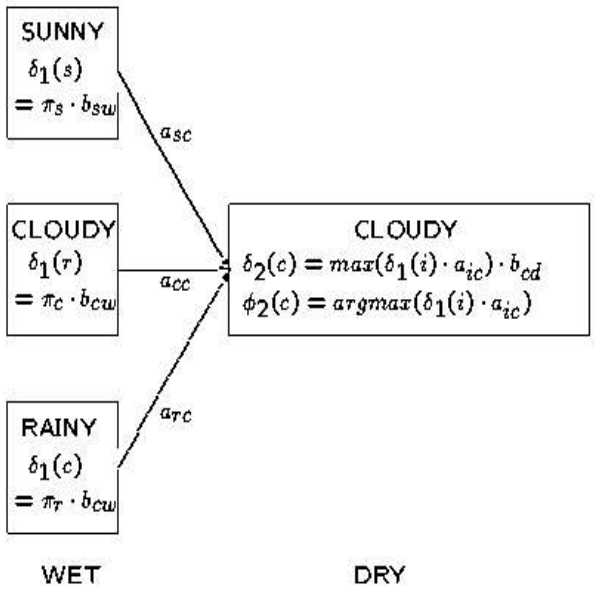

Abbildung 6: Viterbi Algorithm

3.3.5 Learning

Beim Learning oder Trainingsproblem versucht man für ein HMM die Parameter a_{ij} und b_j aus einer gegebenen Trainingssequenz V^T zu ermitteln. Die Anzahl der verborgenen und sichtbaren Zustände ist dabei bekannt. Gesucht werden nur die Übergangswahrscheinlichkeiten. Da es keine Möglichkeit gibt diese Parameter analytisch zu berechnen bedient man sich hierbei einer Abwandlung des EM-Algorithmus (Erwartungs-Modifikation) um diese Werte iterativ abzuschätzen.

$$\xi_{ij}(t) = P(\omega_i(t), \omega_j(t+1)|V^T, M) \tag{17}$$

$$\xi_{ij}(t) = \frac{\alpha_i(t) a_{ij} b_j(v(t+1)) \beta_j(t+1)}{P(V^T|M)} \tag{18}$$

$$\xi_{ij}(t) = \frac{\alpha_i(t) a_{ij} b_j(v(t+1)) \beta_j(t+1)}{\sum_k = 1^N \sum_l = 1^N \alpha_k(t) a_{kl} b_l(v(t+1)) \beta_l(t+1)} \tag{19}$$

ist die Wahrscheinlichkeit eines Übergangs von $\omega(t-1) = i$ nach $\omega(t) = j$ wenn das Modell M die Folge V^T über einen ansonsten beliebigen Pfad generiert hat. Wenn man nun über die Wahrscheinlichkeiten ξ_{ij} für alle Nach-

folgezustände $\omega(t+1) = j$ aufsummiert ergibt sich die Wahrscheinlichkeit γ_i sich zum Zeitpunkt t im Zustand $\omega(t) = i$ zu befinden.

$$\gamma_i(t) = \sum_{j=1}^{N} \xi_{ij}(t) \tag{20}$$

Es lassen sich nun folgende Erwartungswerte ausrechnen:
Die zu erwartende Anzahl von Transitionen von ω_i zu ω_j:

$$\sum_{t=1}^{T-1} \xi_{ij}(t) \tag{21}$$

Der Erwartungswert der Übergänge von ω_i zu jeden beliebigen anderen Zustand:

$$\sum_{t=1}^{T-1} \gamma_i(t) \tag{22}$$

Und die zu erwartende Aufenthaltanzahl in ω_i:

$$\sum_{t=1}^{T} \gamma_i(t) \tag{23}$$

Hierdurch lassen sich dann verbesserte Werte ausrechnen.

$$\overline{\pi}_i = \text{Aufenthaltswahrscheinlichkeit in } \omega_i \text{ zur Zeit } 0 \ = \gamma_i(0) \tag{24}$$

$$\overline{a}_{ij} = \frac{\text{Erwartungswert Übergänge von } \omega_i \text{ zu } \omega_j}{\text{Erwartungswert der Übergänge von } \omega_i} = \frac{\sum_{t=1}^{T-1} \xi_{ij}(t)}{\sum_{t=1}^{T-1} \gamma_i(t)} \tag{25}$$

$$\begin{aligned} \overline{b}_j(k) &= \frac{\text{Erwartungswert der Aufenthalte in } \omega_i \text{ mit Emmission des Symbols k}}{\text{Erwartungswert der Aufenthalte in } \omega_i} \\ &= \frac{\sum_{t=1}^{T} \gamma_i(t) b_i(k)}{\sum_{t=1}^{T} \gamma_i(t)} \end{aligned}$$

Durch wiederholte Durchführung der obigen Berechnung kann man somit eine immer bessere Abschätzung für das HMM geben. Dieser Algorithmus findet jedoch nur eine lokal optimale Lösung. Außerdem stellen sich folgende Probleme:

Wie wählt man die Startwerte?

Für die π und a_{ij} kann man entweder eine gleichförmige oder zufällige Verteilung wählen. Bei den b_j ist es zweckmäßig bereits gute Abschätzungen vorliegen zu haben um eine schnelle Konvergenz des Algorithmus zu gewährleisten. Dies Vorabschätzungen könnten beispielsweise durch manuelle oder Maximum Likelihood-Abschätzung gewonnen werden.

Welches Model verwendet man?

Die Frage nach der Art des Modells kann man nicht generell beantworten sondern hängt davon ab welche Art von Signal man mit diesem Modell modellieren will. Z.B. für Spracherkennung bietet sich ein Links-Rechts Modell an ,da hier der fortlaufende Charakter des Sprachsignals am besten mit einbezogen werden kann.

Scaling

Die bei der Berechnung auftretenden Werte sind meist wesentlich kleiner als eins und die Ergebnisse daher gegen null konvergieren. Da wir aber nur begrenzte Maschinengenauigkeit verwenden können führt dies zwangsläufig zu Rundungsfehlern. Als Lösung bietet sich an die α_i und β_i mit einem Skalierungsfaktor zu multiplizieren, der unabhänging von i und abhänging von t ist:

$$c_t = \frac{1}{\sum_{i=1}^{N} \alpha_i(t)} \tag{26}$$

Dieser Faktor kürzt sich anschließend automatisch wieder heraus.

4 Spracherkennung als Anwendungsbeispiel

In diesem Kapitel soll kurz auf Spracherkennung als mögliche Anwendung dieser mathematischen Modellierung eingegangen werden. Sie bietet sich insbesondere an für sogenannte *Isolated word recognizer*, Spracherkennungssysteme, die nur geringen Wortschatz erkennen sollen, aber dafür von vielen, auch unbekannten Sprechern, verwendet werden können. Beim Isolated word recognizer werden für jedes Wort das später erkannt werden soll ein HMM angelegt. Um dieses nun trainieren zu können, d.h. die Parameter abschätzen zu können, muss man das aufgenommene Sprachsignal zunächst analysieren. Durch Quantisierung wird das Signal in Vektoren zerlegt, die dann die Beobachtungssequenz bilden, mit der das Modell nun trainiert wird.

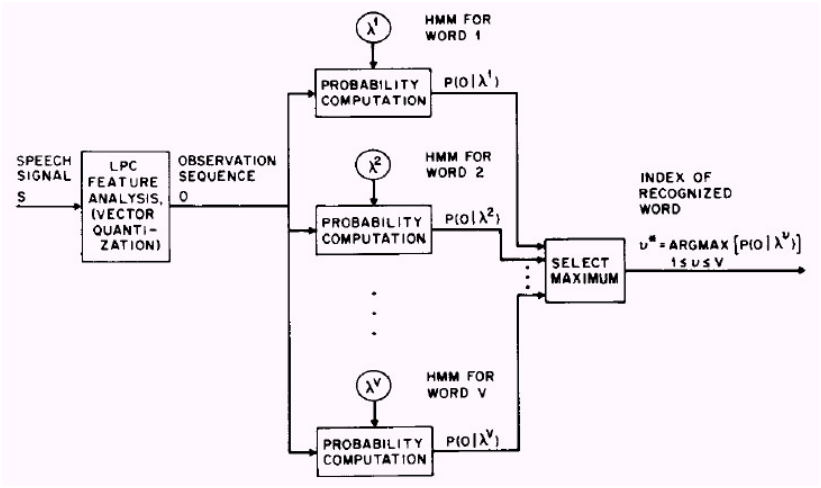

Abbildung 7: Isolated word recognizer

Wenn nun ein Wort wiedererkannt werden soll führt man zunächst wieder eine Quantisierung des Signals durch und berechnet dann für alle Modelle mit dem Forward-algorithmus die Wahrscheinlichkeit das diese Sequenz von diesem Modell stammt. Das Wort mit der maximalen Wahrscheinlichkeit wird dann ausgewählt.

5 Zusammenfassung

Hidden Markov Models können immer dann ein guter Ansatz zur Modellierung eines Systems sein, wenn man das zu Grunde liegende System nicht direkt beobachten kann. Ihre Vorteile sind insbesondere relativ einfach zu handhabende Mathematische Grundlagen und geringe Fehlerraten. Aber Hidden Markov Models haben auch viele Einschränkungen. Die Wahl des richtigen Modells ist oft schwierig, man braucht ausreichend Trainingsdaten für gute Parameterabschätzungen und auch die Markovannahme, dass jeder Zustand nur von seinem Vorgängerzustand abhängt ist nicht für alle Probleme gegeben. Insbesondere bei Spracherkennung sind sie aber ein möglicher Ansatz und führen dort auch zu einer ausreichenden Erkennungsleistung.

6 Verwendete Literatur

Bücher

Rabiner, L.R. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition; Proceedings of the IEEE, Vol.77, Iss.2, Feb 1989; Pages:257-286

Richard O. Duda, Peter E. Hart, David G. Stork Pattern Classification chapter 3.10 Eric Keller (Editor) Fundamentals of Speech synthesis and

Speech recognition Chapter 8 by Kari Torkkola

Internet

http://www.comp.leeds.ac.uk/roger/HiddenMarkovModels/html_dev/main.html Introduction to Hidden Markov Models, University of Leeds

http://www.cnel.ufl.edu/ỹadu/report1.html Speech recognition using Hidden Markov Models, Yadunandana Nagaraja Rao , University of Florida