Handout

Hidden Markow Modelle

Statistische Modellierung latenter Strukturen

Winter 13/14

Eingereicht von David Bauder
betreut von Georg Schollmeyer
München, 17.01.2014

Inhaltsverzeichnis

1 Einleitung

Eines der vielleicht meiststrapazierten pädagogischen Beispiele für Markow-Ketten ist wahrscheinlich das Beispiel eines Übergang vom einen Wetterzustand in einen anderen. Die Wahrscheinlichkeit eines bestimmten Wetterzustands hängt in diesem Beispiel nur vom Wetter des vorhergehenden Tages ab (allgemein ist eine Markow-Kette ein stochastischer Prozess, deren Wert im jetzigen Zeitpunkt nur von ihrem vorhergehenden Wert abhängt). Bei HMMs ist aber das Wetter gar nicht beobachtbar, aber trotzdem die interessierende Größe. Bestimmte Auswirkungen des Wetters sind aber natürlich beobachtbar, mit denen man eventuell auf das Wetter schließen könnte. Beispielsweise lassen nasse, schlammige Schuhe auf schlechtes Wetter schließen oder trockene Schuhe auf gutes. Kennt man die Wahrscheinlichkeiten des Wetterübergangs und die Wahrscheinlichkeit des Schuhaussehens in einem Wetterzustand, könnte man also tatsächlich mit Hilfe eines HMMs auf das Wetter schließen. Oder aber man schätzt diese Übergänge. Beide Probleme sind Gegenstand der folgenden Seiten.

Ein HMM ist eine doppelt-stochastische Markow-Kette, bei der eine (unbeobachtbare) Zustandssequenz entsprechend einer Markow-Kette auf einem diskreten Zustandsraum gezogen wird. Die Beobachtungssequenz wird bedingt auf die Zustandssequenz gezogen. Die unbeobachtbare, latente Sequenz wird als Markow-Kette modelliert, hauptsächlich um vorteilhafte Berechnungseigenschaften zu erhalten, während keine weiteren Annahmen an die Beobachtungssequenz gestellt werden (außer die Bedingung auf den unbeobachtbaren Zustand).

Diese Modellklasse ist auf der einen Seite einfach genug, um sie mit gerechtfertigtem Aufwand anzuwenden und auf der anderen Seite auch flexibel genug, um auch komplexere Fragestellungen angehen zu können. Populär wurde die Methode seit den Arbeiten um Baum et al. (1970) insbesondere in der Spracherkennung von Jelinek (1969) und weiteren Kollegen bei IBM. Eine der ersten und scheinbar jetzt auch am bekanntesten Zusammenfassungen zur HMM-Theorie von Rabiner (1989) mit einem Schwerpunkt auf Anwendungen in der Spracherkennung. Modelle zur Wortsegmentierung wie in Goldwater et al. (2006, TehJor) oder das Zuordnen von Sprachsequenzen zu bestimmten Personen wie in Huang et al. (2001) oder in verschiedenen Open-Source Paketen wie Audioseg demonstrieren die Anwendbarkeit von HMMs in diesen Bereichen. Diese Anwendungen von HMMs waren kürzlich sicher unter vielen Weihnachtsbäumen zu finden: Etwa wenn man Speech-to-Text Transcription in Smartphones gerne nutzt oder auf der Autobahn lieber das Reiseziel per Spracherkennung in das Navigationssystem eingibt als mehr oder weniger zielsicher das Ziel einzutippen.

HMMs sind auch sehr populär in genetischen Anwendungen, beispielsweise zum Auffinden von Cytosin-phosphatidyl-Guanin-Inseln (oder einfach CpG-Inseln) im Ge-

nom. CpG-Inseln sind bestimmte Abschnitte im Genom, die einen erhöhten CG-Gehalt aufweisen und für die Genregulation bzw. Genexpression eine wichtige Rolle spielen. Hier haben sich HMMs und verschiedene Erweiterungen bis hin zu nichtparametrischen bayesianischen Methoden als sehr hilfreich erwiesen (s. Durbin et al., 1998 und für Erweiterungen bspw. Teh et al., 2006 oder Yau et al., 2011).

In diesem Text liegt der Fokus aber eher auf eine Vorbereitung auf den Vortrag. Ausgeklammert werden in diesem Text vor allem die etwas komplizierteren Anwendungen in der Spracherkennung und die Anwendungen in der Genetik. Trotzdem soll ein wenig der Mund wässerig für mögliche Erweiterungen gemacht werden und ein paar Grundlagen diskutiert werden, um etwas in Richtung bayesianischer Erweiterungen stoßen zu können. Zunächst geht es aber erstmal um das Grundmodell und Methoden zur Schätzung verschiedener Fragestellungen.

2 Grundmodell

2.1 Markow-Ketten

Da Markow-Ketten zentral bei HMMs sind, sollten zumindest kurz die beiden zentralen Eigenschaften einer Markow-Kette beschrieben werden. In der Einleitung wurde kurz angesprochen, dass die Wahrscheinlichkeit, einen bestimmten Zustand zu erreichen, nicht davon abhängt, wie der ganze vorhergehende Pfad realisiert wurde, sondern nur vom letzten Zustand. Formal also

$$P(X_t = x_i | X_{t-1} = x_i, ..., X_1 = x_i) \quad = \quad P(X_t = x_i | X_{t-1} = x_i).$$

Zusätzlich sollen auch noch die Wahrscheinlichkeiten, dass ein bestimmter Zustand erreicht wird, immer gleich sein. Diese Annahmen wirken natürlich sehr restriktiv, in kaum einer Anwendung könnte man so eine Annahme inhaltlich vertreten. Denkt man beispielsweise an Spracherkennung, würde die Modellierung über eine Markow-Kette bedeuten, dass das jetzt gesprochene Wort nur vom Wort davor abhängen würde. Trotzdem kann ein Totschlagargument gelten: Es funktioniert im HMM-Kontext sehr gut und erleichtert den rechnerischen Aufwand enorm.

2.2 Beispiel: Das unehrliche Kasino

Eines der einfachsten Illustrationen für ein HMM ist ein Würfelspiel, bei dem die meiste Zeit mit einem fairen Würfel gewürfelt wird, aber hin und wieder der faire Würfel durch einen unfairen Würfel ersetzt wird. Dieser unfaire Würfel fällt mit Wahrscheinlichkeit 1/2 auf eine 6 und mit jeweils der Wahrscheinlichkeit von 1/10 auf die Zahlen 1 bis 5. Das Kasino wechselt vor jedem Wurf von einem fairen zu einem unfairen Würfel mit der Wahrscheinlichkeit 1/20 und von einem unfairen zu einem fairen Würfel mit Wahrscheinlichkeit 1/10.

Welcher Würfel aber nun verwendet wird, kann nicht beobachtet werden. Die Würfel sind hier also die beiden unbeobachteten Zustände des Modells. Die maximale Anzahl an Zuständen bezeichnen wir hier mit K, also ist in diesem Beispiel $K = 2$. Die Übergangsmatrix der unbeobachteten Zustände, bezeichnet mit Q, wäre hier

$$Q \quad = \quad \begin{pmatrix} 0.95 & 0.05 \\ 0.1 & 0.9 \end{pmatrix}$$

Die Wahrscheinlichkeit, der Emission "im ersten Wurf eine 6" ist bei einem fairen

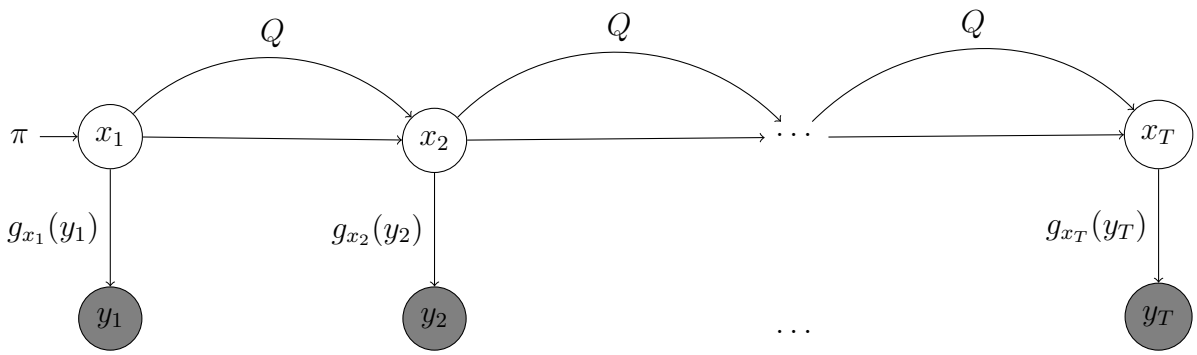

Abbildung 1: Struktur eines hidden Markov Modells. Beobachtbare Komponenten sind grau schattiert.

Würfel gegeben als $g_{\text{fair}}(6) = P(y_1 = 6|x_1 = \text{fair}) = 1/6$, während klarerweise bei einem unfairen Würfel dann $g_{\text{unfair}}(6) = P(y = 6|x = \text{unfair}) = 1/2$ gilt. Die Struktur eines solchen Modells wird in Abbildung 1 gezeigt.

Will man nun eine Beobachtungssequenz simulieren, wählt man zunächst den ersten (unbeobachteten) Zustand x_1, also den Würfel, entsprechend der Ausgangswahrscheinlichkeiten π. Hier spielen wir einfach mit einer Ausgangswahrscheinlichkeit von 1/2 für jeden Würfel. Anschließend wird y_t entsprechend $g_{x_1}()$ gezogen, also mit dem Würfel gewürfelt. Mit den Wahrscheinlichkeiten aus Q wird dann der nächste Zustand bestimmt und entsprechend diesem Zustand wieder eine Beobachtung gezogen. Diesen Algorithmus setzt man nun bis zur gewünschten Sequenzlänge fort. Setzen wir uns beispielsweise zehn Würfelwürfen aus, könnte die Sequenz so aussehen:

Zustand	F	F	F	F	F	F	F	F	U	U
Beobachtung	1	6	5	2	6	6	3	1	6	3

Da die Übergangswahrscheinlichkeiten in Q so gewählt sind, dass selten gewechselt wird, ist die Anzahl der Würfe mit einem fairen Würfel nicht überraschend. Die Frage, wann und ob gewechselt wurde, ist natürlich besonders interessant, selbstverständlich aber auch die Frage nach der unbeobachteten Zustandssequenz.

Verbringt man nun einige Zeit im Kasino und abstrahiert das Würfelspiel in eine HM-Struktur, kommen drei Fragen auf (Rabiner, 1989):

1. Wie berechnet man die Wahrscheinlichkeit einer Beobachtungssequenz bei gegebenem HMM?

2. Wie bestimmt man die wahrscheinlichste Sequenz der verborgenen Zustände X^* bei gegebenem HMM?

3. Wie schätzt man die Parameter eines HMMs bei gegebener Beobachtungssequenz?

Wo es nötig wird gehen wir diese drei Probleme am Beispiel des unfairen Kasinos durch. Das zweite Problem ist durch einen Viterbi-Algorithmus lösbar, der vor allem im Vortrag etwas genauer diskutiert werden soll, auch hinsichtlich seiner statistischen Eigenschaften (bspw. Konsistenz). Die Lösung zum dritten Problem ist deutlich aufwändiger und wird in klassischen Fällen durch den Baum-Welch Algorithmus erreicht, ein spezieller EM-Algorithmus. Um Probleme zwei und drei angehen zu können, müssen wir allerdings zuerst an die Wahrscheinlichkeit einer Beobachtungssequenz bei gegebenem Modell kommen, also das erste Problem sinnvoll ausdrücken.

2.3 Elemente und Notation eines HMM

Da die Notation bei HMMs eine recht große Hürde darstellen kann, werden wir hier zunächst die "Zutaten" eines HMMs darstellen. Spätestens nach der dritten unterschiedlichen Darstellung von HMMs in der Literatur merkt man, dass die Notation zu jede Menge Verwirrung führen kann, gerade wenn man versucht, die Algorithmen in den nächsten Abschnitten nachzuvollziehen. Sollte irgendwo in diesem Handout irgendwas verwechselt werden, würde ein Hin

- K: (Diskrete) Anzahl an unbeobachteten Zuständen des Modells.
- M: Anzahl an möglichen Beobachtungen.
- T: Anzahl an Beobachtungen einer Sequenz bzw. Länge einer Sequenz.
- $X = (x_1, x_2, ..., x_T)$: Sequenz der unbeobachteten Zustände.
- $Y = (y_1, y_2, ..., y_T)$: Sequenz der beobachteten Realisationen ("Emissionen").
- $\pi = (\pi_1 = P(x_1 = i), ..., \pi_K)$: Ausgangswahrscheinlichkeit für unbeobachteten Zustand, $\sum_{i=1}^{K} \pi_i = 1$.
- $Q = \{q_{ij} : i = 1, ..., K; j = 1, ..., K\}$ Übergangsverteilung der unbeobachteten Zustände, hier meist eine stochastische Matrix mit $q_{ij} = P(x_{t+1} = j | x_t = i)$, also gilt auch $\sum_{j=1}^{K} q_{ij} = 1$.
- $G = \{g_k(y_t) : k = 1, ..., K; t = 1, ..., T\}$: Bedingte Likelihood der Beobachtungen mit $g_k(y_t) = P(y_t | x_t = k)$.
- $\theta = \{\pi, Q, G\}$: Parametervektor eines vollständig spezifizierten HMMs.

Die Anzahl K der unbeobachtbaren Zustände kann in vielen praktischen Anwendungen aus dem Kontext abgeleitet werden. Die Zustände sind in einer Weise so verbunden, dass ein Zustand vom anderen erreicht werden kann. Die Anzahl an möglichen unterschiedlichen Beobachtungen M kann beispielsweise die Anzahl an Buchstaben eines Alphabets oder die Anzahl der Seiten eines Würfels sein. Natürlich sind Q und G besonders interessant. Die Matrix $Q := \{q_{ij}; i = 1, ..., K; j = 1, ..., K\}$ ist befüllt mit den Einträgen der Wahrscheinlichkeit eines Übergangs von Zustand i in Zustand j, also $q_{ij} = P(x_{t+1} = j|x_t = i)$, natürlich mit der Bedingung, dass $\sum_{j=1}^{K} q_{ij} = 1$. Die Wahrscheinlichkeit einer Beobachtung ist dagegen gegeben als $G = \{g_k(y_t); k = 1, ..., K; t = 1, ..., T\}$ und gilt, analog zu Q, als "Emissionsmatrix". Hier ist $g_k(y_t) = P(y_t|x_t = k)$. Gemeint ist also die Wahrscheinlichkeit, y zum Zeitpunkt t zu ziehen, wenn man sich im Zeitpunkt t in Zustand $x_t = k$ befindet.

3 Problem 1: Wahrscheinlichkeit einer Beobachtungssequenz

Um die Wahrscheinlichkeit der Beobachtungssequenz $Y = y_1, ..., y_T$ bei gegebenem Modell zu bestimmen, also $P(Y|\theta)$, ist der natürlichste Ansatz wahrscheinlich das Aufsummieren jeder möglichen Zustandssequenz $X = x_1, ..., x_T$ der gleichen Länge T. Diese Möglichkeit soll nun zuerst vorgestellt werden. Für eine gegebene Zustandssequenz X wäre die Wahrscheinlichkeit für eine Beobachtungssequenz Y, bei unterstellter Unabhängigkeit der Beobachtungen voneinander, einfach

$$P(Y|X, \theta) = \prod_{t=1}^{T} P(y_t|x_t, \theta) = \prod_{t=1}^{T} g_{x_t}(y_t).$$

Die Wahrscheinlichkeit, die Zustandssequenz Y bei gegebenem Modell θ ergibt sich logischerweise aus dem Produkt der Übergangswahrscheinlichkeiten und der Ausgangswahrscheinlichkeit π, also als

$$P(X|\theta) = \pi \prod_{t=2}^{T} P(x_t|x_{t-1}).$$

Die Wahrscheinlichkeit, dass X und Y gemeinsam bzw. "gleichzeitig" auftreten, also die gemeinsame Wahrscheinlichkeit von X und Y, kann unter diesen Voraussetzungen also einfach als das Produkt der beiden eben erwähnten Wahrscheinlichkeiten geschrieben

werden, also als

$$P(Y, X|\theta) \quad = \quad P(Y|X, \theta)P(X|\theta).$$

Die Wahrscheinlihckeit einer Beobachtungssequenz bei gegebenem Modell ergibt sich entsprechend unseres Ansatzes dann daraus, $P(Y, X|\theta)$ also über alle möglichen Zustandssequenzen zu summieren, also als

$$P(Y|\theta) \quad = \quad \sum_X P(Y|X, \theta)P(X|\theta) = \sum_X \pi \prod_{t=2}^{T} P(x_t|x_{t-1}) \prod_{t=1}^{T} g_{x_t}(y_t). \tag{1}$$

Mit K^T möglichen Zustandssequenzen wäre es natürlich keine gute Idee, Gleichung (1) direkt zu verwenden (bei einer Sequenz von zehn Würfelürfen wären es immerhin 1024 mögliche Zustandssequenzen).

Eine Lösung mit geringerem Komplexitätsgrad bietet der Forward/Backward-Algorithmus. Teilt man die Beobachtungs- und Zustandssequenzen in einen Teil bis zum Zeitpunkt t und einen Teil von $t + 1$ und T auf, ergibt sich die gemeinsame Wahrscheinlichkeit für die Beobachtungssequenz bis zum Zeitpunkt t (Forward-Ansatz) abhängig vom Zustand k als

$$\alpha_t(k) \quad = \quad P(y_1, ..., y_t|x_t = k|\theta) = \sum_{j=1}^{K} \alpha_{t-1}(j) a_{jk} g_k(y_t), \tag{2}$$

mit $\alpha_1(i) = \pi_i g_k(y_t)$. Die Wahrscheinlichkeit der verbleibenden Beobachtungen bei $x_t = k$ ergibt sich dann (Backward-Teil) als

$$\beta_t(k) \quad = \quad P(y_{t+1}, ..., y_T|x_t = k|\theta) \tag{3}$$

und $\beta_T(i) = 1$. Da die bedingte Wahrscheinlichkeit von $y_{t+1}, ..., y_T$ bei gegebenem $x_t = k$ unabhängig ist von den vorangegangenen Werten $y_1, ..., y_t$, gilt auch, dass

$$P(Y, x_t = k|\theta) \quad = \quad \alpha_t(k) \cdot \beta_t(k),$$

aber insbesondere auch

$$P(Y|\theta) \quad = \quad \sum_{k=1}^{K} P(Y, x_t = k|\theta) = \sum_{k=1}^{K} \alpha_t(k) \cdot \beta_t(k) = \sum_{k=1}^{K} \alpha_T(k),$$

was schließlich auch die Lösung zu Problem 1 darstellt. Während die direkte Lösung noch K^T Schritte zur Berechnung benötigen würde, benötigt dieser Algorithmus nur $K^2 \cdot T$ Schritte. Natürlich muss man nicht den Forward- und den Backwardansatz durchführen, es reicht der eine oder der andere Ansatz.

Um das Ganze jetzt aber in eine etwas "handlichere" Form zu bekommen, geben die folgenden Schritte eine kompakte Zusammenfassung des Forward-Algorithmus:

Forward:

1. Initialisierung: $\alpha_1(i) = \pi_i g_i(y_1)$, $1 \leq i \leq K$
2. Induktion: $\alpha_{t+1}(j) = \left(\sum_{i=1}^{K} \alpha_t(i) a_{ij}\right) g_j(y_{t+1})$, $1 \leq t \leq T-1$, $1 \leq j \leq K$.
3. Terminierung: $P(Y|\theta) = \sum_{i=1}^{K} \alpha_T(i)$

Oder bereits

Der Backward-Algorithmus wird hauptsächlich in in der Lösung zu Problem 3 verwendet und wird eignetlich nicht zur Lösung von Problem 1 gebraucht. Der Vollständigkeit halber hier aber auch die andere Richtung.

Backward:

1. Initialisierung: $\beta_T(i) = 1$, $1 \leq i \leq K$
2. Induktion: $\beta_t(i) = \sum_{j=1}^{K} a_{ij} g_j(y_{t+1}) \beta_{t+1}(j)$, $t = T-1, T-2, ..., 1$, $1 \leq i \leq K$.

Beide Algorithmen findet man als R-Code auch im Anhang und ist auch für den Algorithmus zur Lösung des dritten Problems als Input notwendig.

4 Problem 2: Bestimmen der optimalen Zustände

Es gibt mehrere Wege, Problem 2 zu lösen. Insbesondere aber ist die Lösung des zweiten Problems davon abhängig, was man unter einer optimalen Sequenz versteht. Das intuitiv vielleicht eingängigste Qualitätskriterium wäre die erwartete Anzahl korrekter individueller Zustände. Der gängige Algorithmus zur Lösung des zweiten Problems ist der Viterbi-Algorithmus.

Für diesen Ansatz beginnt man mit einer Variable für die Wahrscheinlichkeit, in Zustand x_i zum Zeitpunkt t zu sein, bei gebener Beobachtungssequenz Y und Modell θ, formal

$$\gamma_t(i) \quad = \quad P(x_t = k|Y, \theta),$$

bzw. ausgedrückt in Forward- und Backward-Variablen

$$P(x_t = k|Y,\theta) \quad = \quad \frac{\alpha_t(i)\beta_t(i)}{P(Y|\theta)} = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^{K}\alpha_t(i)\beta_t(i)}.$$

Die Normalisierung ist notwendig, sodass $\gamma_t(i)$ tatsächlich ein Wahrscheinlichkeitsmaß ist. Mit $\gamma_t(i)$ kann nun der jeweils wahrscheinlichste Zustand $x_t = \arg\max_{1\leq\leq K}(\gamma_t(i))$, $1 \leq t \leq T$. Das wäre zumindest eine Idee. Allerdings kann die so geschätzte Zustandssequenz letztendlich unzulässig sein, beispielsweise wenn die Übergangsmatrix der Zustände Q Nulleinträge aufweist. Eine Lösung für dieses Problem kann einfach ein anderes Optimalitätskriterium sein. Anstatt nur das wahrscheinlichste x_t zu bestimmen, kann man beispielsweise die erwartete Anzahl an korrekten Zustandspaaren (x_t, x_{t+1}) (oder mehr) optimieren. Wenn man diese Idee konsequenz anwendet, wird man versuchen, die beste Zustands*sequenz* finden wollen, also einfach $P(X|Y,\theta)$ bzw. $P(X,Y|\theta)$ zu maximieren. Eine Möglichkeit, diesen Ansatzin der Praxis anzuwenden, ist der Viterbi-Algorithmus.

Anfangs sei

$$\delta_t(i) \quad = \quad \max_{x_1,\ldots,x_{t-1}} P(x_1,\ldots,x_t = i; y_1,\ldots,y_t|\theta)$$

die größte Wahrscheinlichkeit Wahrscheinlichkeit einer Zustandssequenz bis t, die in Zustand x_i stoppt. Das kann über die Rekursion

$$\delta_t(i) \quad = \quad \max_{1\leq j\leq K}(\delta_{t-1}(j)a_{ij})g_i(y_t) \tag{4}$$

auch berechnet werden. Um nun (4) umzusetzen, führen wir zunächst einen Array $\psi_t(j)$ ein, der die jeweils $\delta_t(j)$ maximierenden Argumente enthält. Dann führen wir die folgenden vier Schritte aus:

1. Initialisierung, für alle i:

$$\begin{aligned}\delta_1(i) &= \pi_i g_i(y_1)\\ \psi_1(i) &= 0\end{aligned}$$

2. Rekursion, für alle j:

$$\begin{aligned}\delta_t(j) &= \max_i(\delta_{t-1}(j)a_{ij})g_i(y_t),\ 2 \leq t \leq T \\ \psi_t(j) &= \arg\max_i(\delta_{t-1}(j)a_{ij})g_i(y_t),\ 2 \leq t \leq T\end{aligned}$$

3. Terminierung:

$$x_T^* = \arg\max_i \delta_T(i)$$

4. Sequenz:

$$x_t^* = \psi_{t+1}(x_{t+1}^*),\ t = T-1, T-2, ..., 1$$

Hier verwendet man in der ersten Gleichung des Rekursionsschrittes im Viterbi-Algorithmus über die Maximierung der vorhergehenden Zustände, anstatt wie im Induktionsschritt der Forward-Prozedur die Summe der $\alpha_t(i)a_{ij}$.

5 Fortsetzung: Das unehrliche Kasino

Um den Formelpaketen in den letzten Absätzen jetzt aber etwas mehr Leben einzuhauchen, wollen wir das Schätzen der Zustandssequenzen hier etwas detaillierter am Beispiel der (un)fairen Würfel durchgehen. Wenn wir mit einem Würfel spielen, kennt man sowohl die Elemente des Vektors "mögliche Ausprägungen" als auch die Länge dieses Vektors, also einen Vektor mit Einträgen von 1 bis 6 und einer Länge von 6. Zusätzlich wissen wir auch, dass jedes Ereignis gleichwahrscheinlich ist, wenn ein fairer Würfel verwendet wird. Bei einem unfairen Würfel gehen wir davon aus, dass die Ereignisse 1 bis 5 gleichwahrscheinlich sind, das Ereignis 6 aber genauso wahrscheinlich ist wie die Zahlen 1 bis 5. Wir kennen also die Emissionsmatrix. Entsprechend Abschnitt 2.2 wurde eine Sequenz von Ereignissen beobachtet, deren unbeobachtete Zustände nun gefunden werden sollen. Die folgende Funktion führt einen Viterbi-Algorithmus für das Würfelproblem aus:

```
viterbimatrix <- function(sequence, transitionmatrix, emissionmatrix){
  sequence <- sequence
  numstates <- dim(transitionmatrix)[1]
  v <- matrix(NA, nrow=length(sequence), ncol=dim(transitionmatrix)[1])
  v[1,] <- 0
  v[1,1] <- 1
```

```
7
8   for(i in 2:length(sequence)){
9     for(j in 1:numstates){
10      statejprobobservationi <- emissionmatrix[j, sequence[i]]
11      v[i,j] <- statejprobobservationi * max(v[(i-1),]*transitionmatrix[,j])
12    }
13  }
14  return(v)
15 }
```

Übergeben werden muss also eine Beobachtungssequenz "sequence", die Übergangsmatrix der Zustände und die Emissionsmatrix. Der eigentliche Algorithmus beginnt in Zeile 8. Zeile 10 gibt die Wahrscheinlichkeit eines Würfelwurfereignisses, gegeben dass wir für Wurf i in Zustand j sind. Diese Wahrscheinlichkeit kann direkt aus der Emissionsmatrix abgelesen werden, entspricht also dem gesuchten $g_i(y_t)$ aus dem letzten Abschnitt.

Zeile 11 ist der Hauptteil des Rekursionsschrittes: Das erste Argument der Maximumsfunktion entspricht hier $\delta_{t-1}(j)$, das zweite Argument wiederum Element a_{ij} der Übergangsmatrix. Beide Argumente werden an jedem Beobachtungszeitpunkt zunächst für alle Zustände berechnet und schließlich für jeden Beobachtungspunkt (innere und äußere Schleife). Ein Ergebnis des Algorithmus für eine Sequenz von 100 Würfelwürfen ist hier dargestellt:

	Position	Zustand
geschätzte Zustände	1 – 36	fair
	37 – 51	unfair
	52 – 53	unfair
	54 – 54	fair
	55 – 100	unfair
wahre Zustände	1 – 8	fair
	9 – 10	unfair
	11 – 11	fair
	12 – 19	unfair
	20 – 24	fair
	25 – 47	unfair
	48 – 100	fair

Da die wahre Sequenz kurze Phasen enthält und die Übergangsmatrix Q zwar richtig im Viterbi-Algorithmus spezifiziert wurde, diese Matrix aber kurze Phasen eher nicht unterstützt, schafft es der Algorithmus nicht, diese kurzen Phasen zu beschreiben (unter anderem auch wegen der rekursiven Natur des Algorithmus). Allerdings ist die Sequenz

mit 100 Beobachtungen zum einen nicht besonders lang, insofern könnte man sich fragen, ob der Viterbi-Algorithmus nicht mit zunehmender Beobachtungsgröße präziser wird. Im Vortrag also auch die Frage beantwortet werden, ob der Viterbi-Algorithmus ein paar wünschenswerte statistische Eigenschaften wie Konsistenz besitzt und unter welchen Bedingungen der Algorithmus besonders gut oder besonders schlecht funktioniert.

6 Problem 3: Schätzen der Modellparameter

Um das Modell mit der größten Wahrscheinlichkeit für eine gegebene Beobachtungssequenz zu finden, kann man versuchen, einen Maximum Likelihood Ansatz zum Laufen zu bringen. Die Idee ist dann für $\theta = \{\pi, Q, G\}$ die durch

$$L(\theta|X,Y) = \sum_X P(Y|X,\theta)P(X|\theta) = \sum_X \pi \prod_{t=2}^{T} P(x_t|x_{t-1}) \prod_{t=1}^{T} g_{x_t}(y_t)$$

(vgl. Gleichung 1) gegebene (log)-Likelihood zu maximieren, also

$$\theta^* \quad = \quad \arg\max_{\theta} \ \log\left(L(\theta|X,Y)\right).$$

Zu beachten ist, dass zur Lösung von Problem 1 und 2 bereits die Kenntnis (oder Schätzung) des Modells vorausgegangen ist. Die Lösung von Problem 3 ist also nicht ganz unwichtig.

Analytisch ist das Problem natürlich nicht zu lösen, uns fehlen ja die unbeobachteten Zustandswerte. Bei den Schlagwörten "Maximum Likelihood" in Kombination mit "fehlende Werte" springt auch schnell "EM-Algorithmus" in den Kopf (Dempster, 1977), der zur Lösung von Problem 3 auch tatsächlich verwendet wird. Der ursprüngliche Algorithmus von Baum und Welch (1970) ist allerdings nicht sehr "statistikerfreundlich", da er nicht in typischer EM-Form geschrieben ist (man kannte den EM-Algorithmus damals quasi noch nicht "offiziell"), sondern auf Forward-Backward-Formulierungen aufbaut. Bilmes (1998) leitet die Parameter etwas typischer her (und auch etwas näher an Computerintensive Methoden I).

Mit Hilfe des EM-Algorithmus möchte man eine Likelihood $L(\theta, z)$ bezüglich θ maximieren, beobachtet aber nur $L(\theta, x)$, wobei x ein Teil des Vektors z ist (also z unbeobachtete Daten enthählt). Man führt, nach Wählen eines Startwerts $\theta^{(0)}$ für $i = 0$, den E- und M-Schritt bis zur Konvergenz aus:

- E-Schritt: $Q(\theta) = Q(\theta|\theta^{(i)}) = \mathbb{E}\left(\log(L(\theta, x)|z, \theta^{(i)})\right)$
- M-Schritt: Maximiere $Q(\theta)$ und erhalte die neue Schätzung $\theta^{(i+1)}$.

Die Q-Funktion ist in diesem Fall also

$$Q(\theta, \theta^{(i-1)}) \quad = \quad \sum_X \log\left(P(Y, X|\theta)\right) P(Y, X|\theta^{(i-1)}),$$

bzw. mit (vgl. Gleichung (1))

$$P(Y, X|\theta) \quad = \quad \pi \prod_{t=1}^{T} P(x_t|x_{t-1}) g_{x_t}(y_t)$$

kann die Q-Funktion auch nach Auflösen des Logarithmus geschrieben werden als

$$\begin{aligned} Q(\theta, \theta^{(i-1)}) \quad &= \quad \sum_X \log(\pi) \prod_{t=1}^{T} P(Y, X|\theta^{(i-1)}) \\ &+ \quad \sum_X \left(\sum_{t=2}^{T} \log(P(X_{t-1}|x_t)) \right) P(Y, X|\theta^{(i-1)}) \\ &+ \quad \sum_X \left(\sum_{j=1}^{T} \log(g_{x_j}(y_j)) \right) P(Y, X|\theta^{(i-1)}) \end{aligned}$$

Der erste Summand dagegen kann vereinfacht werden zu

$$\sum_X \log(\pi) P(Y, X|\theta^{(i-1)}) \quad = \quad \sum_X \log(\pi_i) P(Y, X_0 = x_i|\theta^{(i-1)})$$

Leitet man nun nach π_i ab und bestimmt die Extremstellen unter der Einschränkung, dass $\sum_{i=1}^{K} \pi_i = 1$ ergibt sich

$$\hat{\pi}_i \quad = \quad \frac{P(Y, X_0 = x_i|\theta^{(i-1)})}{P(Y|\theta)}.$$

Der zweite Summand kann auch geschrieben werden als

$$\sum_{i=1}^{K} \sum_{j=1}^{K} \sum_{t=2}^{T} \log(a_{ij}) P(Y, X_{t-1} = x_i, X_t = x_j|\theta^{(i-1)}).$$

Leitet man diesen Ausdruck nun partiell nach a_{ij} ab und berücksichtigt wieder die Einschränkung $\sum_{j=1}^{K} a_{ij} = 1$, erhält man

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T} P(Y, X_{t-1} = x_i, X_t = x_j | \theta^{(i-1)})}{\sum_{t=1}^{T} P(Y, X_{t-1} = x_i | \theta^{(i-1)})}$$

Analog erhält man auch einen Ausdruck für die Emissionsmatrix, g_i mit $\sum_{m=1}^{M} g_i(m)$:

$$\hat{g}_i(m) = \frac{\sum_{t=1}^{T} P(Y, X_t = x_i | \theta^{(i-1)}) \mathbb{I}_{y_t=m}}{\sum_{t=1}^{T} P(Y, X_t = x_i | \theta^{(i-1)})}$$

Um aber etwas den Bogen zu den vorhergehenden Abschnitten zu schlagen, können diese Variablen auch über die Forward- und Backward-Variablen $\alpha_t(i)$ und $\beta_t(i)$ geschrieben werden:

$$\begin{aligned} \hat{\pi}_i &= \gamma_i(0) \\ \hat{a}_{ij} &= \frac{\sum_{t=1}^{T} p_{t-1}(i,j)}{\sum_{t=1}^{T} \gamma_i(t-1)}, \end{aligned}$$

mit

$$p_t(i,j) = \frac{\alpha_t(i) a_{ij} g_i(y_{t+1}) \beta_{t+1}(j)}{\sum_{k=1}^{K} \alpha_t(k) \beta_t(k)}$$

$\sum_{t=1}^{T} p_t(i,j)$ kann als die erwartete Anzahl von Übergängen von x_i nach x_j bei der Realisierung von Y interpretiert werden. Natürlich kann auch der Schätzer für die Emissionsmatrix mit diesen Variablen geschrieben werden:

$$\hat{g}_i(m) = \frac{\sum_{t=1}^{T} \gamma_i(t) \mathbb{I}_{y_t=k}}{\sum_{t=1}^{T} \gamma_i(t)}$$

Nach diesem Block ist es natürlich eher interessant, ob das ganze überhaupt funktioniert. Ein (zumindest funktionierender) R-Code ist im Anhang zu finden. Bezogen auf das unfaire Kasino möchte man also herausfinden, nach welchem System die Würfel getauscht werden und auch welcher Wahrscheinlichkeitsverteilung man bei welchem Würfel ausgesetzt ist. Nach etwa 100 dokumentierten Würfelwürfen kommt es einer Besucherin etwas merkwürdig vor und sie geht richtigerweise davon aus, dass ein zweiter, unfairer Würfel zum Einsatz kommt. Spontan rechnet sie einen Baum-Welch Algorith-

mus und erhält Schätzungen für Q und g.

$$\begin{aligned}
\hat{Q} &= \begin{pmatrix} 0.92 & 0.08 \\ 0.12 & 0.88 \end{pmatrix} \\
\hat{g} &= \begin{pmatrix} 0.14 & 0.07 & 0.00 & 0.02 & 0.03 & 0.47 \\ 0.12 & 0.19 & 0.09 & 0.02 & 0.0 & 0.38 \end{pmatrix}
\end{aligned}$$

Ganz von ihrem Verdacht möchte sie nicht ablassen und wartet, bis sie eine Sequenz von 1000 Würfelwürfen beobachtet hat, da sie vermutet, dass dadurch ihre Schätzung verbessert wird. Sie erhält hier

$$\begin{aligned}
\hat{Q} &= \begin{pmatrix} 0.68 & 0.32 \\ 0.22 & 0.78 \end{pmatrix} \\
\hat{g} &= \begin{pmatrix} 0.18 & 0.20 & 0.10 & 0.25 & 0.20 & 0.07 \\ 0.08 & 0.08 & 0.10 & 0.04 & 0.09 & 0.60 \end{pmatrix}
\end{aligned}$$

und fühlt sich in ihrem Misstrauen bestätigt. Aus statistischer Sicht wäre die Frage natürlich offen, inwiefern diese längere Sequenz tatsächlich die Schätzung verbessert oder inwiefern man sagen kann, dass sich die Zeilen der Emissionsmatrix unterscheiden. Das wird in diesem Handout aber nicht mehr diskutiert, sondern im Vortrag.

7 Erweiterungen

Da seit dem ersten Aufkommen von HMMs in den 60ern schon einiges an Zeit vergangen ist und in diesem halben Jahrhundert zum einen die Rechenkapazitäten unvernachlässigbar stark gestiegen sind und zum anderen die Menge an verfügbaren Daten(sätzen) jede Vorstellungskraft sprengt, kann man vermuten, dass diese Entwicklung auch einen Einfluss auf HMMs hatte. Insbesondere können jetzt natürlich Algorithmen implementiert werden, die einen deutlich größeren Rechenaufwand voraussetzen. Eine Möglichkeit, an die man denken könnte, ist ein bayesianischer Ansatz. Hier scheint insbesondere die Arbeit von Teh et al. (2006) ziemlich eingeschlagen zu sein. Mit Hilfe eines nichtparametrisch bayesianischen[1] Ansatzes kommt man hier tatsächlich zu einem HMM mit einer unendlichen Anzahl möglicher Zustände. Das bedeutet also nichts anderes als mehr Datennähe. Die Anzahl der Zustände muss nicht im Vorfeld festgelegt

[1] Die Formulierung wirkt eher wie ein Oxymoron wie beispielsweise "herrenloses Damenfahrrad" oder "Brennholzverleih". Letztendlich ist damit nur gemeint, dass der mögliche Parameterraum unendlichdimensional ist, das Modell aber nur eine begrenzte Anzahl an Parametern besitzt.

werden, sondern wird anhand der Daten geschätzt. Der wichtigste Baustein im hier vorgestellten Ansatz ist der hierarchische Dirichletprozess (HDP).

Der "normale" Dirichletprozess kann sinnvoll in Modellen angewendet werden, in denen eine Modellkomponente eine Zufallsvariable mit unbekannter Kardinalität ist, beispielsweise ein Clusterindikator in einer Mischverteilung. Der *hierarchische* Dirichletprozess wird bei mehreren Gruppen an Daten sinnvoll, wenn jedes Modell für eine Datengruppe selbst eine diskrete Zufallsvariable unbekannter Kardinalität besitzt und man diese Variablen über die Gruppen binden möchte. Hier kann man sich wieder eine Mischverteilung vorstellen, bei der man versucht, Cluster über mehrere Clusterprobleme zu verwenden. Teh et al. (2006) verwenden hier folgendes Beispiel: Versucht man Artikel eines wissenschaftlichen Magazins in Themen zu gruppieren, beispielsweise in "Lebensdaueranalyse" oder "Maschinelles Lernen", dann könnte es von Interesse sein, welche eventuell nicht explizit gemachten (latenten), aber gemeinsam behandelten Themen in diesen Gruppierungen zu finden wären.

Der Dirchletprozess $DP(\alpha_0, Q_0)$ ist im Grunde ein Maß über Maße mit dem Konzentrationsparameter $\alpha_0 \in \mathbb{R}^+$ und einer Wahrscheinlichkeitsverteilung Q_0. Q_0 wird selbst dann diskret sein, wie über verschiedene Repräsentationen gezeigt werden kann, nachzulesen beispielsweise im Skript zu Schätzen und Testen II, S. 82. Q_0 kann aber selbst als $Q_0 \sim DP(\gamma, H)$ angesehen werden, insofern ergibt sich recht natürlich eine hierarchische Struktur.

Zwei Fragen sind jetzt natürlich: Was hat das mit HMMs zu tun und falls es damit was zu tun hat, warum verwendet man den hierarchischen Dirichlet-Prozess und nicht den normalen? Zum einen kann ein HMM als eine Menge endlicher Mischverteilungen gesehen werden, quasi eine für jeden Wert eines Zustandes x_t. Die Übergangswahrscheinlichkeit $P(x_t|x_{t-1})$ kann damit als Mischverhältnis interpretiert werden, g als Mischkomponente. Wenn man nun jeweils ein DP-Mischmodell verwenden möchte, muss man aber beachten, dass man das HMM dann aber als Menge von DP-Mischmodellen betrachten muss, also ein DP-Mischmodell für jeden Wert des momentanen Zustands. Werden diese Mischmodelle nicht in irgendeiner Weise verbunden, wäre die Menge an Zuständen in einem Wert eines bestimmten Zustands disjunkt von der Menge, die verfügbar wäre, wäre man in einem anderen Zustand zum gleichen Zeitpunkt. Das Resultat wäre eine Baum- und keine Kettenstruktur. Eine hierarchische Struktur verhindert das Problem.

Um es formal zusammenzufassen: Wir betrachten hier eine Menge an Übergangsmatrizen (bzw. eher Übergangsverteilungen) gezogen aus einem HDP, also

$$\begin{aligned} Q_0|\gamma, H &\sim DP(\gamma, H) \\ Q_y|\alpha, Q_0 &\sim DP(\alpha, Q_0) \text{ für } x \in X \end{aligned}$$

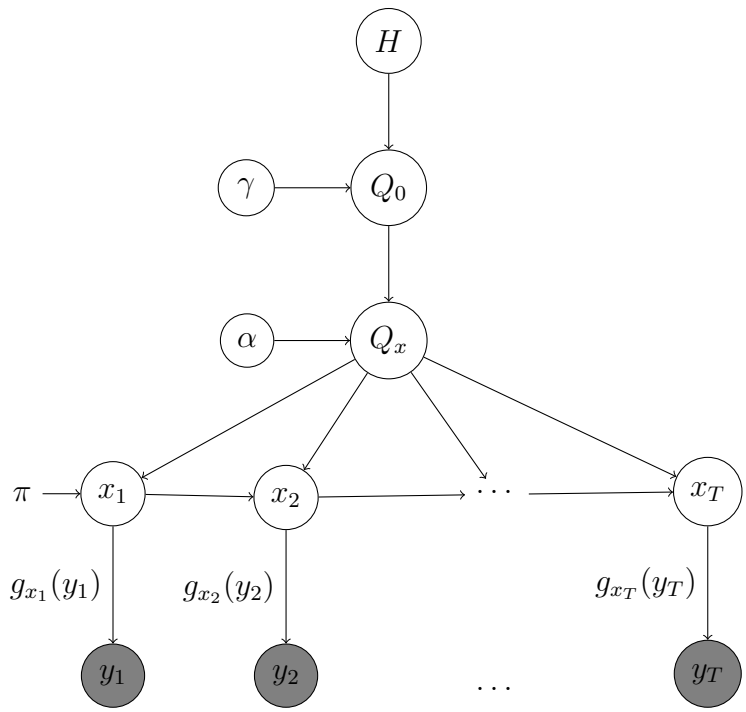

Abbildung 2: Struktur eines HDP-HMMs.

mit H als Basisverteilung. Q_0 erlaubt, dass bei jedem Übergang auch alle Zustände wieder erreicht werden können. Die bedingten Verteilungen der Sequenz der latenten Variablen $x_1, ..., x_T$ und die der beobachteten Variablen $y_1, ..., y_T$ sind

$$\begin{aligned} x_t | x_{t-1}, Q_{y_{t-1}} &\sim Q_{y_{t-1}} \quad t = 1, ..., T \\ y_t | x_t &\sim g_{x_t}(y_t) \end{aligned}$$

In graphischer Form kann das Modell wie in Abbildung 2 dargestellt werden.

Dieses Modell ist letztendlich ein Modell mit einer nichtzählbaren Anzahl an Dirichlet-Prozessen, was letztendlich aber nur ein theoretisches Problem ist, da die Pfade eines HDP-HMMs letztlich auch endlich sind. Allerdings erhöht sich durch diesen HMM-Ansatz der Rechenaufwand enorm (und auch die Programmierschwierigkeit...), insofern sollte sich der Aufwand auch in besseren Schätzungen niederschlagen. Das tut er. Je größer die wahre Anzahl der versteckten Zustände, desto stärker sind parametrische HMMs "verwirrt", schätzen also schlechter. Der "Verwirrungsgrad" der HDP-HMMs ist unabhängig von der wahren Anzahl der Zustände, demonstriert bei Teh et al. (2006), also wird Problem 2 besser gelöst.

8 Diskussion

Im Vortrag

Literatur

Bau, L., Petrie, T., Soules., G. und N. Weiss. 1970. A maximization technique occuring in the statistical analysis of probabilitstic functions of markov chains. *The Annuals of Mathematical Statistics*, 41(1):164 – 171.

Durbin, R., Eddy, S., Krogh, A. und G. Mitchison. 1998. *Biological Sequence Analysis*, Cambridge, UK: Cambridge University Press.

Goldwater, S., Griffiths, T.L. und M. Johnson. 2006. Contextual dependencies in unsupervised word segmentation. In *Processdings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*.

Huang, S. und S. Renals. 2007. *Spoken Language Processing*. Prentice-Hall, Upper Saddle River, NJ.

Jelinek, F. 1969. A fast sequential decoding algorithm using a stack. *IBM Journal of Research Developments*, 13, 675–685.

Rabiner, R. 1989. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257 –286.

Teh, Y., Jordan, M., Beal, M. und D. Blei. 2006. Hierarchical dirichlet processes. *Journal of the American StatisticalAssociation*, 101(476):1566–1581.

Yau, C., Papaspiliopoulos, O., Roberts, G. und C. Holmes. 2011. Bayesian nonparametric hidden Markov models with applications in genomics. *Journal of the Roxal Statistical Society: Series B*, 73(1):37–57.

Anhang

Um die Algorithmen etwas besser nachvollziehen zu können, sind hier die (nicht notwendigerweise gut programmierten) R-Codes zur Forward-/Backward-Methode, zum Viterbi- und Baum-Welch Algorithmus angegeben. Die Codes kann ich gern auch per Email senden (david.bauder@gmx.de).

8.1 Viterbi-Algorithmus

```
## Bastle Zutaten:

observations <- c("1", "2", "3", "4", "5", "6")

states <- c("Fair", "unfair")
Fairprobs <- c(0.8, 0.2)
Unfairprobs <- c(0.1, 0.9)
transitionmatrix <- matrix(c(Fairprobs, Unfairprobs), 2,2,byrow=T)
rownames(transitionmatrix) <- states
colnames(transitionmatrix) <- states
transitionmatrix

Fairstate <- c(1/6, 1/6, 1/6, 1/6, 1/6, 1/6)
Unfairstate <- c(0.1, 0.1, 0.1, 0.1, 0.1, 0.5)
emissionmatrix <- matrix(c(Fairstate, Unfairstate), 2,6,byrow=T)
rownames(emissionmatrix) <- states
colnames(emissionmatrix) <- observations
emissionmatrix




hmmseq <- function(states, # Vektor der mglichen Zustnde
                   transitionmatrix, # bergangsmatrix der Zustnde
                   emissionmatrix, # Emissionsmatrix
                   inprobs, # Ausgangswahrscheinlichkeiten
                   seqlength # Lnge der generierten Sequenz(en)
                   ){
  observations <- observations
  states <- states
  myseq <- character()
  mystates <- character()

  # Auswhlen des ersten Zustandes:
  firststate <- sample(states, 1, rep=T, prob=inprobs)
  probs <- emissionmatrix[firststate,]
```

```
  # Auswhlen des ersten observations gegeben dem ersten State
  firstnuc <- sample(observations, 1, rep=T, prob=probs)
  myseq[1] <- firstnuc
  mystates[1] <- firststate

  for(i in 2:seqlength){
    laststate <- mystates[i-1]
    stateprobs <- transitionmatrix[laststate,]
    state <- sample(states, 1, rep=T, prob=stateprobs)
    probs <- emissionmatrix[state,]
    nuc <- sample(observations, 1, rep=T, prob=probs)
    myseq[i] <- nuc
    mystates[i] <- state
  }
  return(data.frame(myseq, mystates))
}

initialprobs <- c(0.5, 0.5)
T <- 100
set.seed(42)
seq <- hmmseq(states, transitionmatrix, emissionmatrix, initialprobs, T)
seq
nucsequence <- as.vector(seq[,1])
statesequence <- as.vector(seq[,2])



## Viterbi
#(cran.r-project.org/doc/contrib/Krijnen-IntroBioInfStatistics.pdf, S.209)

viterbi <- function(sequence, transitionmatrix, emissionmatrix){
  # Zustandsnamen:
  states <- rownames(emissionmatrix)

  #Viterbimatrix:
  v <- viterbimatrix(sequence, transitionmatrix, emissionmatrix)

  probablestatepath <- apply(v, 1, function(x) which.max(x))

  # Ausgabe:
  lastnuc <- sequence[1]
  lastprobablestate <- probablestatepath[[1]]
  lastprobablestatename <- states[lastprobablestate]
  startpos <- 1
  for(i in 2:length(sequence)){
    nuc <- sequence[i]
    probablestate <- probablestatepath[[i]]
    probablestatename <- states[probablestate]
```

```
    if(probablestatename != lastprobablestatename){
      print(paste("Position", startpos,"-", (i-1),
                  "Wahrscheinlichster Zustand: ", lastprobablestatename))
      startpos <- i
    }
    lastnuc <- nuc
    lastprobablestatename <- probablestatename
  }
  print(paste("Position", startpos, "-", i, "Wahrscheinlichster Zustand: ",
              lastprobablestatename))
}


viterbimatrix <- function(sequence, transitionmatrix, emissionmatrix){
  sequence <- sequence
  numstates <- dim(transitionmatrix)[1]
  v <- matrix(NA, nrow=length(sequence), ncol=dim(transitionmatrix)[1])
  v[1,] <- 0
  v[1,1] <- 1

  for(i in 2:length(sequence)){
    for(j in 1:numstates){
      statejprobobservationi <- emissionmatrix[j, sequence[i]]
      v[i,j] <- statejprobobservationi * max(v[(i-1),]*transitionmatrix[,j])
    }
  }
  return(v)
}

viterbi(nucsequence, transitionmatrix, emissionmatrix)
```

8.2 Baum-Welch-Algorithmus

```
## Forward-Variable
# Inputs:
# - y: Beobachtungsvektor
# - pi: "initial distribution"
# - Q: bergangsmatrix
# - g: Emissionsmatrix
#
# Output:
# phi := P(X_t = x |y_{1:t})
# c(t) := P(Y_t = y_t | Y_{1:t-1}=y_{1:t-1})

Forward <- function(y, pi, Q, g){
  N <- length(y)
  dims <- dim(Q)
```

```
  result <- list()
  result$phi <- matrix(nrow=dims[1], ncol=N)
  result$c <- array(0,N)

  alpha <- pi*g[,y[1]]
  result$c[1] <- sum(alpha)
  result$phi[,1] <- alpha/result$c[1]

  for (t in 2:N){
    H <- (result$phi[,t-1]%*%Q)*g[,y[t]]
    result$c[t] <- sum(H)
    result$phi[,t] <- H/result$c[t]
  }
  result
}


## Backward-Variable
# Inputs:
# - y: Beobachtungsvektor
# - Q: bergangsmatrix
# - g: Emissionsmatrix
# - c: bedingte Likelihoods
#
# Outputs:
# beta = P(y_{t+1:n} | X_t = x) / P(y_{t+1:n} | y_{1:t}

Backward <- function(y, Q, g, c){
  N <- length(y)
  dims <- dim(Q)
  beta <- matrix(1, nrow=dims[1], ncol=N)
  for(t in seq(N-1,1,-1)){
    beta[,t]=Q%*%(g[,y[t+1]]*beta[,t+1])/c[t+1]
  }
  beta
}


## Baum-Welch Algorithmus
# Inputs:
# - y: Beobachtungsvektor
# - pi: Ausgangswahrscheinlichkeiten
# - tol: Stoppkriterium
# - maxiter: Erlaubte Anzahl an Iterationen
#
# Outputs:
# Q: Schtzung fr die bergangsmatrix
# g: Schtzung fr die Emissionsmatrix
```

```
# ll: log-likelihood von y f r Q und g

Baumwelch <- function(y, pi, tol=0.00001, maxiter = 1000){
  N <- length(y)
  k <- length(pi)
  M <- max(y)

  Y <- matrix(0, nrow=N, ncol=M)
  for(t in 1:N){
    Y[t, y[t]]=1
  }

  Q <- matrix(runif(k*k), nrow=k)
  Q <- Q/apply(Q,1,sum)

  g <- matrix(runif(k*M), nrow=k)
  g <- g/apply(g,1,sum)

  iter <- 0
  Q.alt <- Q-tol
  g.alt <- g-tol

  while ((sum(abs((Q.alt-Q))) + sum(abs(g.alt-g)) > tol) & (iter < maxiter)){
    iter <- iter+1
    forward <- Forward(y, pi, Q, g)
    backward <- Backward(y,Q,g,forward$c)
    posteriori <- forward$phi*backward

    L <- Q*(forward$phi[,1:(N-1)]%*%t(backward[,2:N]*g[,y[2:N]]/
                    (matrix(1,nrow=k,ncol=1)%*%forward$c[2:N])))
    R <- posteriori%*%Y

    Q.alt <- Q
    g.alt <- g
    Q <- L/apply(L,1,sum)
    g <- R/apply(R,1,sum)
  }
  result <- list()
  result$Q <- Q
  result$g <- g
  result$ll <- sum(log(forward$c))
  result$iter <- iter
  result
}

# Testen, nachdem Sequenz  ber HMMseq generiert wurde
pi <- c(.5, .5)
Baumwelch(as.numeric(nucsequence), pi, tol=0.000001, maxiter=10000)
```