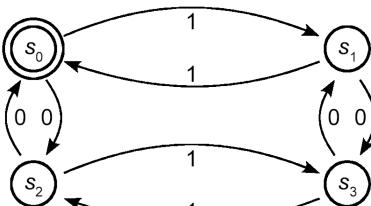


Formal:

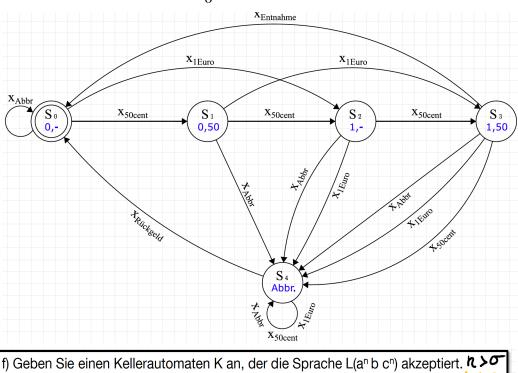
$$Z = (\mathcal{S}, \mathcal{X}, \mathbf{T}, s_0, \mathcal{F})$$

$\mathcal{S}$  ein Set mit einer endlichen Anzahl von Zuständen  
 $\mathcal{X}$  zulässiges Alphabet für die zu verarbeitende Symbolfolge  $X$   
 $\mathbf{T}$  Transitionsfunktionen für die Zustände in  $\mathcal{S}$   
 $s_0$  Anfangszustand  
 $\mathcal{F}$  ein Set von festgelegten Endzuständen



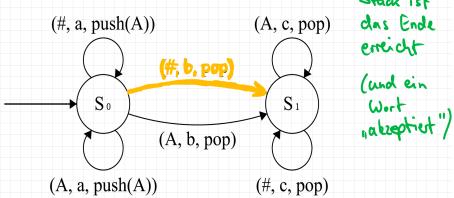
Transitionsfunktion als Regel:  
 $t(s^-, x_i) = s^+$ ,

Beispiel:  $\mathcal{S} = \{s_0, s_1, s_2, s_3\}$   
 $\mathcal{X} = \{0, 1\}$   
 $\mathcal{F} = \{s_0\}$

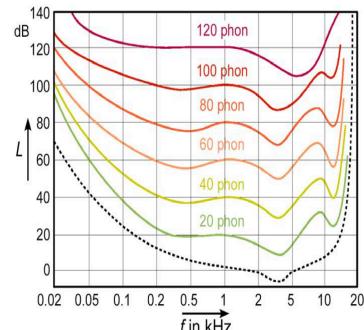
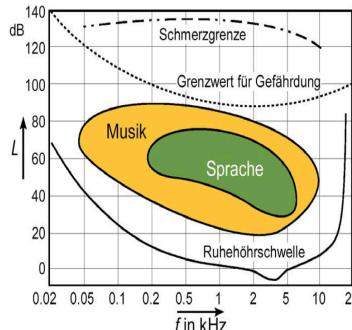


- f) Geben Sie einen Kellerautomaten K an, der die Sprache  $L(a^n b^n c^n)$  akzeptiert.  $n > 0$   
Es sei vorgegeben:  $\mathcal{V} = \{\#, a, b, c\}$   $y_0 = \#$   $\mathcal{F} = \{\#\}$  d.h.: bei leerem  
1. Markiere Zustände mit Nichtterminalsymbolen  
2. Übertrage jede Transition in eine Produktionsregel  
3. Für jeden Endzustand: erstelle Regel mit leerem Wort rechts

- $P = \{\$ \rightarrow A, A \rightarrow aA, A \rightarrow bB, B \rightarrow b, B \rightarrow cC, C \rightarrow c\}$



aaa bccc  
b

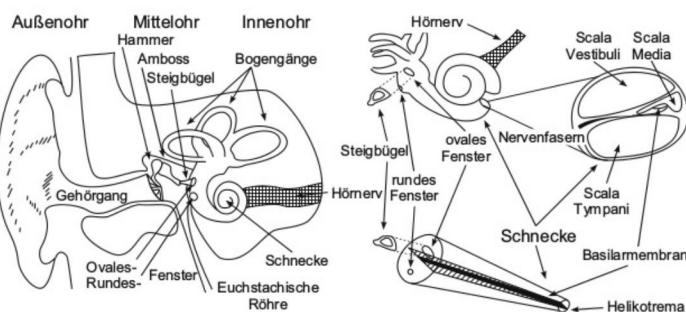


Psychoakustik		Physik	
Bezeichnung	Einheit	Bezeichnung	Einheit
Tonheit $Z$	Bark	Frequenz $f$	Hz
Verhältnistonhöhe $V$	mel	Schalldruck $p$	$\frac{N}{m^2} = \frac{W s}{m^3} = Pa$
Lautstärkepegel $L_a$	Phon	Schallschnelle $v$	$\frac{m}{s}$
Lautheit $N$	sone	Schallintensität $I$	$\frac{W}{m^2} = \frac{N}{s \cdot m}$
		Schalldruckpegel $L$	dB
		Schalleistung $P_{ak}$	$W = \frac{N \cdot m}{s}$
		Bezugsschalldruck $p_0 = 2 \cdot 10^{-5} \frac{N}{m^2} = 20 \mu Pa$ ,	
		Bezugsintensität $I_0 = 1.0 \cdot 10^{-12} \frac{W}{m^2}$	

### 3.3 Ohr

• 20Hz-20kHz Schallreize nicht linear

• **Physik ↔ Psychoakustik:** Frequenz↔Tonheit/Verhältnistonhöhe, Schalldruckpegel(1kHz, 40db)↔Lautstärkepegel(1kHz, 40Phon)/Lautheit(sone, linear mit Empfindung)



Wichtigste Teile:

• Trommelfell→Gehörknochen→Schnecke:

– Basilmembran: Haarzellen Schwingung→Nervenimpulse. Frequenz-Ort Umwandlung

#### 3.3.1 Frequenzgruppen

• 24 Frequenzgruppen auf der Basilmembran

#### 3.3.2 Verdeckungen

• Spektrale: Mithörschwelle angehoben auch in der Umgebung der Frequenz vom Maskierer. Je lauter, desto breiter die Verdeckung ist.

• Zeitliche: Vorverdeckung, weiß man nicht warum. Nachverdeckung, wegen dem Abklingverhalten der Haarzellen. Benutzt im MP3 Datenreduktion.

### 4 SUCHE

Grundlage ist die Darstellung eines Problems in einem Zustandsraum (state space), meist Baumstruktur. Dieser wird mit Beginn der Suche zur besseren Effektivität beschnitten (pruning). Unterschiede zwischen den Algorithmen: Art der Queue und Update-Funktion. **Beim Aufstellen beachten:** Kosten addieren sich auf vom Start zum Ziel

#### 4.1 Einfache Suchverfahren

- Depth First Immer zuerst in die Tiefe (LIFO, wie in einem Labyrinth)
- Breadth First Zuerst alle auf einer Ebene bevor man tiefer geht (FIFO)
- Dijkstra (~ Uniform-Cost): Nächster Knoten ist Knoten mit minimalen Kosten (wie  $A^*$  mit  $h(n) = 0$ )

## 4.2 Heuristische Suche

- **A-Algorithmus:**  $f(n) = g(n) + h(n)$  (Priority Queue)
- **A\* (A-Star):**  $f(n) = g(n) + h(n)$  mit  $0 \leq h(n) \leq h^*(n)$  (heuristische Kosten als priority)
  - $h^*(n)$  = exakte Kostenfkt,  $h(n)$  = aktuelle Schätzung (monoton)
  - $A^*$  ⇒ Heuristik schätzt immer kleiner gleich als tatsächliche Kosten
  - wenn  $h(n) = h^*(n)$ , dann wird der kürzeste Weg sofort gefunden

**Listendarstellung für A\*:**  $L_i = \begin{bmatrix} s_k(s_{i-1} \dots s_0) : \text{Kosten}_k || \\ s_j(s_{i-1} \dots s_0) : \text{Kosten}_j || \end{bmatrix}$ ,  $k$  und  $j$  sind mögliche folge Zustände von  $i - 1$ , welche evaluiert werden müssen. Beginnend in start und dann aufnehmen aller folge Knoten aus dem Suchbaum.

## 5 PRÄDIKATENLOGIK UND LOGISCHES SCHLIESSEN

Wissen algorithmisch verarbeitbar ⇒ aus bestehenden Fakten neue Fakten ableiten oder Behauptungen beweisen.

### 5.1 Logiken

- **Aussagenlogik:** nur wahr/falsch, verknüpfbar mit:

- · (UND, Konjunktion); + (ODER, Disjunktion)
- $\neg$  (NICHT, Negation);  $\Rightarrow$  (Implikation)
- Überprüfbar mit Wahrheitstabelle

- **Prädikatenlogik:** Analyse & Bewertung von Prädikaten die Beziehung zwischen Objekten und Variablen beschreiben, sowie deren logische Verknüpfung. Prädikate sind Elementaraussagen

- Existenz-Quantor,  $\forall$  All-Quantor
- führt zu einem Problemorientierten Regelwerk den Axiomen, Behauptungen (Theorem) erfragbar, Antwort darauf wird mittels logischen Schließen generiert. Zu viele Axiome für Wahrheitstabelle.
- Umformungsregeln mit  $(\star, \diamond) \in \{\cdot, +, \wedge, \diamond, \neq, *\}$ :
  - $A \equiv \neg(\neg A)$  Doppelte Negation
  - $A + A = A$  und  $A \cdot A = A$  Idempotenz
  - $A \cdot B = B \cdot A$  und  $A + B = B + A$  Kommutativ
  - $A \star (B \star C) = (A \star B) \star C$ ,  $\star \in \{\cdot, +\}$  Assoziativität
  - $A \diamond (B \star C) = (A \diamond B) \star (A \diamond C)$  Distributivität
  - $\neg(A \diamond B) \equiv \neg A \star \neg B$  De Morgan
  - $A \Rightarrow B \equiv \neg B \Rightarrow \neg A$  Kontraposition
  - $A \Rightarrow B \equiv \neg A + B$
  - $A \Leftrightarrow B \equiv (A \Rightarrow B) \cdot (B \Rightarrow A) \equiv (A \cdot B) + (\neg A \cdot \neg B)$
  - $\neg(\forall x) A(x) \equiv (\exists x) (\neg A(x))$
  - $\neg(\exists x) A(x) \equiv (\forall x) (\neg A(x))$
  - $(\forall x)(A(x) \cdot B(x)) \equiv (\forall x) A(x) \cdot (\forall y) B(y)$

- Disjunktive Normalform (DNF, Disjunktion von Konjunktionstermen ( $\cdot$ ) + ( $\cdot$ ): Umwandlung zu KNF über De Morgan Regel (siehe 6. oben))
- Konjunktive Normalform (KNF, Konjunktion von Disjunktionstermen ( $+$ ) · ( $+$ )):
  1. Eliminierung aller Äquivalenzen (Regel 9)
  2. Eliminierung aller Implikationen (Regel 8)
  3. Negation nach innen (Regeln 6, 10, 11)
  4. Neue Variablen für jeden Quantifizierer (nur einmal  $\forall x$  oder  $\exists x$  für alle Variablen)
  5. Eliminierung aller Existenz-Quantoren (Ersetzen mit Skolemfkt.)
  6. Ausklammern der All-Quantoren & Entfernen dieser
  7. Distributivgesetz zur Transformation in KNF (Regel 5)
  8. Jeder "Und-Verknüpfe" Teil wird nun als eigene Klausel betrachtet
  9. Jede Klausel bekommt ihre eigenen Variablen

- **Tautologie: Beweis durch Widerspruch:** 1. Negiere Formel 2. Bringe Formel in KNF 3. Zeige Kontradiktion ⇒ Resolutionsverfahren

- **Resolution:** Verknüpfung der Klauseln aus KNF, dabei werden komplementäre Literale eliminiert. Beispiel:  $(A + B)$  verknüpft mit  $(\neg B)$  ergibt  $(A)$ . Der Beweis ist vollendet, falls die Resolution die leere Klausel ergibt. Andernfalls ist der Beweis gescheitert.

## 6 GRAMMATIKEN

- erste linguistische Verwendung von Grammatiken: 1903, W. Wundt, heutige Form 1956 N.Chomsky
- Nutzung in Informatik für Programmiersprachen und Compilerbau. In KI vor allem in Computerlinguistik, d.h. natürlichsprachliche Systeme und Dialogbildung
- Beschreibung einfacher natürlicher Sätze:  $<\text{Satz}> \rightarrow <\text{Hauptsatz}> <\text{Nebensatz}>$ ,  $<\text{Hauptsatz}> \rightarrow <\text{Artikel}> <\text{Hauptsatz}>$ ,  $<\text{Hauptsatz}> \rightarrow <\text{Nomen}>$ ,  $<\text{Nebensatz}> \rightarrow <\text{Verb}>$  usw.

## 6.1 Kontextfreie Grammatiken

CFG (Context Free Grammar) wird durch das Set  $\mathcal{G} = \{V, T, P, S\}$  beschrieben mit  $V \equiv$  Variable (<Ausdruck>, Großbuchstabe);  $T \equiv$  Terminalie ("x", Kleinbuchstaben);  $P \equiv$  Produktionsregel (<Ausdruck>  $\rightarrow$  "x"  $x \in V \cup T$ ;  $A \rightarrow x|y|z$ );  $S \equiv$  Startsymbol.

### 6.2 Normalformen von Grammatiken

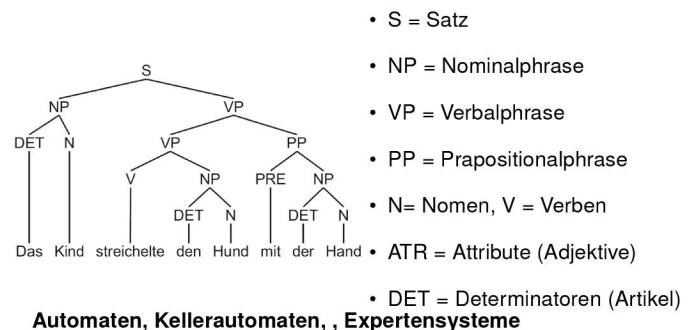
- **Chomsky-Normalform:** (CNF) Nur Produktionsregeln, bei denen auf der rechten Seite entweder nur zwei Variablen oder ein terminaler Ausdruck stehen, also:  $A \rightarrow BC$  oder  $A \rightarrow a$  mit  $A, B, C \in V$  und  $a \in T$

Umwandlung in CNF durch Ersetzung von Terminalen bzw. Ersetzung von überzähligen Variablen durch zulässige Produktionsregeln:

- $S \rightarrow aB \Rightarrow S \rightarrow A_1 B$ ,  $A_1 \rightarrow a$
- $S \rightarrow aBB \Rightarrow S \rightarrow A_1 B_1$ ,  $B_1 \rightarrow BB$ ,  $A_1 \rightarrow a$

- **Backus-Naur-Form:** Nichtterminalsymbole in spitzen Klammern  $<>$ ; Alternativen wie in CNF durch senkrechten Strich „|“; Sequenzen lassen sich durch Terminalsymbole und Nichtterminalsymbole darstellen; Wiederholungen werden durch Rekursionen dargestellt:  $<A> \rightarrow a | a <A>$

- **Parsing:** Produktionsregeln der Grammatik so lange anwenden, bis alle Variablen aus  $V$  durch terminale Symbole aus  $T$  ersetzt sind und der Satz nur noch aus terminalen Symbolen besteht. Kann als Baum dargestellt werden.



### 6.3 Dialoggestaltung:

Menüauswahl, Formular Ausfüllen, Kommandosprachen, Natürlichsprachlicher Dialog, direkte Manipulation, Multimediadialog

## 7 ABSTANDSKLASSIFIZIERUNG

- Distanz eines Mustervektors  $\mathbf{x}$  zu einer bestimmten Klasse  $k$  und Mittelwertsvektor  $\mathbf{m}_k$ :

$$d_k(\mathbf{x}, \mathbf{m}_k) = (\mathbf{x} - \mathbf{m}_k)^T \cdot \mathbf{W}_k \cdot (\mathbf{x} - \mathbf{m}_k)$$

- **Mittelwerte von Klassen:**  $\mathbf{m}_k = \frac{1}{M_k} \sum_{i=1}^{M_k} \mathbf{r}_{k,i}$

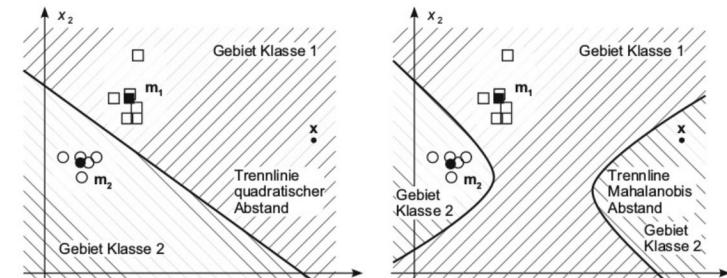
- Klassifizierung von  $\mathbf{x}$ :  $\hat{k}_x = \operatorname{argmin}_{1 \leq k \leq K} d_k(\mathbf{x}, \mathbf{m}_k)$

- Für den Namen des Klassifikators ist die Form der Gewichtsmatrix  $\mathbf{W}_k$  entscheidend.

- **Quadratischer (Euklidischer) Abstand:** mit  $\mathbf{W}_k = \mathbf{W} = \mathbf{W}^{-1} = \mathbf{I}$  als Einheitsmatrix. Somit vereinfacht sich der Abstand zum Skalarprodukt:  $d_k(\mathbf{x}, \mathbf{m}_k) = (\mathbf{x} - \mathbf{m}_k)^T (\mathbf{x} - \mathbf{m}_k)$

- **Mahalanobis-Abstand:** Inverse der Kovarianzmatrix für Distanzberechnung:  $\mathbf{W}_k = \mathbf{W}_{K,k}^{-1} = \mathbf{C}_k^{-1}$  mit  $\mathbf{W}_{K,k} = \left( \frac{1}{M_k} \sum_{i=1}^{M_k} \mathbf{r}_{k,i} \mathbf{r}_{k,i}^T \right) - \mathbf{m}_k \mathbf{m}_k^T$

- Die Trennfunktion für den Mahalanobis-Abstand sind Kegelschnitte, also Geraden, Ellipsen, Parabeln oder Hyperbeln.



## 8 HIDDEN MARKOV MODELLE

- können nicht nur Merkmalsvektoren, sondern ganze Sequenzen klassifizieren
- **Merkalsextraktion:** mittels Mel Frequency Cepstral Coefficient (MFCC) sind 12 Werte, die aus der spektral Anaylse der Fenster kommen
- in Spracherkennung: 1 HMM pro Wort bei Erkennung auf Wortebene (Einzelworterkennner) oder 1 HMM pro Phonem bei der Erkennung fließend gesprochener Sprache (auch Neuerkennung unbekannter Wörter möglich)
- „Finde diejenige Klasse, die die Beobachtung  $\mathbf{o} = (o_1, \dots, o_T)$  am besten nachbilden kann.“ d.h. in Sprachsystemen Variationen ausgleichen.
- **Markov-Modelle:** Grundlage der HMM, Markov-Ketten. Stochastische Prozesse, deren aktueller Zustand nur vom vorausgegangenen Zustand abhängt. Übergang der Zustände durch Pfeile gekennzeichnet. Die Matrix  $\mathbf{A}$  gibt die Übergangswahrscheinlichkeiten ab. Außerdem ist ein Startzustand  $q_1$  nötig. Im Allgemeinen kommt jeder Zustand in Frage; diese Wahrscheinlichkeiten werden im Vektor der Einsprungswahrscheinlichkeiten  $\mathbf{e} = [p(q_1 = s_1), \dots, p(q_1 = s_N)]^T$  zusammengefasst.

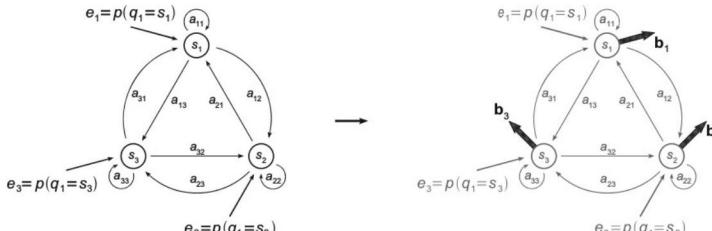
- **Hidden Markov Modelle:** Erweiterung: Jeder Zustand  $s_i$  kann mit einer Wahrscheinlichkeit  $b_{mi}$  eine Beobachtung  $v_m$  aus der Menge der möglichen Beobachtungen  $\mathbf{v}$  mit  $b_{mi} = p(v_m | s_i)$  „aussenden“.
- **Vollständige Beschreibung** mit  $\lambda = (\mathbf{e}, \mathbf{A}, \mathbf{B})$ ; mit der Beobachtungswahrscheinlichkeit  $p(\mathbf{o}|\lambda)$
- Von der Beobachtungsfolge  $\mathbf{o}$  kann i.A. nicht auf die durchlaufene Zustandsfolge  $\mathbf{q}$  geschlossen werden („hidden“).
- Übergangswahrscheinlichkeiten

$$\mathbf{A} = p\{q_{t+1} = s_j | q_t = s_i\} = \begin{bmatrix} a_{11} & \cdots & a_{1N} \\ \vdots & \ddots & \vdots \\ a_{N1} & \cdots & a_{NN} \end{bmatrix}$$

- Beobachtungswahrscheinlichkeiten

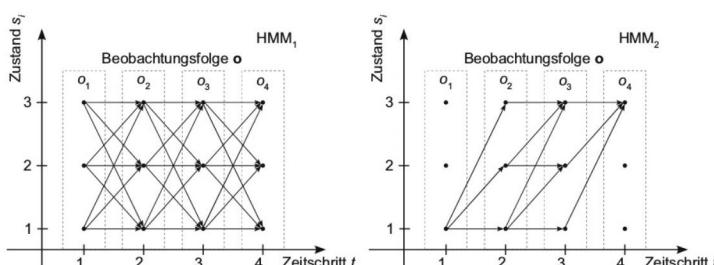
$$\mathbf{B} = \begin{bmatrix} p(v_1 | s_1) = b_{11} & \cdots & p(v_1 | s_N) = b_{N1} \\ \vdots & \ddots & \vdots \\ p(v_M | s_1) = b_{1M} = \mathbf{b}_1 & \cdots & p(v_M | s_N) = b_{NM} = \mathbf{b}_N \end{bmatrix}$$

- **Ergodisches HMM:**  $\mathbf{A}$  voll besetzt, HMM ergodisch, d.h. voll verbunden ist.
- **Links-Rechts-HMM:**  $\mathbf{A}$  hat obere Dreiecksform (oben rechts); es gibt keine Rücksprünge, nützlich für zeitliche Abfolgen, z.B. Sprachverarbeitung



links: Markov-Modell, rechts: Hidden-Markov-Modell

### 8.1 Trellis-Diagramme



ergodisches Modell (links); Links-Rechts-Modell (rechts).

- Alle möglichen zeitlichen Zustandsfolgen können damit dargestellt werden.

$$p(\mathbf{o}|\lambda_k) = \sum_{q \in Q} e_{q1} b_{q1}(o_1) \prod_{t=2}^T a_{qt-1} q_t b_{qt}(o_t)$$

1. „Unmögliche“ Zustände streichen
2. Alle möglichen Übergänge eintragen
3. Streichen aller eingehenden Kanten in einen Zustand ohne ausgehende Kanten (Sackgassen)

## 8.2 Vorwärts-Algorithmus

- Wie GKI Filtering aber ohne  $\alpha$  für Normalisierung
- Effizientere Berechnung der Wahrscheinlichkeit, Rechenoperationen:  $O P_v \approx T \cdot N^2$ . mithilfe der Vorwärtswahrscheinlichkeit  $\alpha_t = P(o_1, o_2, \dots, o_t, q_t = s_i | \lambda_k)$
- 1. Init.:  $\alpha_1(i) = e_i b_i(o_1)$ ,  $1 \leq i \leq N$
- 2. Indukt.:  $\alpha_{t+1}(j) = \left[ \sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(o_{t+1})$ ;  $1 \leq t \leq T-1$ ;  $1 \leq j \leq N$
- 3. Terminierung:  $P(\mathbf{o}|\lambda_k) = \sum_{i=1}^N \alpha_T(i)$

## 8.3 Baum-Welch-Algorithmus

- Benutzt zur Optimierung der  $\lambda_k = (\mathbf{e}, \mathbf{A}, \mathbf{B})$ , da es analytisch nicht möglich ist, mithilfe der Rückwärtswahrscheinlichkeit  $\beta_t$ .
- $\gamma_t(i) = p(q_t = s_i | \mathbf{o}, \lambda_k)$ ;  $\xi_t(i,j) = p(q_t = s_i, q_{t+1} = s_j | \mathbf{o}, \lambda_k)$ ;  $\sum_{t=1}^{T-1} \gamma_t(i) \equiv$  alle Aufenthalte im Zustand  $s_i$ ;  $\sum_{t=1}^{T-1} \xi_t(i,j) \equiv$  alle Übergänge vom Zustand  $s_i$  im Zustand  $s_j$
- Damit kann man  $\hat{\lambda}_k = (\hat{\mathbf{e}}, \hat{\mathbf{A}}, \hat{\mathbf{B}})$  schätzen und iterativ wieder berechnen (dies ist EM-Algorithmus).

### 8.3.1 Rückwärts-Algorithmus

- Rückwärtswahrscheinlichkeit  $\beta_t(i) = P(o_{t+1}, o_{t+2}, \dots, o_T | q_t = s_i, \lambda_k)$
- 1. Init.:  $\beta_T(i) = 1$ ,  $1 \leq i \leq N$

$$2. \text{ Indukt.: } \beta_t(i) = \sum_{j=1}^N a_{ij} b_j(o_{t+1}) \beta_{t+1}(j); t = T-1, T-2, \dots, 1$$

## 8.4 Viterbi-Algorithmus

- Kenntnis des wahrscheinlichsten Pfades; Änderung des Induktionsschrittes, der nicht die Summe aller Pfadwahrscheinlichkeiten, sondern nur die des wahrscheinlichsten Pfades mitführt. Dabei ist  $\Psi_t(j)$  der Pfad, d.h. die durchlaufene Zustandsfolge.
- Diese Implementierung des Viterbi-Algorithmus wird mit den Worten „Add, Compare, Select“ zusammengefasst (auch wenn die Addition in diesem Fall eine Multiplikation ist).
- 1. Init:  $\delta_1(i) = e_i b_i(o_1)$  and  $\Psi_1(i) = 0$ , for  $1 \leq i \leq N$
- 2. Induktion:

$$\delta_t(j) = \max_{1 \leq i \leq N} [\delta_{t-1}(i) a_{ij}] \cdot b_j(o_t); 2 \leq t \leq T; 1 \leq j \leq N$$

$$\Psi_t(j) = \operatorname{argmax}_{1 \leq i \leq N} [\delta_t(i) a_{ij}]; 2 \leq t \leq T; 1 \leq j \leq N$$

$$3. \text{ Terminierung: } P^* = \max_{1 \leq i \leq N} [\delta_T(i)], q_T^* = \operatorname{argmax}_{1 \leq i \leq N} [\delta_T(i)]$$

- 4. Ermittlung der wahrscheinlichsten Zustandsfolge:

- Einfach die Zeile in  $\Psi_N$  auswählen, die in  $\delta_N$  den größten Eintrag hat. Der enthaltene Wert bestimmt die Zeile in  $\Psi_{N-1}$ , die den nächsten Wert gibt für  $\Psi_{N-2}$  und so weiter bis zum Startzustand zurück. Die Zeilen stehen dabei für die durchlaufenen Zustände.
- Berechnung aus dem Buch:

$$q_t^* = \Psi_{t+1}(q_{t+1}^*), \quad t = T-1, T-2, \dots, 1$$

## 9 MATRIZENRECHNUNG

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$