

Universität Potsdam  
Institut für Informatik  
Lehrstuhl Maschinelles Lernen



---

# Hidden-Markov-Modelle

Tobias Scheffer

# Hidden-Markov-Modelle: Wozu?

- Spracherkennung:
  - ◆ Akustisches Modell.
- Geschriebene Sprache:
  - ◆ Part-of-Speech-Tagging,
  - ◆ Informationsextraktion.
- Biologie:
  - ◆ Finden von Genen in DNA,
  - ◆ Vorhersage der Protein-Struktur.

# Markov-Prozesse

- $X_1, \dots, X_n$ : Zufallsvariablen.
- Allgemein gilt:  $P(X_1, \dots, X_n) = P(X_1) \prod_{i=2}^n P(X_i | X_{i-1}, \dots, X_1)$
- Zufallsvariablen bilden eine Markovkette, gdw:  
$$P(X_1, \dots, X_n) = P(X_1) \prod_{i=2}^n P(X_i | X_{i-1})$$
- Jede Variable  $X_i$  nur von Vorgänger  $X_{i-1}$  abhängig.

- Markov-Modell:  
Probabilistischer endlicher  
Automat, Folge der Zustände  
ist Markov-Kette.

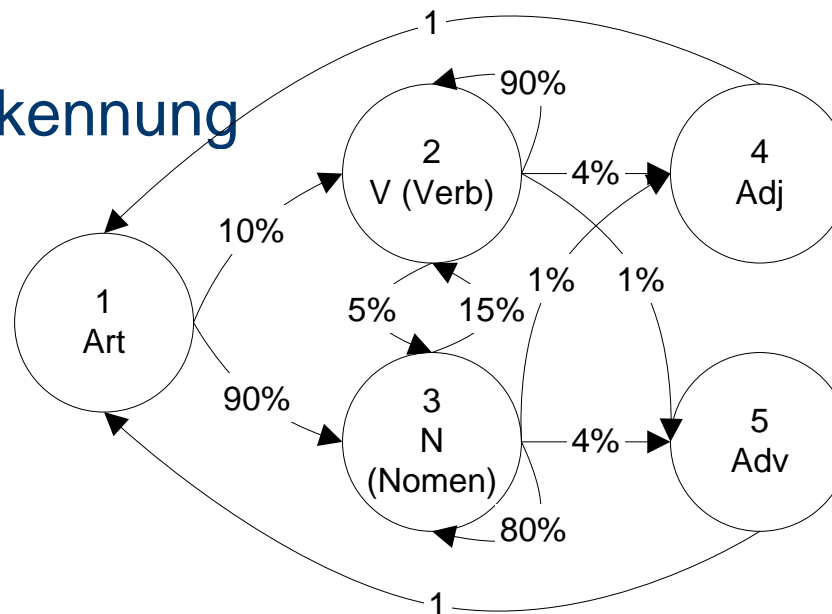
- (Andrei Markov, 1856-1922)



# Markov-Modell

- Zustände 1, ..., N (Folge der Zustände ist Markov-Kette),
- Transitionswahrscheinlichkeiten  $a_{ij}$ ;  $\sum_{j=1}^N a_{ij} = 1$
- Startwahrscheinlichkeiten  $\pi_i$ .
- Zustand zur Zeit t:  $q_t$ .

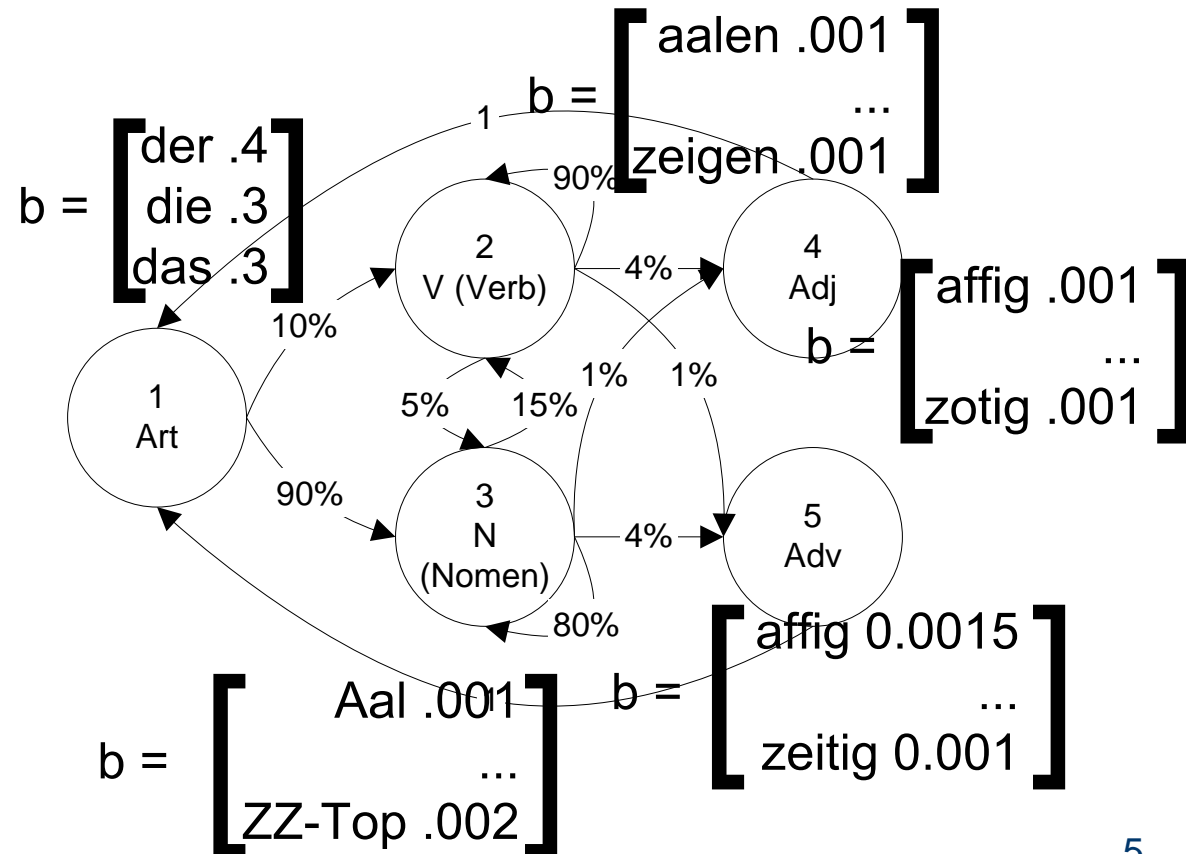
- Z.B. Wortartenerkennung („Part-of-Speech Tagging“)



- P(Präp, Nomen, Nomen, Verb)?

# Hidden-Markov-Modell

- Folge der Zustände ist nicht sichtbar.
- Statt dessen: Zustände emittieren Beobachtungen  $O_t$  (mit Wahrscheinlichkeit  $b_i(O_t)$ ).



# Hidden-Markov-Modell, Definitionen

- Zustände  $\Omega = \{1, \dots, N\}$   $q_t$ : Zustand zur Zeit  $t$ .
- Übergangswahrscheinlichkeiten  $A = \{a_{ij}\}$
- Startwahrscheinlichkeiten  $\pi = \{\pi_i = P(q_1 = i)\}$
- Beobachtungswahrscheinlichkeiten  
 $B = \{b_i(O_t) = P(O_t | q_t = i)\}$
- HMM definiert durch Parameter  $\lambda = (A, B, \pi)$

# Markov-Annahmen

- Markov-Annahme für Zustandsfolgen:

$$P(q_t | q_{t-1}, \dots, q_1) = P(q_t | q_{t-1})$$

- Markov-Annahme für Beobachtungen:

$$P(O_t | q_t, q_{t-1}, \dots, q_1) = P(O_t | q_t)$$

# Drei Basisprobleme

- Likelihood einer Beobachtungsfolge:
  - ◆ „Wie gut passt ein Modell zu einer Beobachtungsfolge?“
  - ◆ Berechne  $P(O_1, \dots, O_T | \lambda)$
- Wahrscheinlichkeit eines Zustandes zur Zeit t:
  - ◆ „In welchem Zustand ist eine Beobachtung  $O_t$  erzeugt worden?“
  - ◆ Berechne  $P(q_t = i | O_1, \dots, O_T, \lambda)$
- Lernproblem
  - ◆ „Gegeben viele Beobachtungsfolgen, finde die Parameter des HMMs!“
  - ◆ Berechne  $\arg \max_{\lambda} P(\{(O_1, \dots, O_T), \dots\} | \lambda)$



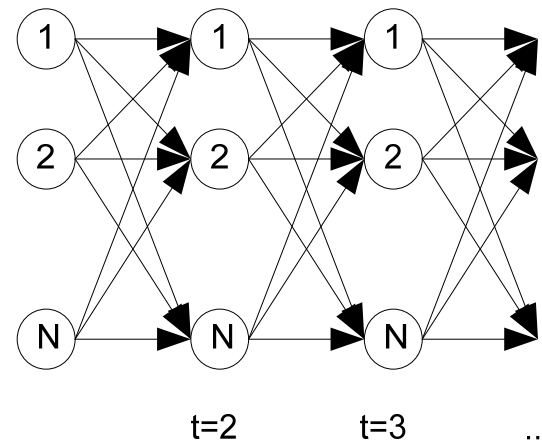
# „Wie gut passt ein Modell zur Beobachtungsfolge?“

$$\begin{aligned} \blacksquare P(O_1, \dots, O_T | \lambda) &= \sum_{\text{alle}(q_1, \dots, q_T)} P(O_1, \dots, O_T | q_1, \dots, q_T, \lambda) P(q_1, \dots, q_T | \lambda) \\ &= \sum_{\text{alle}(q_1, \dots, q_T)} b_{q_1}(O_1) \dots b_{q_T}(O_T) \pi_{q_1} a_{q_1 q_2} \dots a_{q_{T-1} q_T} \end{aligned}$$

- # Summanden =  $N^T$ .
- Auswertung exponentiell in der Länge der Eingabe.
- Bei Auswertung werden dieselben Wahrscheinlichkeiten wiederholt berechnet
- Gesucht: polynomieller (linearer) Algorithmus.
- Dynamische Programmierung: Zwischenergebnisse speichern.

# Trellis

- Trellis: Array über Zustände x Zeit.
- Hilft der Vorstellung, muss man aber nicht implementieren.



- Rekursive Hilfsvariablen
  - ◆  $\alpha_t(i) = P(O_1, \dots, O_t, q_t = i | \lambda)$
  - ◆  $\beta_t(i) = P(O_{t+1}, \dots, O_T | q_t = i, \lambda)$
  - ◆  $\gamma_t(i) = P(q_t = i | O_1, \dots, O_T, \lambda)$

# „Forward“

- Wahrscheinlichkeit einer initialen Beobachtungsfolge und eines Zustands:

$$\alpha_t(i) = P(O_1, \dots, O_t, q_t = i \mid \lambda)$$

- Theorem:

$$\alpha_1(i) = \pi_i b_i(O_1)$$

$$\alpha_{t+1}(j) = \left( \sum_{i=1}^N \alpha_t(i) a_{ij} \right) b_j(O_{t+1})$$

- Nach Theorem kann  $\alpha$  durch dynamische Programmierung berechnet werden:
  - ◆ Initialisiere  $\alpha_1(i)$ .
  - ◆ Für  $t$  von 2 bis  $T$ : berechne  $\alpha_t(i)$  unter Verwendung der schon bestimmten  $\alpha_{t-1}(i)$ .

# „Forward“: Beweis

- Induktionsverankerung:

$$\begin{aligned}\alpha_1(i) &= P(O_1, q_1 = i | \lambda) \\ &= P(q_1 = i | \lambda)P(O_1 | q_1 = i, \lambda) = \pi_i b_i(O_1)\end{aligned}$$

- Induktionsschritt  $t \rightarrow t+1$

$$\begin{aligned}\alpha_{t+1}(j) &= P(O_1, \dots, O_{t+1}, q_{t+1} = j | \lambda) = \sum_{i=1}^N P(O_1, \dots, O_{t+1}, q_t = i, q_{t+1} = j | \lambda) \\ &= \sum_{i=1}^N P(O_1, \dots, O_t, q_t = i | \lambda) P(q_{t+1} = j | q_t = i, O_1, \dots, O_t, \lambda) \\ &\quad P(O_{t+1} | q_{t+1} = j, q_t = i, O_1, \dots, O_t, \lambda) \\ &= \sum_{i=1}^N P(O_1, \dots, O_t, q_t = i | \lambda) P(q_{t+1} = j | q_t = i, \lambda) P(O_{t+1} | q_{t+1} = j, \lambda) \\ &= \left( \sum_{i=1}^N \alpha_t(i) a_{ij} \right) b_j(O_{t+1})\end{aligned}$$

# „Forward“: Termination

- $P(O_1, \dots, O_T | \lambda) = \sum_i \alpha_T(i)$

- Beweis:

$$P(O_1, \dots, O_T | \lambda) = \sum_i P(O_1, \dots, O_T, q_T = i | \lambda) = \sum_i \alpha_T(i)$$

# „Backward“

- $\beta_t(i) = P(O_{t+1}, \dots, O_T \mid q_t = i, \lambda)$

- Theorem:

$$\beta_T(i) = 1$$

$$\beta_t(i) = \left( \sum_{j=1}^N a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \right)$$

- Nach dem Theorem kann  $\beta$  durch dynamische Programmierung bestimmt werden:
  - ◆ Initialisiere  $\beta_T(i)=1$ .
  - ◆ Für  $t$  von  $T-1$  bis  $1$ : bestimme  $\beta_t(i)$  unter Verwendung der  $\beta_{t+1}(j)$ .

# „Backward“: Beweis

- Induktionsverankerung:

$$\begin{aligned}\beta_{T-1}(i) &= P(O_T | q_{T-1} = i, \lambda) \\ &= \sum_j P(O_T | q_T = j, q_{T-1} = i, \lambda) P(q_T = j | q_{T-1} = i, \lambda) \\ &= \sum_j P(O_T | q_T = j, \lambda) a_{ij} \\ &= \sum_j a_{ij} \beta_T(j)\end{aligned}$$

# „Backward“: Beweis

- Induktionsschritt  $t+1 \rightarrow t$



# „Forward Backward“: Wahrscheinlichkeit eines Zustandes

- $\gamma_t(i) = P(q_t = S_i | O_1, \dots, O_T, \lambda)$
- P(Zustand i zur Zeit t | Beobachtungssequenz)
- $$\begin{aligned}\gamma_t(i) &= P(q_t = S_i | O_1, \dots, O_T, \lambda) \\ &= \frac{P(q_t = S_i, O_1, \dots, O_T | \lambda)}{P(O_1, \dots, O_T | \lambda)} \\ &= \frac{P(q_t = S_i, O_1, \dots, O_t, O_{t+1}, \dots, O_T | \lambda)}{P(O_1, \dots, O_T | \lambda)} \\ &= \frac{P(q_t = S_i, O_1, \dots, O_t | \lambda) P(O_{t+1}, \dots, O_T | q_t = i, \lambda)}{P(O_1, \dots, O_T | \lambda)} \\ &= \frac{\alpha_t(i) \beta_t(i)}{P(O_1, \dots, O_T | \lambda)}\end{aligned}$$

# Forward-Backward-Algorithmus

- (Forward)
- Initialisiere  $\alpha_1(i) = \pi_i b_i(O_1)$  (alle Zustände  $i$ )
- Für  $t$  von 1 bis  $T-1$ 
  - ◆ Berechne (für alle  $j$ )  $\alpha_{t+1}(j) = \left( \sum_{i=1}^N \alpha_t(i) a_{ij} \right) b_j(O_{t+1})$
- Berechne  $P(O_1, \dots, O_T | \lambda) = \sum_i \alpha_T(i)$
- (Backward)
- Initialisiere  $\beta_T(i) = 1$
- Für  $t$  von  $T-1$  bis 1
  - ◆ Berechne (für alle  $i$ )  $\beta_t(i) = \left( \sum_{j=1}^N a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \right)$
  - ◆ Berechne (für alle  $i$ )  $\gamma_t(i) = \frac{\alpha_t(i) \beta_t(i)}{P(O_1, \dots, O_T | \lambda)}$

# Forward-Backward-Algorithmus

- Läuft mit linearem Aufwand (iteriert zweimal über die Beobachtungssequenz).
- Berechnet  $\gamma_t(i) = P(q_t = i \mid O_1, \dots, O_T, \lambda)$   
und  $P(O_1, \dots, O_T \mid \lambda)$

# „Welches Modell passt am besten?“

- Bsp: Worterkennung. Ein HMM für jedes Wort, das erkannt werden soll.
- Gegeben: Sprachsignal (Beobachtungssequenz), gesucht: Welches der Wörter wurde gesagt?
- $\arg \max_k P(\lambda_k | O_1, \dots, O_T)$   
=  $\arg \max P(O_1, \dots, O_T | \lambda_k) P(\lambda_k)$
- Likelihood durch Forward-Algorithmus, A-Priori-Wahrscheinlichkeit durch Abzählen der Worthäufigkeit in der Trainingsmenge.

# „Welcher Zustand hat Beobachtung erzeugt?“

- Beispiele:
  - ◆ Part-of-Speech-Tagging, ein Zustand pro Part-of-Speech,
  - ◆ Gensequenzanalyse, Zustände entsprechen Tags, mit denen das Genom annotiert werden soll.
- Problem 1: Welcher einzelne Zustand zur Zeit  $z$  passt am besten zur Beobachtungsfolge?
  - ◆  $\arg \max_i \gamma_t(i) = \arg \max_i P(q_t = i | O_1, \dots, O_T, \lambda)$
  - ◆ Bestimmung durch Forward-Backward-Algorithmus
- Problem 2: Welche komplette Zustandsfolge passt am besten zur Beobachtungsfolge?
  - ◆  $\arg \max_{(q_1, \dots, q_T)} P(q_1, \dots, q_T | O_1, \dots, O_T, \lambda)$
  - ◆ Bestimmen mit Viterbi-Algorithmus

# Viterbi-Algorithmus, Theorem

- $\delta_t(i) = \max_{q_1, \dots, q_{t-1}} P(q_1, \dots, q_{t-1}, q_t = i, O_1, \dots, O_t | \lambda)$

- **Theorem:**

$$\delta_{t+1}(j) = \left( \max_i \delta_t(i) a_{ij} \right) b_j(O_{t+1})$$

- **Beweis:**

$$\delta_{t+1}(j) = \max_{q_1, \dots, q_t} P(q_1, \dots, q_t, q_{t+1} = j, O_1, \dots, O_{t+1} | \lambda)$$

$$= \max_{q_1, \dots, q_t} P(q_1, \dots, q_t, O_1, \dots, O_t | \lambda) P(q_{t+1} = j | q_t, \dots) P(O_{t+1} | q_{t+1} = j, \dots)$$

$$= \left( \max_i \left( \max_{q_1, \dots, q_{t-1}} P(q_1, \dots, q_t = i, O_1, \dots, O_t | \lambda) \right) a_{ij} \right) b_j(O_{t+1})$$

$$= \left( \max_i \delta_t(i) a_{ij} \right) b_j(O_{t+1})$$

- Zustand zur Zeit t auf wahrscheinlichstem Pfad:  $\psi_t(j)$

# Viterbi-Algorithmus

- Initialisierung:  $\delta_1(i) = \pi_i b_i(O_1)$
- Initialisierung:  $\psi_1(i) = 0$
- Für  $t$  von 1 bis  $T-1$  und  $j$  von 1 bis  $N$ :
  - ◆  $\delta_{t+1}(j) = \left( \max_i \delta_t(i) a_{ij} \right) b_j(O_{t+1})$
  - ◆  $\psi_{t+1}(j) = \left( \arg \max_i \delta_t(i) a_{ij} \right)$
- Termination  $q_T^* = \arg \max_i \delta_T(i)$
- Für  $t$  von  $T-1$  bis 1
  - ◆  $q_t^* = \psi_{t+1}(q_{t+1}^*)$
- Ausgabe der Zustandsfolge  $q_1^*, \dots, q_T^*$

# Lernproblem

- Gegeben: Sammlung von Beobachtungsfolgen.
- Gesucht HMM-Parameter  $\lambda$ .
- Sichtbare Zustände
  - ◆ Z.B. Part-of-Speech-Tagging: Jede Beobachtung ist mit dem zugehörigen Zustand markiert.
  - ◆ Schätzen der Parameter durch Zählen der Starthäufigkeiten, Transitionen, Beobachtungen.
- Unsichtbare Zustände
  - ◆ Z.B. Worterkennung: Nur Sprachsignal gegeben, Zustandsfolgen sind unbekannt.
  - ◆ Lernen der Parameter durch Baum-Welch-Algorithmus.



# Sichtbare Zustände

- Trainingsmenge

$$S = \langle (O^{(1)}, Q^{(1)}), \dots, (O^{(m)}, Q^{(m)}) \rangle$$

$$O^{(k)} = O_1^{(k)}, \dots, O_{T_k}^{(k)}$$

$$Q^{(k)} = q_1^{(k)}, \dots, q_{T_k}^{(k)}$$

- ML-Schätzer:

- ◆  $\pi_i = \frac{\text{\#Sequenzen } k \text{ mit } q_1^{(k)} = i}{\text{\#Sequenzen}}$

- ◆  $a_{ij} = \frac{\text{\#Stellen mit } q_t^{(k)} = i, q_{t+1}^{(k)} = j}{\text{\#Stellen mit } q_t^{(k)} = i}$

- ◆  $b_i(o) = \frac{\text{\#Stellen mit } q_t^{(k)} = i, O_t = o}{\text{\#Stellen mit } q_t^{(k)} = i}$

# Sichtbare Zustände

- Trainingsmenge

$$S = \langle (O^{(1)}, Q^{(1)}), \dots, (O^{(m)}, Q^{(m)}) \rangle$$

$$O^{(k)} = O_1^{(k)}, \dots, O_{T_k}^{(k)}$$

$$Q^{(k)} = q_1^{(k)}, \dots, q_{T_k}^{(k)}$$

- MAP-Schätzer:

- ◆  $\pi_i = \frac{(\# \text{Sequenzen } k \text{ mit } q_1^{(k)} = i) + \alpha_i}{\# \text{Sequenzen} + \sum_i \alpha_i}$

- ◆  $a_{ij} = \frac{(\# \text{Stellen mit } q_t^{(k)} = i, q_{t+1}^{(k)} = j) + \alpha_j}{(\# \text{Stellen mit } q_t^{(k)} = i) + \sum_j \alpha_j}$

- ◆  $b_i(o) = \frac{(\# \text{Stellen mit } q_t^{(k)} = i, O_t = o) + \alpha_o}{(\# \text{Stellen mit } q_t^{(k)} = i) + \sum_j \alpha_j}$

# Implementierung von HMMs

- Wahrscheinlichkeiten werden sehr klein.
- Daher: rechnen mit logarithmierten Wahrscheinlichkeiten.
- (Aus Multiplikationen werden Additionen).

# Skalierung

- Forward-Backward und Viterbi multiplizieren viele Wahrscheinlichkeiten auf, numerisch kommt dabei schnell 0 heraus.
- Mit negativen Log-Wahrscheinlichkeiten arbeiten, statt mit Wahrscheinlichkeiten.
- Forward-Backward- und Viterbi lassen sich entsprechend umformulieren.

# Hidden-Markov-Modelle: Erweiterungen

- Hierarchische HMM: Knoten können Unterknoten haben, für Part-of-Speech-Tagging interessant (z.B. kommt ein Verb in einer Verbphrase vor).
- Conditional Random Fields und Hidden-Markov-Support-Vektor-Maschinen:
  - ◆ Diskriminative (statt generativer) Sequenzmodelle.
  - ◆ Modellieren  $P(q \mid O)$  statt  $P(q, O)$ .
- Mehrdimensionale HMM: Markov Random Fields (Bildverarbeitung).
- Probabilistische, kontextfreie Grammatiken: CFG statt endlichen Automaten.

# Diskriminative Modelle

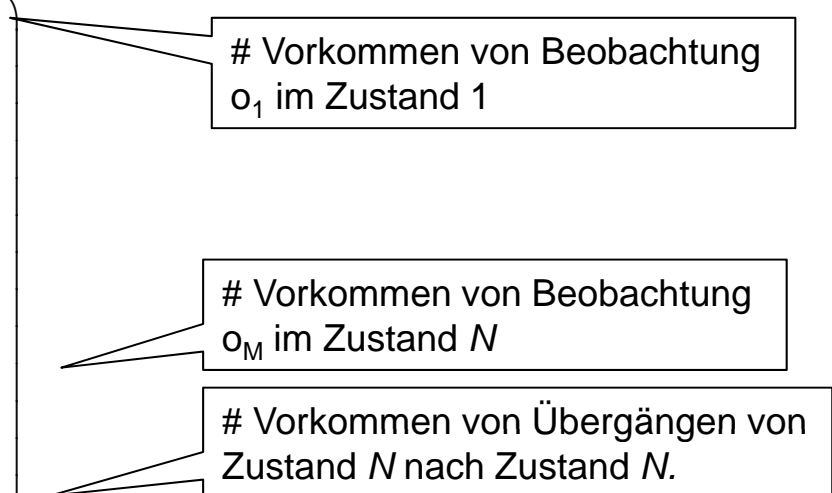
- Optimierungskriterium beim Lernen von HMMs:
  - ◆ Bei ML-Schätzer:  $P((O^{(1)}, Q^{(1)}), \dots, (O^{(m)}, Q^{(m)}) | \lambda)$
  - ◆ Bei MAP-Schätzer:  $P(\lambda | (O^{(1)}, Q^{(1)}), \dots, (O^{(m)}, Q^{(m)}))$   
 $= P((O^{(1)}, Q^{(1)}), \dots, (O^{(m)}, Q^{(m)}) | \lambda) p(\lambda)$
- Bei vielen Anwendungen eigentlich gewünscht: korrekte Zustände gegeben Beobachtungen.
  - ◆  $P(Q^{(1)}, \dots, Q^{(m)} | O^{(1)}, \dots, O^{(m)}, \lambda)$
  - ◆ Lässt sich mit diskriminativen Modellen optimieren.

# Diskriminative Modelle: HM-SVM

- Vorhersage:

- ◆  $q_1^* \dots q_T^* = \arg \max_{q_1 \dots q_T} f(Q_1 \dots Q_T, O_1, \dots, O_t)$   
 $= \arg \max_{q_1 \dots q_T} \mathbf{w}^T \Phi(Q_1 \dots Q_T, O_1, \dots, O_t)$

- $\Phi$  wird so konstruiert, dass es dieselben Informationen enthält, die auch HMM verarbeitet:

$$\Phi(O, Q) = \begin{pmatrix} \#[q_t = 1][O_t = o_1] \\ \dots \\ \#[q_t = 1][O_t = o_M] \\ \dots \\ \#[q_t = N][O_t = o_1] \\ \dots \\ \#[q_t = N][O_t = o_M] \\ \#[q_t = 1][q_{t+1} = 1] \\ \dots \\ \#[q_t = N][q_{t+1} = N] \end{pmatrix}$$


- # Vorkommen von Beobachtung  $o_1$  im Zustand 1
- # Vorkommen von Beobachtung  $o_M$  im Zustand  $N$
- # Vorkommen von Übergängen von Zustand  $N$  nach Zustand  $N$ .

# Diskriminative Modelle: HM-SVM

- Optimierungskriterium: richtige Zustandsfolge für jede Beobachtungsfolge bei maximalem Margin:
  - ◆  $\min |\mathbf{w}|^2 + \sum_k \xi_k$  so dass für alle Beispiele  $k$ , alle  $\bar{Q} \neq Q^{(k)}$ 

$$\mathbf{w}^T (\Phi(O^{(k)}, Q^{(k)}) - \Phi(O^{(k)}, \bar{Q})) \geq 1 - \xi_k$$
- $\Phi$  wird so konstruiert, dass es dieselben Informationen enthält, die auch HMM verarbeitet:

$$\diamond \quad \Phi(O, Q) = \begin{pmatrix} \#[q_t = 1][O_t = o_1] \\ \dots \\ \#[q_t = 1][O_t = o_M] \\ \dots \\ \#[q_t = N][O_t = o_1] \\ \dots \\ \#[q_t = N][O_t = o_M] \end{pmatrix}$$

# Vorkommen von Beobachtung  $o_1$  im Zustand 1

# Vorkommen von Beobachtung  $o_M$  im Zustand  $N$



# Diskriminative Modelle: HM-SVM

- Optimierungskriterium: richtige Zustandsfolge für jede Beobachtungsfolge bei maximalem Margin:
  - ◆  $\min |\mathbf{w}|^2 + \sum_k \xi_k$  so dass für alle Beispiele  $k$ , alle  $\bar{Q} \neq Q^{(k)}$   
 $\mathbf{w}^T (\Phi(O^{(k)}, Q^{(k)}) - \Phi(O^{(k)}, \bar{Q})) \geq 1 - \xi_k$
- Optimierung:
  - ◆ Iteriere über alle Beispiele,
  - ◆ Füge negatives Pseudo-Beispiel mit größter Margin-Verletzung der Arbeitsmenge hinzu,
  - ◆ Optimiere über Arbeitsmenge
  - ◆ Wiederhole letzte beiden Schritte bis kein neues negatives Beispiel mehr Margin verletzt.

# HMM vs. HM-SVM

- Entscheidungsfunktionen HMM vs HM-SVM:
  - ◆  $f(Q, O)$  vs.  $\log P(Q, O)$
  - ◆ Gleiche Form der linearen Entscheidungsfunktion.
- Optimierungskriterien:
  - ◆  $P(\lambda | Q^{(1..m)}, O^{(1..m)}) = P(Q^{(1..m)}, O^{(1..m)} | \lambda) p(\lambda)$
  - ◆ Vs. max Margin mit  $\mathbf{w}^T (\Phi(O^{(k)}, Q^{(k)}) - \Phi(O^{(k)}, \bar{Q})) \geq 1 - \xi_k$
  - ◆ Generativ vs. diskriminativ.

# Diskriminative Modelle: CRF

- Modell von  $P(Q|O)$  statt  $P(Q,O)$ 
  - ◆  $P(Q|O) \propto \exp[\mathbf{w}^T \Phi(Q,O)]$
- Optimierungskriterium:
  - ◆  $\arg \max_{\mathbf{w}} P(Q^{(1)}, \dots, Q^{(m)} | O^{(1)}, \dots, O^{(m)}, \mathbf{w}) p(\mathbf{w})$   
=  $\arg \max_{\mathbf{w}} \log P(Q^{(1)}, \dots, Q^{(m)} | O^{(1)}, \dots, O^{(m)}, \mathbf{w}) - \Omega(\mathbf{w})$   
=  $\arg \max_{\mathbf{w}} \sum_i \mathbf{w}^T \Phi(Q^{(i)}, O^{(i)}) - \Omega(\mathbf{w})$
- HM-SVM vs. CRF:
  - ◆ HM-SVM: Maximiere Margin, so dass kein Beispiel Margin verletzt.
  - ◆ CRF: Entscheidungsfunktionswert für richtige Zustandssequenz möglichst hoch.