

①

Maschinelles Lernen Hidden Markov Modelle (Topic 9)

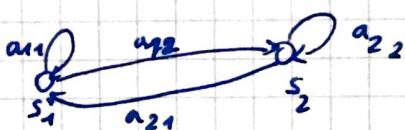
Motivation

- Signale in der realen Welt häufig verrauscht und besitzen nicht-deterministische Eigenschaften
- Signale werden mit stochastischen Modellen, z.B. Markov Prozess modelliert

Der Vorteil durch stochastische Signalmodelle:

- Erkennung der Signale
- theoretische Beschreibung von signalverarbeitenden Systemen
- Simulationen
- in Praxis meist sehr gut

Diskreter Markov Prozess



N diskrete Zustände

$$S = \{s_1, s_2, \dots, s_N\}$$

Zeitpunkte der Zustandsübergänge

$$t = 1, 2, \dots$$

Aktueller Zustand zur Zeit $t = q_t$

Übergangswahrscheinlichkeiten mit einer Tabelle modelliert:

$$A = \{a_{ij}\} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix} \quad \begin{array}{l} s_1 \text{ (Regen)} \\ s_2 \text{ (Gewölk)} \\ s_3 \text{ (sonnig)} \end{array}$$

Beispiel:

Wenn heute sonnig, wie wahrscheinlich ist es, dass das Wetter in den nächsten 7 Tagen Sonnig, sonnig, Regen, Regen, sonnig, bewölkt, sonnig ist?

Beobachtung: $O = s_3, s_3, s_3, s_1, s_1, s_3, s_2, s_3$

Lösung: $P(O|\text{Modell}) = P(s_3, s_3, s_3, s_1, s_1, s_3, s_2, s_3 | \text{Modell})$

$$\begin{aligned} &= P(s_3) \cdot P(s_3 | s_3) \cdot P(s_3 | s_3) \cdot P(s_1 | s_3) \\ &\quad \cdot P(s_1 | s_1) \cdot P(s_3 | s_1) \cdot P(s_2 | s_3) \cdot P(s_3 | s_2) \\ &= \pi_{s_3} \cdot q_{33} \cdot q_{33} \cdot q_{31} \cdot q_{11} \cdot q_{13} \cdot q_{32} \cdot q_{22} \\ &= 1 \cdot 0.8 \cdot 0.8 \cdot 0.1 \dots = 1.536 \cdot 10^{-4} \end{aligned}$$

Markov-Bedingung

- die Wahrsch. einen Zustand zu erreichen, ist nur von seinem direkten Vorgängerzustand abhängig:

$$\begin{aligned} &P(q_{t+1} = s_j | q_t = s_i, q_{t-1} = s_k, \dots) \quad (1. \text{ Ordnung}) \\ &= P(q_{t+1} = s_j | q_t = s_i) \end{aligned}$$

- die Wahrsch. eines Zustandsübergangs ist unabhängig von der Zeit (Zeitinvarianz)

Hidden Markov Modelle (HMM)

Bisher: Ereignisse (Zustände) sind direkt beobachtbar gewesen

HMM: • Beobachtung ist eine stochastische Funktion der Zustands
→ Zustände können nur indirekt beobachtet werden

- Doppelt stochastischer Prozess:

- a) der nicht beobachtbare Prozess ist stochastisch
- b) er wird beobachtet durch einen stochastischen Prozess, der die Sequenz beobachteter Symbole erzeugt

Zwei Ebenen

(nicht beobachtbarer Prozess)
(der, der ihn produziert)

Definition von HMM

Fünf-Typel $\lambda = \{S, V, A, B, \Pi\}$

S : Menge der Zustände $S = \{S_1, S_2, \dots, S_N\}$
V : Menge der Ausgabezeichen $V = \{v_1, \dots, v_M\}$

q_t : Zustand zur Zeit t

A : Matrix der Übergangswahrscheinlichkeiten $A = (a_{ij})$
wobei a_{ij} die Wahrsch. ist, dass S_i nach S_j kommt

B : Menge der Emissionswahrscheinlichkeit, $b_i(v_k)$ ist die Wahrsch., v_k im Zustand S_i zu beobachten

Π : Die Verteilung der Anfangswahrscheinlichkeiten, π_i ist die Wahrsch., dass S_i der erste Zustand ist

Es gibt 3 grundlegende Probleme

Problem 1 (Evaluationsproblem)

Gegeben: Modell $\lambda = \{S, V, A, B, \Pi\}$

Gesucht: Wahrscheinlichkeit $P(O | \lambda)$

d.h. für die Ausgabe $O = O_1 O_2 \dots O_T$

Wie gut erklärt ein Modell eine Beobachtungssequenz?

Problem 2 (Dekodierungsproblem)

Gegeben: Ausgabesequenz O und Modell λ

Gesucht: wahrscheinlichste Zustandsfolge Q , die O erklärt

Finden der „korrekten“ Zustandssequenz (verborgen)
→ Optimalitätskriterium?

Problem 3 (Lern- oder Optimierungsproblem)

Gegeben: Ausgabesequenz O und Suchraum für Modelle

Gesucht: Anpassung der Parameter $\lambda = \{S, V, A, B, \Pi\}$,
sodass O besser erklärt werden kann
→ sodass $P(O | \lambda)$ maximal

Optimierung der Parameter (Training)

Beispiele: „Worterkenner“ Schritte:

1) Aufbau von HMMs für jedes zu erkennende Wort P3

2) Verstehen der aufgebauten Modelle P2 → sinnvolle Verbesserungen

3) Erkennen unbekannter Wörter durch Finden des besten Modells P1 (PS)

Der naive Ansatz für Problem 1 (Ausgabesequenz schätzen)

- Angenommen, Zustandsfolge Q ist bekannt (und Beobachtungen sind unabhängig), dann Ausgabe:

$$P(O | Q, \lambda) = \prod_{t=1}^T P(O_t | q_t, \lambda) = b_{q_1}(O_1) b_{q_2}(O_2) \dots b_{q_T}(O_T)$$

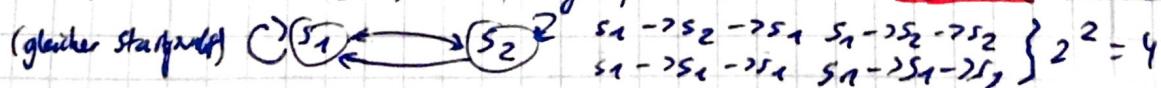
- Aber(!), für jede Zustandsfolge $q_1 \rightarrow q_2 \rightarrow \dots \rightarrow q_T$, lässt sich die Wahrscheinlichkeit berechnen,
dass sie auftritt

$$P(Q | \lambda) = \pi_{q_1} a_{q_1 q_2} a_{q_2 q_3} \dots a_{q_{T-1} q_T}$$

- Dann lässt sich doch die Wahrsch. berechnen, dass die Ausgabesequenz O erzeugt wird, indem ich die Wahrsch. zu allen möglichen Zustandsfolgen berechne und diese mit der w. multipliziere
dass sie O erzeugen (und dann aufaddiere)? Ja...

$$P(O | \lambda) = \sum_{\text{alle } Q} P(O | Q, \lambda) P(Q | \lambda) = \pi_{q_1} b_{q_1}(O_1) a_{q_1 q_2} b_{q_2}(O_2) a_{q_2 q_3} \dots$$

Problem: Es wird über alle möglichen Pfade aufsummiert. Es gibt N^T Pfade ($N = |S|, T = \text{Zeitrücksicht}$), jeder Pfad kostet $O(T)$ Berechnungen, was bedeutet: $O(TN^T)$ Laufzeit.



Die effizientere Lösung für Problem 1:

Ziel:

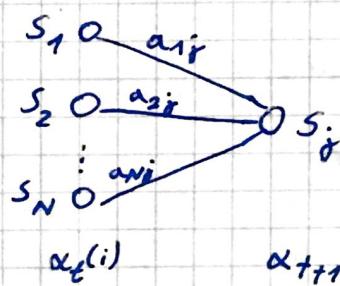
- exponentiellen Aufwand vermeiden durch Rekursion/Induktion
(bzw. Dynamic Programming)
- Berechne Teilresultate und speichere diese
- keine erneuten Berechnungen von bereits durchgeführten Berechnungen

Definiere: Vorwärts-Algorithmus

$$(1) \quad \alpha_t(i) = P(O_1 O_2 \dots O_t | q_t = s_i; \lambda)$$

1) Initialisiere: $\alpha_1(i) = \pi_i b_i(O_1)$ - Anfangswahrsch.
wahrsch., dass O_1 in Zustand (i) beobachtet wird

2) Induktion/Rekursion:



(Inspirierte durch Markov-Eigenschaft:
Übergang zu S_j ist nur abhängig von aktuellem Zustand)

(1) bedeutet die Wahrsch. zum Zeitpunkt t das Präfix $O_1 O_2 \dots O_t$ beobachtet zu haben und im Zustand $s_i \in S$ zu sein

Für $\alpha_{t+1}(j)$ müssen nur die Pfade aller Vorgängerzustände s_i zum Zeitpunkt t betrachtet werden und daher über alle aufsummiert werden und mit der Emissionswahrscheinlichkeit für O_{t+1} multipliziert werden

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

3) Terminierung:

Angenommen man ist nur an der Wahrscheinlichkeit der Ausgabesequenz interessiert (State eher uninteressant) =>

$$P(O | \lambda) = \sum_{i=1}^N \alpha_T(i)$$

Wahrsch. der Ausgabesequenz ist die Summe aller Beobachtungen bis T (und dem Landen in State i)

Laufzeitverbesserung: $O(N^2 T)$

Definiere: Rückwärts-Algorithmus
(analog zu Vorwärts-Algorithmus)

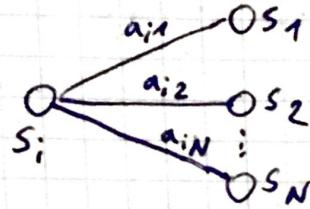
$$(1) \quad \beta_t(i) = P(O_{t+1} O_{t+2} \dots O_T | q_t = s_i; \lambda) \quad t = T-1, T-2 \text{ etc.}$$

Werte werden für Lernalgorithmus benötigt

1) Initialisiere: $\beta_T(i) = 1$

2) Induktion / Rekursion

$$+ \leftarrow t_{t+1}$$



$\beta_t(i)$

$\beta_{t+1}(j)$

Für $\beta_t(i)$ müssen die Pfade über alle Nachfolgerzustände s_j zum Zeitpunkt $t+1$ betrachtet werden

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

(1) $\beta_t(i)$ bezeichnet die Wahrscheinlichkeit des Suffixes $O_{t+1} O_{t+2} \dots O_T$ zu beobachten, falls das HMM im Zustand $s_i \in S$ gewesen ist

3) Terminierung:

$$P(O|\lambda) = \sum_{j=1}^N \pi_j b_j(O_1) \cdot \beta_1(j)$$

Ansatz für Problem 2 (Wahrscheinlichste Zustandsfolge, die O erklärt)

Definition der „optimalen“ Zustandsfolge?

Ein Ansatz: wähle die Zustände q_t , die unabhängig voneinander am wahrscheinlichsten sind

Definiere:

$$\gamma_t(i) = \frac{\alpha_t(i) \beta_t(i)}{P(O|\lambda)} = \frac{\alpha_t(i) \beta_t(i)}{\sum_{i=1}^N \alpha_t(i) \beta_t(i)} \quad \begin{array}{l} \text{die W. zum Zeitpunkt } t \text{ des Präfix} \\ O_1 \dots O_t \text{ multipliziert mit W. zum Zp. } t \\ \text{das Suffix } O_{t+1} \dots O_T \text{ zu sehen} \end{array}$$

Lösung: (für alle t mit $1 \leq t \leq T$) $\boxed{q_t = \arg \max_{1 \leq i \leq N} \gamma_t(i)},$

finde Lösung, indem über $t=1 \dots T$ iteriert wird

für z.B. „abc“ und $t=2$: „ab“ iteriere über alle Zustände

> Aber ein Problem: Bei nicht vollständig vernetztem

HMM ergibt dies evtl. keinen

gültigen Pfad (z.B. bestes

$q_t = s_i$ und $q_{t+1} = s_j$ aber $a_{ij} = 0$)

Besser: Wahl der insgesamt besten Zustandsfolge über Maximierung von:

$$P(Q|O, \lambda) \\ \Rightarrow \text{entspricht der Maximierung von } P(Q, O | \lambda)$$

$$\arg \max_Q P(Q|O, \lambda) = \arg \max_Q \frac{P(Q, O | \lambda)}{P(O | \lambda)} = \arg \max_Q P(Q | O | \lambda) \quad (\text{siehe Sheet})$$

=> Viterbi - Algorithmus

⇒ wie Vorwärts - Algorithmus, nur speichert Vorrätsvariable maximale Wahrscheinlichkeit bis der ein Präfix $O_1 O_2 \dots O_t$ beobachtet wurde und in Zustand s_i gelandet wird

$$\delta(i) = \max_{q_1, q_2, \dots, q_{t-1}} P(q_1, q_2, \dots, q_t = s_i, O_1, O_2 \dots O_t | \lambda)$$

mit Induktions Schritt:

Unterschied: Maximierung statt Summierung

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

die Rückwärtsvariable speichert wahrscheinlichsten

Vorgängerzustand: $\psi_t(j)$ (für jeden Zeitpunkt und Zustand)

(nicht wie bei Rückwärts - Alg. !)

Partielle Wahrsch. repräsentieren hier die Wahrsch. des von wahrsch. Pfad zum Zeitpunkt t anstatt die Wahrsch. aller vorherigen möglichen Pfade

(S)

Maschinelles Lernen 1 Hidden Markov Modelle (Topic 9)

Viterbi - Algorithmus Fortsetzung

1) Initialisierung: $\delta_1(i) = \pi_i b_i(O_1)$
 $\psi_1(i) = 0$

2) Induktion / Reduktion

$\arg \max$ vs. \max
 $A(x) = 100 - (x-6)^2$
 $\max_x f(x) = 100$
 $\arg \max_x f(x) = 6$

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

$$\psi_{t+1}(j) = \arg \max_i [\delta_t(i) a_{ij}] \quad \text{--- merkt sich für jeden Zeitpunkt und j. Zustand welcher Vorgängerzustand an der Maximierung beteiligt war}$$

3) Terminierung:

- Wähle unter den Ziellnoten die höchste Wahrts. aus:

$$p^* = \max_i [\delta_T(i)] = p(Q, O | \lambda)$$

- der Ziellnoten(inde)r:

$$q_T^* = \arg \max_i [\delta_T(i)]$$

4) Pfadermittlung (Bestimmen der Zustandsfolge rückwärts):iteriere über alle $t = T-1, T-2, \dots, 1$

$$q_t^* = \psi_{t+1}(q_{t+1}^*)$$

nur rückwärts möglich, da nur Vorgängernoten gespeist werden

 q_{T+1}^* am Anfang gleich q_T^*
Knoten speichern kein Wissen über NachfolgerProblem 3

- schwierigstes der drei Probleme
- kein analytischer Lösungsweg bekannt
- > bei gegebener endlicher Ausgabesequenz O gibt es keinen optimalen Weg, um die Modellparameter zu schätzen

Möglichkeit:

Lokale Maximierung von $p(O | \lambda)$ mit iterativer Prozedur, z.B.:Baum - Welch - AlgorithmusGegeben: Trainingssequenz O_{Training} und Hypothesenraum für Modelle $\lambda = \{\Sigma, V, A, B, \pi\}$

Gesucht: Modell, das die Daten am besten erklärt:

$$\bar{\lambda} = \arg \max_{\lambda} P(O | \lambda)$$

Ansatz: Hypothesenraum wird so gewählt, dass die Anzahl der Zustände vorgegeben wird.

Nur die stochastischen Modellparameter werden angepasst: $\bar{\lambda} = \{\bar{A}, \bar{B}, \bar{\pi}\}$ mithilfe von Expectation Maximization (EM)

(6)

Idee:

- beginne mit zufälligen Modell λ , berechne $P(O_{\text{Training}} | \lambda)$
- bestimme die erwartete Anzahl von Zustandsübergängen (zwischen 2. t und 2 sowie weg von einem Zustand)
- Neuschätzung der Übergangs- und Emissionswahrscheinlichkeiten:
Berechnung eines neuen Modells
- iteriere bis (lokales) Maximum erreicht ist

Algorithmus1) Berechnen der temporären Variablen (Posterior)

Nutzen des Forward-/Backward-Algorithmus

- a) Wahrsch. für das Befinden in Zustand i zum Zp. t gegeben O und λ

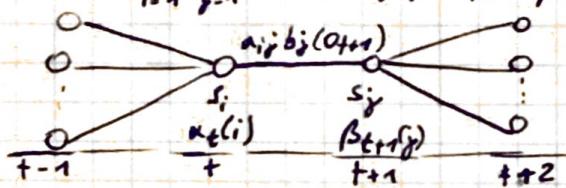
$$\gamma_t(i) = \frac{\alpha_t^e(i) \beta_t(i)}{\sum_{i=1}^N \alpha_t(i) \beta_t(i)}$$

jed. bezeichnet w. zum Zp. t das Präfix $O_1 O_2 \dots O_t$ beobachtet zu haben und im Zustand s_i es zu sein

- b) Wahrsch. eines Zustandsübergangs von Zustand i nach j zum Zp. t gegeben O, λ

betrachtet alle seine Vorgänger $s_t(i, j) = \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)$ ← betrachtet alle seine Nachfolger

$$\sum_{i=1}^N \sum_{j=1}^N \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j) \leftarrow \text{Normalisierung mit Gesamtwahrsch. aller States für Ausgesequenz } o_1 \dots o_T$$

2) Updates durchführen (Erwartungswerte für Maximierung nutzen)

$$a) \pi_i^* = \gamma_t(i)$$

$$b) a_{ij}^* = \frac{\sum_{t=1}^{T-1} s_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \quad \leftarrow \text{Erwartete Anzahl Zustandsübergänge von Zustand } s_i \text{ nach } s_j$$

← Erwartete Anzahl } s_i besucht

$$c) b_i^*(u) = \frac{\sum_{t=1}^T 1_{O_t=u} \gamma_t^e(i)}{\sum_{t=1}^T \gamma_t^e(i)} \quad \leftarrow \begin{array}{l} \text{Erwartete Anzahl, für die } \\ \text{die Beobachtung/Ausgabe } u \\ \text{gleich dem Ausgabezeichen/Vokabularzeichen } v_k \text{ ist} \end{array}$$

← Erw. Anzahl Zustandsüb. von s_i aus bew. wie häufig in s_i gewesen

P(D|h)

Es gilt $P(O|\lambda) \geq P(O|\tilde{\lambda})$ und es wird der Maximum Likelihood Estimate bestimmt

→ damit verbessert sich das Modell iterativ auf der Menge der Trainingddaten

→ Training abgebrochen wenn Verbesserungen nur noch minimal, keine Garantie für globales Maximum

→ es können auch andere Erwartungswerte für Updates eingesetzt werden
(nicht nur anhand Forward/Backward-Algo)

- Wann Viterbi und wann Baum-Welch-Algorithmus?

Wenn Modell bekannt und Zustandsfolge unbekannt → Viterbi

Wenn Modell unbekannt, kann Viterbi nicht eingesetzt werden da

(A, B, π) bekannt sein muss, um Viterbi-Algo. zu verwenden, stattdessen nutze Baum-Welch-Algorithmus.