Fakultät für Informatik
Übung Grundlagen der künstlichen Intelligenz
Wintersemester 2024/25

Philipp Dahlinger
philipp.dahlinger@kit.edu
Nicolas Schreiber
nicolas.schreiber@kit.edu

Übungsblatt 6

MDPs und HMMs

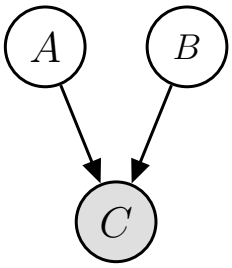
Abgabe online auf ILIAS bis 06. February 2025, 12:00

Für die Abgabe bitte ein PDF mit der eingescannten Lösung für Aufgabe 1 und 2 sowie das bearbeitete Notebook für Aufgabe 1 und 2 hochladen.

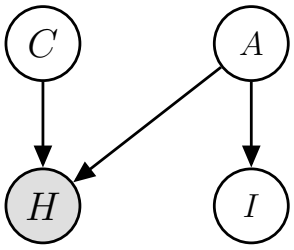
Aufgabe 1 *Graphical Models* (13 Punkte)

a) **"Explaining Away"** Das "Explaining away" ist ein gängiges Argumentationsmuster in probabilistischen grafischen Modellen mit einer "V-Struktur", wie in Abbildung 1a dargestellt. Angenommen, zwei Ursachen beeinflussen eine Wirkung positiv. Das "explaining away"-Prinzip besagt, dass eine weitere Konditionierung auf eine Ursache die Wahrscheinlichkeit der anderen Ursache verringert, wenn man die Wirkung berücksichtigt.

Betrachten wir das grafische Modell für eine medizinische Diagnose zwischen Erkältung und Allergie in Abbildung 1b und wenden wir das Konzept des "Explaining Away" an. Die Zufallsvariablen C, A, H und I entsprechen jeweils einer Erkältung, einer Allergie, einem Husten und juckenden Augen.



(a) "Explaining Away" Struktur



(b) Das grafische Modell der medizinischen Diagnose

Die gemeinsame Verteilung kann faktorisiert werden als $p(C, A, H, I) = p(C)p(A)p(H \mid C, A)p(I \mid A)$. Die jedem dieser Faktoren entsprechenden Wahrscheinlichkeitsverteilungen sind in den folgenden Tabellen angegeben.

C	$p(C)$
0	0.75
1	0.25

A	$p(A)$
0	0.9
1	0.1

A	I	$p(I\|A)$
0	0	0.75
1	1	0.75
0	1	0.25
1	0	0.25

C	A	H	$p(H\|C,A)$
0	0	0	0.75
0	1	1	0.75
1	0	1	0.75
1	1	1	0.75
0	0	1	0.25
0	1	0	0.25
1	0	0	0.25
1	1	0	0.25

Ihre Aufgabe lautet wie folgt: 4 P.

- Berechnen Sie die bedingte Wahrscheinlichkeit $p(C = 1 \mid H = 1)$.
- Berechnen Sie die bedingte Wahrscheinlichkeit $p(C = 1 \mid H = 1, I = 1)$.
- Kann Erkältung (C) durch juckendes Auge (I) "wegerklärt" werden? Begründen Sie.

b) **State Estimation (Coding Task)** Implementieren Sie den Forward und Backward Algorithmus für ein HMM Modell in Python. Die Forward und Backward Nachricht wird benutzt, um eine State Estimation durchzuführen (filtering und smoothing). Benutzen Sie dafür das beigefügte Jupyter Notebook. 9 P.

Aufgabe 2 *MDP, Value and Policy Iteration* (7 Punkte)

In der letzten Episode von Storage Wars hast du den Zuschlag für einen Lagerraum bekommen, der zuvor einem japanischen Geschäftsmann gehörte. In einer Kiste mit alten Sachen findest du ein kleines Spielzeug mit einem Bildschirm. Sofort erkennst du, dass es sich um einen Prototyp eines Tamagotchi [1] handelt. Da du vor kurzem von duften KI Algorithmen gehört hast, fängst du an, an die Berechnung einer optimalen Strategie zu denken.

Im Ausgangszustand ist Filibert (wie auch immer du auf diesen Namen kamst) *glücklich*. Unabhängig von seinem Zustand kannst du zwei Aktionen ausführen. Du kannst Filibert *spielen* lassen, oder ihn *schlafen* schicken.

So lange Filibert *glücklich* ist bringt *spielen* eine Belohnung von $+16$ und mit einer Wahrscheinlichkeit von 75 % bleibt er *glücklich*. Filibert *schlafen* zu legen während er *glücklich* ist ändert seinen Zustand nicht und gibt eine Belohnung von 0.

Nach einem anstrengenden Tag voller Spiele wird Filibert *müde*. Genauer gesagt hat *spielen* eine 10% Chance Filibert *müde* zu machen, was jedoch immer noch eine Belohnung von $+16$ bedeutet. Im *müden* Zustand wäre das naheliegendste zu *schlafen*. Manchmal ist er sehr müde, so, dass er trotz Schlaf in 25% der Fälle weiterhin müde bleibt. In den restlichen 75% wird er wieder *glücklich*. Allerdings ist *schlafen* langweilig und bringt daher in jedem Fall eine Belohnung von 0. Außerdem kannst du entscheiden Filibert spielen zu lassen. Im *müden* Zustand zu *spielen* macht Filibert mehr und mehr müde, immer resultierend in *Tod* durch völlige Verausgabung und einer Belohnung von -100. *Tot* zu sein ist ein endgültiger Zustand und kann nur durch einen Neustart des Systems zurückgesetzt werden.

Hinweise: Der reward für das Auswählen einer Aktion in einem Zustand ist in dieser Aufgabe unabhängig vom Zustand in den im nächsten Zeitschritt übergegangen wird. Genauer gesagt ist zum Beispiel der reward $r(s = s_1, a = a_1, s' = s_i) = r \,\forall\, i$ und wir können verkürzt $r(s, a, s') = r(s, a)$ schreiben. Starte mit einer Value Funktion $V_0(s) = 0 \,\forall s$. Per Definition ist $V_t(s = dead) = 0 \,\forall\, t$. Die zeitunabhängige optimale Strategie für ein Problem mit unendlichem Horizont ist gegeben durch $\pi^*(s) = \arg\max_a Q^*(s, a)$.

a) **MDP** Entwerfe mit den oben gegebenen Hinweisen das MDP welches die Zustände, Transitionen und Belohnungen von Filibert beschreibt (ähnlich zum Racing Beispiel aus der Vorlesung). Zeichne für jeden Zustand (*glücklich*, *müde*, *tot*) einen Kreis und verbinde sie mit den zu den Aktionen (*spielen*, *schlafen*) passenden Pfeilen um die Transitionen auszudrücken. Notiere zusätzlich die Übergangswahrscheinlichkeiten und die zugehörigen Belohnungen. 1 P.

b) **Value Iteration** Nachdem du nun das dem Problem zugrunde liegende MDP herausgefunden hast, kannst du mit dem Berechnen der optimalen Strategie für einen unendlichen Horizont beginnen. Erinnere dich an die Defintion der Value Funktion

$$V^*(s) = \max_a Q^*(s, a)$$

mit

$$Q^*(s, a) = r(s, a) + \gamma \sum_{s'} P(s' \mid s, a)\, V^*(s').$$

[1] `https://de.wikipedia.org/wiki/Tamagotchi`

Setze $\gamma = 0.7$ und berechne zwei Iterationen von Value Iteration. Schreibe die optimale Policy für $t = 0$ und $t = 1$ auf. 3 P.

c) **MDPs (Coding Task)** Implementieren Sie die Value Iteration und Extraktion der optimalen Policy in Python. Benutzen Sie dafür das beigefügte Jupyter Notebook. 3 P.