OPGAVE 1: BAYES RULE

Bayes rule speelt een belangrijke rol in AI (denk bijvoorbeeld aan Naive Bayes bij machine learning). In week 5 komt Bayes ook weer terug. Daarnaast is het goed om Bayes rule te begrijpen, want ook in de media en bij justitie[1] maken mensen denkfouten omdat ze niet de invloed van de z.g.n. a-priori kansen begrijpen. Onderstaande casus komt uit het boek 'Thinking fast and slow' van Kahneman.

In de stad zijn twee taxibedrijven, de Groene en de Blauwe, actief in de stad. On de stad is afgelopen nacht een hit-and-run misdrijf gepleegd, en hierbij was een taxi betrokken.

Je krijgt de volgende gegevens: 85% van de taxi's in de stad zijn Groen en 15% zijn Blauw. Een getuige identificeerde de taxi als Blauw. De rechtbank testte de betrouwbaarheid van de getuige in dezelfde omstandigheden als de avond van het ongeval en concludeerde dat de getuige elk van de twee taxi's 80% van de tijd correct identificeerde, maar dus 20% van de tijd niet.

Wat is de waarschijnlijkheid dat de taxi die bij het ongeluk betrokken was Blauw was? Geef niet alleen het antwoord maar ook de berekening. Maak hierbij gebruik van de letters H en E, als volgt:

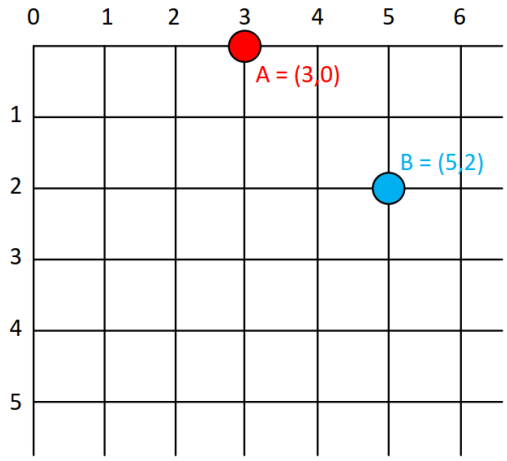
- H = de taxi was blauw (de hypothese)
- E = de getuige identificeerde de taxi als blauw (het bewijs of de test)

[1] De zaak-Lucia de Berk is een beruchte gerechtelijke dwaling. Het gaat om een rechtszaak tegen de kinderverpleegkundige Lucia de Berk, die beschuldigd werd van meerdere moorden. Na 6,5 jaar onterechte detentie werd Lucia de Berk in 2010 vrijgesproken. Bij de veroordeling speelde statistische bewijs een grote rol. De econometrist Aart de Vos was de eerste die in de NRC op basis van Bayes rule aantoonde dat deze bewijsvoering niet juist was.

OPGAVE 2: MARS ROBOT

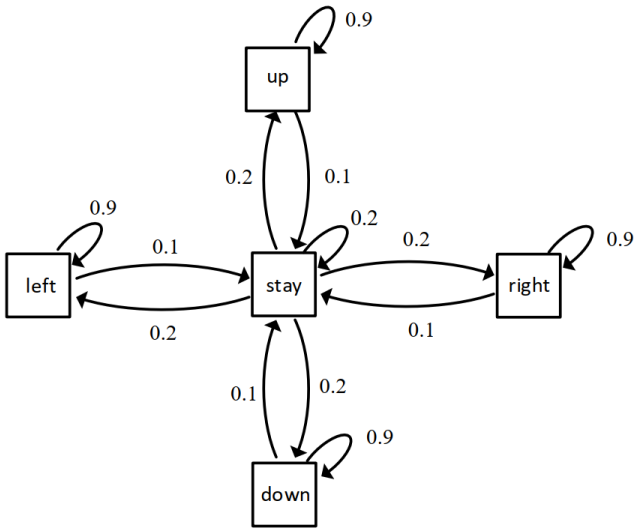
Stel dat een Mars-robot rondzwerft in een gebied dat gemodelleerd is als een rooster van 15 bij 15. De exacte locatie van de robot weten we niet, maar we krijgen wel onnauwkeurige metingen door via de sensoren op Mars. In deze opgaven gebruiken we een Hidden Markov Model om de bewegingen van de robot te modelleren.

De bewegingen van de robot zijn vrij voorspelbaar. Zie figuur 1. Bij elke tijdstap doet de robot één van de volgende vijf acties: hij blijft staan, gaat naar links of naar rechts, of gaat omhoog of naar beneden. De positie van de robot op tijdstip t hangt dus af van zijn positie op tijdstip t-1 *en van de actie* op tijdstip t-1.



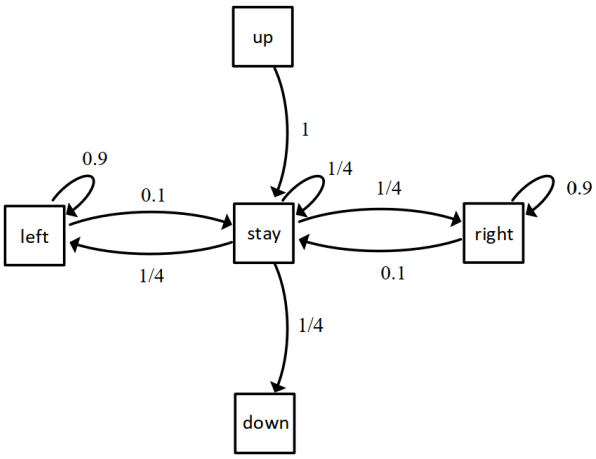
Figuur 1: Bewegingen van de robot op een rooster.
In positie A: L, R, D, S en in positie B: L, R, D, U, S. Zie ook figuur 2.

De acties van de van de robot zijn afhankelijk van de vorige actie. Zie het transitiediagram in figuur 2. Als we aannemen dat de robot zich *niet* op de grens van het rooster bevindt dan is zijn actie als volgt: als de vorige actie een beweging was (links, rechts, omhoog, omlaag), dan beweegt de robot weer in dezelfde richting met waarschijnlijkheid 0.9 en blijft staan met waarschijnlijkheid 0.1. Stond de robot stil, dan blijft hij weer staan met waarschijnlijkheid 0.2, of beweegt in een richting met waarschijnlijkheid 0.2. Merk op dat bijvoorbeeld (links, rechts, links) geen geldig pad is, maar (links, stay, rechts) wel.



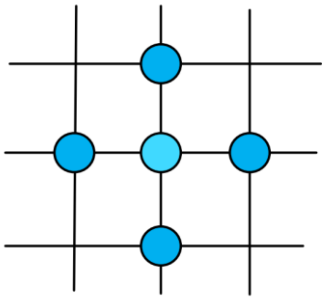
Figuur 2: Transitiediagram (robot bevindt zich niet op een grens)

De robot kan niet over de grens van het rooster heen. Wanneer de robot zich aan de bovenkant van het rooster bevindt, bijvoorbeeld op positie A = (3,0) is (zie figuur 1) dan is een actie ‘up’ niet mogelijk. Er blijven dan nog vier mogelijke acties over. In dat geval is het diagram in figuur 3 van toepassing: was de vorige actie van de robot ‘up’, dan blijft hij op A. Verder kan de vorige actie niet ‘down’ geweest zijn, want dan zou de robot zich buiten het rooster begeven. Eenzelfde model en redenering geldt ook voor de andere grenzen. Zie figuur 3.



Figuur 3: Transitiediagram (robot bevindt zich in positie A van figuur 1)

De exacte positie van de robot is onbekend. We hebben alleen toegang tot metingen van onnauwkeurige sensoren die in berichten naar de aarde worden verzonden. Wanneer de robot zich in een positie P bevindt, dan kunnen de sensoren P doorgeven, maar ook één van de buren van P. Deze kansen zijn uniform verdeeld, dus elk van deze mogelijke metingen (waarnemingen) hebben een kans van 1/5. Bevindt de robot zich in (7,7) dan is er een kans van 0.2 dat de sensoren (7,8) doorgeven. Bevindt de robot zich op de grens, dan geven de sensoren alleen posities door binnen het rooster (er blijven dan vier i.p.v. vijf posities over). Zie figuur 4.



Figuur 4: Wanneer de werkelijke positie in het centrum is,
dan zijn er vijf mogelijke waarnemingen met gelijke kansen.

Op Blackboard is de file hmm.zip te vinden. Wanneer op de startknop wordt gedrukt dan zal de robot een geheel willekeurig pad uitvoeren. Daarnaast bevat model.py al enkele functies die nodig zijn voor het verder implementeren van het programma.

Bij de opgaven (a) t/m (h) mag je aannemen dat het rooster oneindig groot is, dus dat er geen problemen zijn met de randen van het bord.

a) De functie transition_model is al gegeven. Gebruik deze functie om de robot 100 stappen door het rooster te laten maken. Je mag hierbij aannemen dat de startpositie van de robot (7,7) is met (vorige) actie ‘stay’.
b) Wat is de kans dat een *willekeurige* move 'left' is? Wat is de kans dat een *willekeurige* move 'stay' is?
c) Stel dat de robot in de toestand ‘stay’ zit. Wat is de kans dat de robot *precies* 3 keer (dus niet 4 keer) een ‘stay’ achter elkaar uitvoert?

d) Wat is de gemiddelde tijdsduur dat het model in de toestand 'stay' zit. Dus als de robot oneindig lang door zou lopen, hoe lang zit de robot dan gemiddeld in de toestand 'stay'?
Tip: Bedenk dat de gemiddelde of verwachte waarde = de gewogen som over alle waarden, waarbij de gewichten de kansen op die waarden zijn. De formule is te vinden in 'A Tutorial on Hidden Markov Models' p 259.
e) Hoeveel mogelijk paden zijn er van 10 stappen? Merk op (1) dat een 'stay' niets toevoegt aan een pad en (2) als je bijvoorbeeld een 'left' hebt gedaan, de volgende actie een 'left' of een 'stay' moet zijn. Schrijf hiervoor een kort (recursief) programma dat het aantal paden telt.
f) Het doorlopen van alle paden bij (e) kan je zien als het doorlopen van een boom. Hoe groot is dan de (gemiddelde) branching factor van deze boom? (Ik kom op 2.19).
g) Hoeveel paden zijn er van (7,7) naar (10,10) met de beperking dat een pad precies 6 stappen lang is?
h) Hoeveel paden zijn er van (x1,y1) naar (x2, y2) met de beperking dat de lengte van het pad gelijk is aan de Manhattan afstand (= het kortste pad)?

We ontvangen nu berichten van de sensoren, zodat we een schatting kunnen gaan maken van het werkelijke pad dat de robot aflegt met het Viterbi-algoritme. We nemen aan dat initieel elke positie even waarschijnlijk is en dat de initiële actie van de robot 'stay' is. Je ziet in de code dat een toestand een combinatie is van positie en vorige actie. Als we als voorbeeld positie (5,4) waarnemen dan kan zijn er maar vijf mogelijke posities waar de robot zich op dat moment kan bevinden. Als de vorige acte een 'links' was, dan kan de huidige actie alleen maar een 'links' of een 'stay' zijn. Omdat we alle deze kansen weten gegeven een reeks van waarnemingen kunnen we ook het meest waarschijnlijke pad berekenen.

i) Als er een waarneming (5,5) is, is wat is de kans dat dit ook juist is?
j) In de file 'observations_v1.txt' zijn in de laatste kolom de observaties (x, y) opgenomen (voor 25 stappen). Wat is het meest waarschijnlijke pad geweest? M.a.w. implementeer het Viterbi-algoritme. Diverse helper-functies zijn al gegeven.
Tip: het vermenigvuldigen van getallen tussen 0 en 1 wordt al snel heel onnauwkeurig. Beter is het daarom te werken met logaritmen. Vermenigvuldigen wordt dan optellen, immers ln(ab) = ln(a) + ln(b).

k) Waar in je code (regel nummers) is zichtbaar dat de complexiteit van Viterbi O(N^2) is?
l) Als de kans op een toestand in de Viterbi graaf heel klein is, zou je dan niet het pad naar deze toestand kunnen negeren? M.a.w is pruning mogelijk?
m) Met Viterbi kunnen we ook omgaan met ontbrekende waarnemingen. Dit is te zien in de file 'observations_v2.txt'. Gebruik nu deze file voor de waarnemingen. Wat is het meest waarschijnlijke pad geweest?
n) Hoeveel fouten maakt het Viterbi algoritme (verschil werkelijke pad en geschatte pad)? Bepaal dit met en zonder missende observaties (v1 en v2).
o) Wat is de complexiteit van het Viterbi-algoritme? Motiveer je antwoord. In hoeverre verschilt deze complexiteit van Dijkstra's kortste-pad algoritme?
p) Zijn er nog manieren om je programma (implementatie van Viterbi) te versnellen?