FACULTEIT INGENIEURSWETENSCHAPPEN
EN ARCHITECTUUR

Kunstmatige intelligentie

Bert De Saffel

Master in de Industriële Wetenschappen: Informatica Academiejaar 2018–2019

Gecompileerd op 23 maart 2019

UNIVERSITEIT
GENT

Inhoudsopgave

Hoofdstuk 1

Kunstmatige intelligentie

- Twee doelen van kunstmatige intelligentie:
  - Het laten overnemen, door machines, van taken waarvoor intelligentie vereist is.
  - Studie van natuurlijke intelligentie.
- Twee vormen om kennis in te brengen in een computersysteem:
  - Expliciete kennis.
  - Kennis kan zelf verworven worden.

1.1 Kunnen machines denken?

- Twee voorbeelden.
  - ELIZA:
    - Computerprogramma dat zich voordoet als een pyschotherapeut.
    - Maakt gebruik van simpele vervangingsregels.
    - Probeert de conversatie zo te sturen zodat de echte persoon het meest moet vertellen.
  - Chinese kamer:
    - Denkrichting die aantoont dat een entiteit eerst iets moet begrijpen, vooraleer er van intelligentie sprake is.
      1. Iemand die geen Chinees kent wordt in een kamer gebracht.
      2. Door een luik krijgt hij briefjes in het Chinees aangereikt, en de bedoeling is dat hij daar schriftelijk een zinnige antwoord op teruggeeft.
      3. De persoon krijgt handboeken waarin conversieregels staan.
    - De proefpersoon volgt mechanisch de regels vanuit het handboek, zodat hij wel intelligent gedrag vertoont, maar de berichten niet begrijpt.
- **Denken is elke vorm van complexe informatieverwerking waarvan de onderliggende mechanismen niet volledig gekend zijn.**
- **Turingtest**:
  - Proefpersoon kan contact maken met twee entiteiten: een mens en een machine, maar hij weet niet wie de mens of machine is.
  - De proefpersoon kan eender welke vragen stellen aan beide entiteiten.
  - Als de proefpersoon er niet in slaagt om na zijn vragenronde de entiteit aan te duiden die een machine is, dan is de machine geslaagd voor de Turingtest.

1.2 Toepassingen van AI en data mining

- **Classificatie:**
  - Stel een verzameling van k klassen.
  - Een bepaalde invoer met gelinkt worden aan één van die klassen.
  - Harde classificatie: beperkt aantal duidelijk van elkaar gescheiden klassen. Hier spreekt men ook van patroonherkenning.
  - Zachte classificatie: continue overgang van de klassen.
- Toepassingen:
  - Aanbevelingssystemen.
  - Kwaliteitscontrole.
- Probleemgestuurd: uitgaande van een probleem een oplossing zoeken.
- Datagestuurd: vanuit bestaande informatie problemen zoeken die ermee opgelost kunnen worden. Dit wordt ook data mining genoemd. Vaak moet de data eerst gereorganiseerd worden vooraleer de informatie nuttig wordt.

1.3 Leren

- Moderne AI houdt zich bezig met systemen met een zeer groot aantal aanpasbare parameters. Zulke systemen noemt men **massief lerende systemen**.
- **Voorbeelden** van massief lerende systemen:
  - Neurale netwerken: trachten het biologische denksysteem na te bootsen.
  - Hidden Markov Model: wordt gebruikt bij de analyse van allerhande sequenties, waarbij de toestand soms onbekend is.
- Parameters hebben niet noodzakelijk een betekenis, en is daarom ook onmogelijk om ze met de hand in te voeren. Daarom laat men een systeem leren, met behulp van **drie methoden**:
  - Algoritmisch leren: Er wordt gedemonstreerd hoe een bepaalde actie moet uitgevoerd worden. Het systeem kan hierna deze actie inoefenen door het herhalen van deze instructies. Deze vorm komt goed overeen met het programmeren van een computer.
  - Leren met supervisie: Hier wordt er geen gebruik gemaakt van een algoritme maar eerder van voorbeelden. Deze voorbeelden worden een leerverzameling genoemd en bevatten inputgegevens die het systeem moet leren herkennen, met de daarbij horende resultaten. Er wordt een verband opgelegd tussen een bepaalde input en output.
  - Leren zonder supervisie: Dit gebeurt gedeeltelijk algoritmisch aangezien er enige instructies nodig zijn om de machine op gang te krijgen. De machine zal nadien zelf experimenteren wat er gebeurd bij het aanpassen van verschillende parameters. Het leren gebeurt dus niet met voorbeelden, maar uit eigen ervaring. Hier is er dan ook geen verband tussen het resultaat en de verschillende deeltaken, maar er is wel een algemeen idee wat er aangeleerd moet worden.

1.4 Classificatie

- Classificatie is het mappen van een bepaalde input op een klasse.
- We spreken van een **item** dat we moeten klasseren.
- Dit item wordt gekarakteriseerd door een aantal **meetwaarden**.
- Twee soorten meetwaarden:
  - Sommige metingen kunnen een groot aantal waarden opleveren die voorgesteld kunnen worden als een getal.
  - Andere metingen hebben maar een beperkt aantal waarden, zoals de indeling van van categorieën.
    - ◇ Deze kunnen omgezet worden zodat ze antwoorden zijn op ja-nee vragen, zodatze geconverteerd kunnen worden naar 0 of 1.
    - ✓ Nu zijn alle meetwaarden getallen.
- Aangezien dat alle meetwaarden getallen zijn → standaardvorm: de computer heeft een aantal klassen en moet een getallenrij die de meetwaarden voor een bepaal item bevat toewijzen aan één van de klassen.
- Hoe worden de getallenrijen weergegeven? Een aantal notaties:
  - De n-dimensionale Euclidische ruimte is de de verzameling vectoren met n reële coördinaten.
  - Zo een vector wordt voorgesteld door een vette letter: $\mathbf{v} = (v_1, ..., v_n)$.
  - Soms vanaf 0 beginnen, zodat we $n+1$-dimensionale vectoren hebben: $\mathbf{v} = (v_0, v_1, ..., v_n)$. De waarde v_0 krijgt een speciale betekenis.
  - Reeële getallen die niet deel uitmaken van een vector worden weergegeven met hoofdletters: A, B, ...
  - De nulvector: $0 = (0, ..., 0)$.
  - Het inproduct:

$$\mathbf{v} \cdot \mathbf{u} = \sum_{i}^{n} v_i u_i$$

Het inproduct is lineair: $(A\mathbf{v}) \cdot \mathbf{u} = A(\mathbf{v} \cdot \mathbf{u})$ en $(\mathbf{u} + \mathbf{v}) \cdot \mathbf{x} = \mathbf{u} \cdot \mathbf{x} + \mathbf{v} \cdot \mathbf{x}$.
Het kan ook gedefinieerd worden als

$$\mathbf{v} \cdot \mathbf{u} = ||\mathbf{v}||\ ||\mathbf{u}|| \cos\theta$$

met θ de hoek tussen vector $\mathbf{v}$ en $\mathbf{u}$.
Als $\mathbf{v}$ en $\mathbf{u}$ orthogonaal zijn, dan is het inwendig product, aangezien $\cos(\pi/2) = 0$;

$$\mathbf{v} \cdot \mathbf{u} = 0$$

Als $\theta = 0$, dan

$$\mathbf{v} \cdot \mathbf{u} = ||\mathbf{v}||\ ||\mathbf{u}||$$

Hieruit volgt

$$\mathbf{u} \cdot \mathbf{u} = ||\mathbf{u}||^2$$

- $d(u, v) = ||\mathbf{u} - \mathbf{v}||$ is de lengte van de kortste weg van $\mathbf{u}$ naar $\mathbf{v}$. Hieruit volgt:

$$||\mathbf{u} + \mathbf{v}|| \leq ||\mathbf{u}|| + ||\mathbf{v}||$$

Het kwadraat van beide kanten geeft:

$$\mathbf{u} \cdot \mathbf{v} \leq ||\mathbf{u}|| \, ||\mathbf{v}||$$

Aangezien dat

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}|| \, ||\mathbf{v}||}$$

kan dit omgevormd worden tot de volgende ongelijkheid:

$$-1 \leq \frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}|| \, ||\mathbf{v}||} \leq 1$$

- De afstand en de cosinus geven vaak een goede indruk in hoeverre twee vectoren op elkaar lijken. De cosinus geeft een goede maat voor de afstand tussen twee genormaliseerde vectoren:

$$d\left(\frac{\mathbf{u}}{||\mathbf{u}||}, \frac{\mathbf{v}}{||\mathbf{v}||}\right)^2 = 2 - 2\cos(\mathbf{u}, \mathbf{v})$$

1.5 Informatie en beslissingsbomen

1.5.1 Informatie-inhoud

- Een bericht is enkel nuttig indien ontvanger een betekenis kan geven aan het bericht. De belangrijke elementen voor de informatie-inhoud is dus het bericht zelf en de kennis van de ontvanger.
- Met de kennis kan aan elk mogelijk bericht B een waarschijnlijkheid $P(B)$ toekennen. De informatie-inhoud wordt dan gedefinieerd door

$$-\log_2(P(B)) \text{ bits}$$

Voor $P(B) = 1$ is de informatie-inhoud 0 bits, wat logisch is aangezien de ontvanger niets heeft bijgeleerd van dit bericht.

! De informatie-inhoud van een bericht is niet altijd een geheel getal.

! De informatie-inhoud is nooit negatief.

- Voorbeeld: Stel dat een byte verwacht wordt, maar er is geen idee welke byte. Elke byte is even waarschijnlijk met kans 1/256. De informatie-inhoud van de byte die dan binnenkomt is $-\log_2(1/256)$ bits $= 8$ bits.
- Voorbeeld: Stel een alfabet van 4 letters: A, C, G en T. De waarschijnlijkheid dat ze voorkomen wordt weergegeven in tabel 1.1.

A	70,71 %
C	12,50 %
G	8,39 %
T	8,39 %

Tabel 1.1: De waarschijnlijkheden voor de letters A, C, G en T.

Als de ontvanger dit weet dan wordt de informatie-inhoud voor elke letter:

$$A : -\log_2(0,7071) = 0,5$$
$$C : -\log_2(0,1250) = 3,0$$
$$G : -\log_2(0,0839) = 3,575$$
$$T : -\log_2(0,0839) = 3,575$$

1.5.2 Beslissingsboom

- Elke knoop dat geen blad is bevat een vraag met een beperkt mogelijk aantal antwoorden.
- Elk mogelijk antwoord verwijst naar een kind van de knoop.
- Een item klasseren is een pad vanuit de wortel naar een blad, waarin de klasse staat.
- Hoeveel informatie kan een beslissingsboom geven?
  - Stel k klassen $K_1, K_2, ...K_k$.
  - Stel een verzameling S van items waarbij:
    - $A(S,i)$ het aantal elementen horend bij K_i is in de verzameling en,
    - $|S| = \sum_{i=1}^{k} A(S,i)$ het totaal aantal element is van S.
  - De informatie geleverd door een correcte klassering van alle element is dan:
    (groene formules moeten niet van buiten geleerd worden. Ze worden gegeven bij de examenvraag)

$$E(S) = \sum_{i=1}^{k} A(S,i)\left(-log_2\left(\frac{A(S,i)}{|S|}\right)\right)$$
$$= |S|log_2(|S|) + \sum_{i=1}^{k} A(S,i)(-log_2(A(S,i)))$$

- Het **Iterative Dichotomiser 3 (ID3)** algoritme is een inhalig algoritme dat een beslissingsboom opstelt vanuit een bepaalde dataset.
  - De wortel bevat de vraag die het meeste informatie oplevert.
  - Als het j-de attribuut de leerverzameling L in de deelverzamelingen $L_{j,1}, L_{j,2}, ..., L_{j,n}$ opdeelt, dan is de informatie geleverd door dit attribuut gelijk aan:

$$I(j) = E(L) - \sum_{m=1}^{n_j} E(L_{j,m})$$

  - Het attribuut wordt gekozen waarvoor $I(j)$ maximaal wordt.
  - ! Als $I(j) = 0$, dan behoren ofwel alle items tot dezelfde klasse, ofwel kan er op basis van het attribuut geen klassering gemaakt worden.
  - Na de constructie van de wortel moeten nog n_j deelbomen geconstrueerd worden.
  - Voorbeeld:
    - Veronderstel volgende informatie:
    - Voor elk attribuut kan de resulterende verdeling afgeleidt worden (tabel 1.2):

Beroepscategorie	Jonger dan 20?	Fraudeur?	risico?	frequentie
A	ja	ja	veilig	10
A	ja	nee	riskant	11
A	nee	ja	riskant	18
A	nee	nee	veilig	100
B	ja	ja	veilig	180
B	ja	nee	riskant	8
B	nee	ja	riskant	1
B	nee	nee	veilig	90
C	ja	ja	veilig	50
C	ja	nee	riskant	5
C	nee	ja	riskant	5
C	nee	nee	veilig	50
		Totaal veilig:		480
		Totaal riskant:		48
		Algemeen totaal:		528

Attribuut	waarde	# veilig	# riskant
(1) Beroepscategorie	A	110	29
	B	270	9
	C	100	10
(2) Jonger dan 20?	ja	240	24
	nee	240	24
(3) Fraudeur?	ja	240	24
	nee	240	24

Tabel 1.2: Resulterende verdeling.

◇ Voor elk attribuut kan nu $I(j)$ berekent worden, hier uitgewerkt voor de beroepscategorie:

$$
\begin{aligned}
I(1) &= E(L) - \sum_{m=1}^{n_1} E(L_{1,m}) \\
&= E(L) - (E(L_A) + E(L_B) + E(L_C)) \\
&\text{met} \\
E(L) &= 480(-\log_2(480/528)) + 48(-log_2(48/528)) \\
&= 232.054 \\
E(L_A) &= 110(-log_2(110/139)) + 29(-log_2(29/139)) \\
&= 102.702 \\
E(L_B) &= 270(-log_2(270/279)) + 9(-log_2(9/279)) \\
&= 57.3603 \\
E(L_C) &= 100(-log_2(100/110)) + 10(-log_2(10/110)) \\
&= 48.3447 \\
&\text{zodat} \\
I(1) &= E(L) - (E(L_A) + E(L_B) + E(L_C)) \\
&= 232.054 - 102.702 - 57.3603 - 48.3447 = 23.647
\end{aligned}
$$

◇ Voor $I(2)$ en $I(3)$ bedragen beide uitkomsten 0, aangezien er op basis van die attributen geen informatie kan achterhaald worden. Dit is logisch aangezien de verhouding veilige/risicohoudende klanten voor zowel leeftijd als fraudeur 10:1 is.

- ◇ Als wortel van de beslissingsboom wordt als criterium de beroepscategorie genomen.
- ◇ Voor zowel L_A, L_B en L_C moet apart dezelfde methode uitgevoerd worden. Het kan perfect mogelijk zijn dat op het tweede niveau bij keuze A eerst wordt gecheckt op fraude, maar bij keuze C eerst op leeftijd.
- ! Bij L_C levert geen enkel van de twee attributen informatie op, zodat er random moet gekozen worden:

Attribuut	waarde	# veilig	# riskant
(2) Jonger dan 20?	ja	50	5
	nee	50	5
(3) Fraudeur?	ja	50	5
	nee	50	5

- ◇ Verfijningen:
  - ∗ Invoeren van een drempelwaarde voor de informatiewinst. Als deze te klein is wordt er niet meer opgesplitst. Een blad kan dan meerdere klassen bevatten, elk met een waarschijnlijkheid.
  - ∗ Voor elk mogelijk paar van attributen de informatiewinst berekenen en, als deze te groot is, knopen maken met twee attributen.
  - ∗ Bij effectieve getallen kan ook een drempelwaarde ingevoerd worden. Alle items met een waarde kleiner dan deze drempelwaarde gaan naar links, de andere naar rechts.

1.6 Klasseren zonder leren

- Klassen worden niet vooraf gegeven.
- ! Geen leerverzameling aanwezig.
- Een **groep** van punten is wat door een expert als een groep beschouwd wordt.
  1. Twee punten behoren waarschijnlijk tot dezelfde groep als ze zeer dicht bij elkaar liggen. De expert definieert de afstandsfunctie.
  2. Deze eigenschap wordt **transitief** verdergezet. Twee punten $\mathbf{x}$ en $\mathbf{z}$ behoren tot dezelfde groep als een rij punten $\mathbf{y_1}, ..., \mathbf{y_n}$ bestaat zodanig dat $\mathbf{x}$ zeer dicht bij $\mathbf{y_1}$ ligt, $\mathbf{y_1}$ zeer dicht bij $\mathbf{y_2}$ ligt,..., en $\mathbf{y_n}$ zeer dicht bij $\mathbf{z}$ ligt.

1.6.1 k zwaartepunten

- Op voorhand opgeven dat er k klassen zijn.
- ! Een zwakte, aangezien men nu moet weten hoeveel klassen er op voorhand zijn. Twee problemen:
  - ○ Het aantal gekozen klassen is te groot zodat samenhangende groepen worden opgesplitst. Dit kan opgelost worden samenhorende groepen op het einde samen te nemen.
  - ○ Het aantal gekozen klassen is te klein zodat verschillende groepen samen worden genomen. Dit wordt opgelost door het algoritme meerdere malen uit te voeren met verschillende initialisaties.
- k **zwaartepunten** poogt een leerverzameling L op te delen in k groepen $G_1, ..., G_k$, waarbij k vooraf opgegeven is. Een klasse wordt voorgesteld door haar zwaartepunt m_i, waarbij n_i het aantal vectoren in G_i is:

$$m_i = \frac{1}{n_i} \sum_{x \in G_i} \mathbf{x}$$

- Een punt **z** wordt toegewezen aan een groep als volgt:
  1. Zoek uit de zwaartepunten $\mathbf{m_1}, ..., \mathbf{m_k}$ datgene dat het dichtst bij **z** ligt.
  2. **z** wordt dan toebedeeld aan de bijhorende klasse.
- Het resultaat is een **Voronoi-diagram**. Het negatieve aan zo een diagram is dat het enkele convexe groepen toelaat.

! Groepen die niet convex zijn worden door dit algoritme niet goed ingedeeld.

- Gebaseerd

1.7 Een toepassing: Watson

onbelangrijk

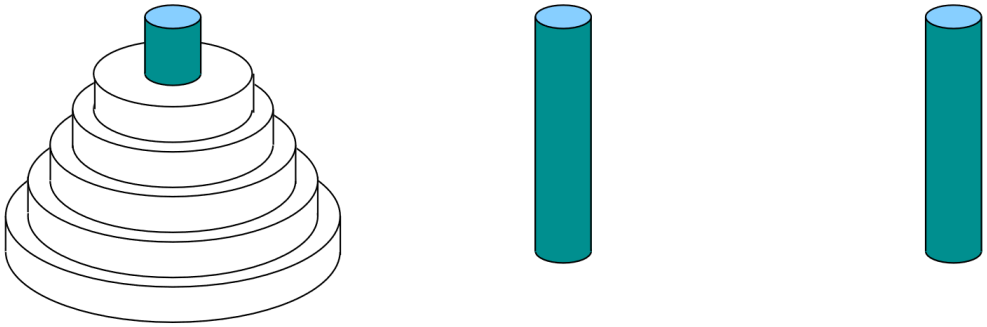
Hoofdstuk 2

Zoeken in zoekruimten

- Een zoekruimte tracht de manier van het menselijk redeneren na te bootsen.
- Drie elementen:
  - Een gerichte graaf, waarin de knopen toestanden zijn, en waarin er een verbinding van toestand a naar toestand b is als er in a een actie mogelijk is die tot b leidt.
  - Een begintoestand.
  - Een doel, dat bestaat uit een verzameling van toestanden.
- Soms is het afgelegde pad belangrijk, bijvoorbeeld wanneer de kost ook belangrijk is.

2.1 STRIPS

- **STanford Research Institute Problem Solver (STRIPS)** = een algemene probleemoplosser.
- Met behulp van een taal wordt het op te lossen probleem beschreven.
- Voorbeeld: de torens van Hanoi (Figuur 2.1).



Figuur 2.1: De torens van Hanoi.

  - Twee niveaus van positieve uitspraken:
    1. Nuldeordepredicaten: Dit zijn strings, zoals *DeLuchtIsBlauw* of *Veilig*.

2. Eersteordepredicaten: Deze zijn functies met een aantal parameters, die objecten aanduiden.

- Elke schijf hangt aan een bepaalde pin, er kan dus een predicaat van de eerste orde ingevoerd worden zoals bijvoorbeeld:

$$\texttt{HangtAan(schijf1, pin1)}$$

- Ook moet duidelijk gemaakt worden dat twee verschillende pinnen effectief verschillend zijn:

$$\texttt{IsNiet(pin1, pin2)}$$

- De begintoestand van de torens van Hanoi kan nu beschreven worden als volgt:

$$\begin{aligned}
&\texttt{HangtAan(schijf1, pin1)} \wedge \\
&\texttt{HangtAan(schijf2, pin1)} \wedge \\
&\texttt{HangtAan(schijf3, pin1)} \wedge \\
&\texttt{HangtAan(schijf4, pin1)} \wedge \\
&\texttt{HangtAan(schijf5, pin1)} \wedge \\
&\quad \texttt{IsNiet(pin1, pin2)} \wedge \\
&\quad \texttt{IsNiet(pin1, pin3)} \wedge \\
&\quad \texttt{IsNiet(pin2, pin1)} \wedge \\
&\quad \texttt{IsNiet(pin2, pin3)} \wedge \\
&\quad \texttt{IsNiet(pin3, pin1)} \wedge \\
&\quad \texttt{IsNiet(pin3, pin2)}
\end{aligned}$$

De predicaten worden gebonden met een logische EN. Ook is het nodig om de symmetrie te definiëren, aangezien STRIPS hier niets van af weet.

- Nu kan een actie gedefinieerd worden om een schijf te bewegen. Voor elke schijf moet dit apart gedaan worden. Hier wordt het uitgewerkt voor schijf 2:

$$\begin{aligned}
&\texttt{BeweegSchijf2(van, naar, tussen)}: \\
&\quad P: \ \texttt{IsNiet(van, naar)} \wedge \texttt{IsNiet(van, tussen)} \wedge \texttt{IsNiet(naar, tussen)} \\
&\qquad \texttt{HangtAan(schijf1, tussen)} \wedge \texttt{HangtAan(schijf2, van)} \\
&\quad + : \ \texttt{HangtAan(schijf2, naar)} \\
&\quad - : \ \texttt{HangtAan(schijf2, van)}
\end{aligned}$$

Elke actie bestaat uit een premisse P die voldaan moet zijn vooraleer de actie kan uitgevoerd worden. De lijst van de toe te voegen uitspraken wordt aangeduid met het $+$ symbool en de lijst met de te verwijderen uitspraken worden aangeduid met het $-$ symbool.

- Om de symmetrie van de begintoestand op te lossen zou een actie `IsSymmetrie(A, B)` aangemaakt kunnen worden als volgt:

$$\begin{aligned}
&\texttt{IsSymmetrie(A, B)}: \\
&\quad P: \ \texttt{IsNiet(A, B)} \\
&\quad + : \ \texttt{IsNiet(B, A)} \\
&\quad - : \ /
\end{aligned}$$

2.2 Efficiënt zoeken in zoekruimten

2.2.1 Breedte-eerst zoeken

- De **vertakkingsfactor** is het gemiddeld aantal nieuwe, nog niet bekeken buren van een onderzochte knoop en wordt voorgesteld door b (branching factor).
- Bij de torens van Hanoi is $b \approx 1,5$.
- Voor een diepte d worden ongeveer

$$1 + b + b^2 + ... + b^d = \frac{b^{d+1} - 1}{b - 1}$$

  knopen bezocht. Voor b voldoende groter dan 1 is dit ongeveer b^d.
- Als we tien zetten voor elke speler willen vooruitkijken bij schaken, moeten er $6^{20} \approx 3.6 \cdot 10^{15} = 3,6$ biljard knopen ontwikkeld worden.
- Indien zowel start- als eindtoestand bekend zijn, kan **bidirectioneel zoeken** toegepast worden. Er wordt vooruit gezocht vanuit het startpunt, en achteruit vanuit het doel, beide met breedte-eerst zoeken. Er moet dan in plaats van b^d knopen slechts $2b^{d/2}$ knopen ontwikkeld worden. Voor $b = 2$ en $d = 20$ krijgen we met breedte-eerst zoeken 10^6 knopen en met bidirectioneel zoeken ongeveer 2000.

2.2.2 Heuristieken

In plaats van breedte-eerst zoeken te gebruiken zou ook een manier kunnen gebruikt worden die de meest interessante knoop volgt, in plaats van ze blindelings af te lopen.

- Bij het zoeken in een graaf worden er aan elke knoop k twee waarden toegekend:
  1. De gekende kost $g(k)$ van de knoop:
     - Deze functie is een schatting van de werkelijke kost $g^*(k)$ van het kortste pad van de startknoop naar k.
     - Het wordt berekend door bij de kost van zijn voorganger v de kost $c(v \to k)$ op te tellen; de kost van de actie die v omzet in k.
     - Er geldt steeds $g(k) \geq g^*(k)$.
  2. De heuristische kost $h(k)$ van de knoop:
     - Deze functie is een schatting van $h^*(k)$, de kost om vanuit de knoop het doel te bereiken via het kortste pad.
- De som van deze twee waarden geven de geschatte kost, verpakt door de evaluatiefunctie f:

$$f(k) = g(k) + h(k)$$

! Interessante knopen heben een lage f waarde.

- **Beste-eerst zoeken**:

  (1) Steek de startknoop s in de verzameling niet-ontwikkelde knopen NOK met $g(s) = 0$. Geef alle andere knopen een voorlopige schatting $g(k) = \infty$.

  (2) Als NOK leeg is stop dan zonder oplossing, ga anders naar (3).

  (3) Zoek de knoop k uit NOK met de laagste evaluatiewaarde en verwijder hem uit NOK.

  (4) Als k in het doel zit, geef het pad naar k terug en stop; ga anders naar (5).

(5) Voor elke buur b van k: bereken $g_k(b) = g(k) + c(k \to b)$ en vergelijk $g_k(b)$ met de voorlopige waarde die $g(b)$ al gekregen had. Als $g_k(b) < g(b)$, vervang dan $g(b)$ door $g_k(b)$ en steek b in NOK als b daar niet in zit.

(6) Ga terug naar (2).

- De functie g is afhankelijk van het probleem, maar h hoeft dit niet te zijn. Er zijn verschillende mogelijkheden voor h. Er is wel de voorwaarde dat ze **toelaatbaar** moeten zijn. Dit wil zeggen dat het een pad vindt en dat dit pad optimaal is.

A* heuristieken

- Een heuristiek is $A*$ als $h(k) \leq h^*(k)$.
- Bewijs: indien er slechts een eindig aantal knopen k_i, met $g^*(k_i) \leq g^*(D)$ zijn, is een $A*$-heuristiek toelaatbaar.
  - Neem een $A*$ heuristiek h en bewijs dat er steeds een knoop k' is die voldoet aan de volgende voorwaarden:
    (1) k' zit in NOK.
    (2) k' ligt op een optimaal pad van s naar D.
    (3) De schatting $g(k')$ is correct: $g(k') = g^*(k')$.
  - Dit is geldig indien k' de startknoop is. Er is minstens één buur k'' die ook op het optimale pad ligt. Door k' te ontwikkelen wordt zijn plaats ingenomen door zijn opvolger k'' op het optimale pad. k'' voldoet aan (3), want
    $$g^*(k'') = g(k') + c(k' \to k'') = g_{k'}(k'')$$
  - Als k' de laatste knoop op het pad is die aan de voorwaarden voldoet dan is de nieuwe $g(k'')$-waarde kleiner dan de vorige en komt k'' zeker in NOK. Bovendien geldt
    $$f(k') = g(k') + h(k') \leq g^*(k') + h^*(k') = g^*(D)$$
  - Er kan geen enkele k ontwikkeld worden met $g(k) > g^*(D)$, want dan zou $f(k) > f(k')$ zijn.
  - Iedere keer als een knoop k ontwikkeld wordt, is er een kleinere waarde voor $g(k)$ die overeenkomt met de kost van een pad s naar k.
  - Ooit zal een doelknoop d gekozen worden voor ontwikkeling en die heeft $g(d) = f(d) \leq g^*(D)$, en is bijgevolg optimaal.

Monotone heuristieken

- Een heuristiek is monotoon als
  1. voor elk paar knopen geldt dat $h(v) - h(k) \leq c(v \to k)$.
  2. $h(d) = 0$ voor alle d in de doelverzameling D.
- Bewijs: Neem een willekeurig pad $k_0k_1, ..., k_n$ tussen twee knopen k_0 en k_n. Dan is de functie $g(k_i) + h(k_i)$ monotoon stijgend als functie van i. Hierin wordt $g(k_i)$ gerekend langs het pad vanaf het startpunt k_0.
  - Voor $i < n$ geldt dat $g(k_i) + c(k_i \to k_{i+1}) = g(k_{i+1})$
  - Ook geldt er dat $h(k_i) \leq h(k_{i+1}) + c(k_i \to k_{i+1})$.

- Deze twee termen optellen en de term $c(k_i \to k_{i+1})$ schrappen levert

$$g(k_i) + h(k_i) \leq g(k_{i+1}) + h(k_{i+1})$$

- Twee gevolgen:
  1. Een monotone heuristiek is $A*$.
  2. Een knoop k wordt nooit ontwikkeld voor $g^*(k)$ gekend is.
- Efficiëntie
  - Voor $h = 0$ krijgen we breedte-eerst zoeken.
  - Voor $h = h*$ krijgen we de optimale heuristiek, die rechtstreeks naar het doel leidt, en die ook monotoon is.
  - Hoe bepalen of een willekeurige monotone heuristiek beter is dan een andere?
    - Laatste ontwikkelde knoop is altijd d met $g^*(d) = g^*(D)$ en $h(d) = 0$.
    - Voor een monotone heuristiek moeten alle knopen ontwikkeld worden waarvoor $g^*(k)+h(k) < g^*(D)$, en geen enkele waarvoor $g^*(k) + h(k) > g^*(D)$.
    - Een monotone heuristiek h_1 ontwikkelt minder knopen dan een monotone heuristiek h_2 als $h_1(k) > h_2(k)$.
    - Als voor sommige k geldt dat $h_1(k) < h_2(k)$ en voor andere k $h_1(k) > h_2(k)$, kan een betere heuristiek opgebouwd worden:

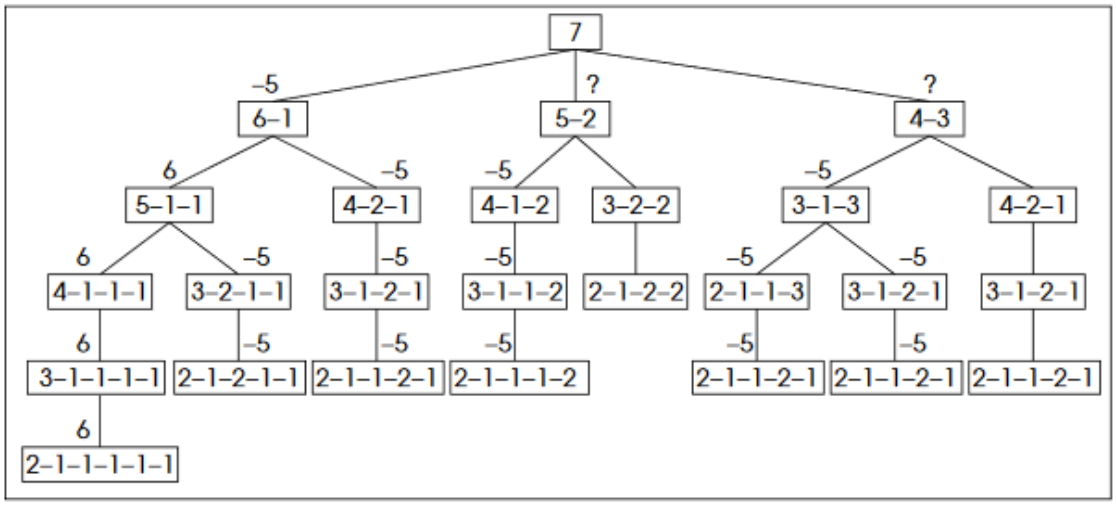
$$h_3 = \max\{h_1(k), h_2(k)\}$$

Relaxatieheuristieken

- Worden bekomen door het originele probleem sterk te vereenvoudigen.
- Er worden verbindingen toegevoegd in de graaf, wat overeenkomt met het schrappen van acties in de premisse.
- Het kortste pad in nieuwe graaf is zeker niet langer dan het kortste pad in de originele graaf. Deze heuristiek is dan ook A^*.

2.3 Spelbomen

- Wat doen indien er geen volledige kennis is over het resultaat van acties?
- Een **nulsomspel** (Engels: two player zero sum game)
  - Twee spelers.
  - Enkel winst mogelijk ten koste van de andere speler.
  - De winst van de ene speler is gelijk aan het verlies van de andere speler.
  - Elke speler probeert zijn winst te maximaliseren.
- Voorbeeld van een nulsomspel:
  - Op een tafel tussen twee spelers liggen een aantal stapels jetons.
  - De speler die aan de zet is moet een stapel nemen, en deze verdelen in twee ongelijke stapels.
  - De speler die niet meer kan zetten verliest.



Figuur 2.2: Een spelboom waarbij de startsituatie een stapel van 7 jetons is.

- De winnaar krijgt als winst het aantal punten gelijk aan het aantal overblijvende stapels, en de verliezer verliest dan ditzelfde aantal punten.
- Er kan een bijhorende boom (figuur 2.2) opgesteld worden die alle mogelijke zetten beschrijft.
- Noem de speler die begint *Max*, en de andere speler *Min*.
- Bij elke eindpositie is de winst of verlies van Max berekent.
- De overige posities kunnen van onder naar boven opgesteld worden:
  1. Als Max aan zet is dan nemen we de grootste score.
  2. Als Min aan zet is nemen we de kleinste score.
- Dit heet **minimax**. Er wordt gebruik gemaakt van $\alpha - \beta$ snoeien om oninteressante zetten (deelbomen) zeker niet te evalueren in de spelboom. Op die manier is er minder geheugen nodig. Het optimale pad is nooit echt gekend, maar er kunnen grenzen gesteld worden.
  - De waarde α staat voor de gewenste minimumscore voor Max.
  - De waarde β staat voor de gewenste maximumscore voor Min.
  - Algemeen wordt er een pad naar een blad gezocht waarbij zowel Max als Min steeds de beste zet kiezen.
  - Neem een knoop k en veronderstel een schatting f die $s(k)$ vervangt. De waarden van voorouders van die knoop kunnen veranderen. Stel $f > s(k)$:
    1. k is een minknoop. Als $f > s(b)\ \forall\ b$ broer van k, dan wordt de s-waarde van de ouderknoop vervangen door f. Nu wordt naar geval 2 gegaan.
    2. k is een maxknoop en geen wortel. De nieuwe $s-$waarde van de ouder is hoogstens f. Nu wordt naar geval 1 gegaan.

Hoofdstuk 3

Expertsystemen

- Kennis van een **echte expert** overnemen in een programma.
- De **gebruiker** moet ook enigszins kennis hebben. Hij moet de conclusies begrijpen en toepassen.
- **MYCIN**: een expertsysteem om te helpen bij de diagnose van bepaalde besmettelijke bloedziekten.
  - Is in staat om een betere diagnose te stellen dan een arts die geen expert is in het gebied.
  - De gebruiker moet wel een arts zijn die de diagnose begrijpt.
- Een expertsysteem maakt gebruik van **vuistregels**, waarop later nog **verfijningen** ingevoerd worden:
  1. werken met **onzekere conclusies**. Dit wordt ingevoerd om contradicties te behandelen indien deze voorkomen.
  2. De invoering van **frames**. Zorgt voor een betere orderning voor grote hoeveelheden kennis.

3.1 Eenvoudige systemen

- Een expertsysteem bestaat uit twee hoofdcomponenten
  1. Een **kennisbank**:
     - bevat **feiten**: dit zijn uitspraken die waar zijn.
     - bevat **regels**: dit heeft de vorm **als**<**premisse**>**dan**<**conclusie**>

       als AS=rond **en** MATERIAAL=staal
       dan SMERING=olie
  2. Een **afleidingssysteem**:
     - Elk probleem bestaat uit begingegevens en een doel.
     - Deze gegevens vormen een lijst van feiten die gekend zijn.
     - Het doel is ook een lijst van uitspraken.
     - Het doel is bereikt als één uitspraak uit de lijst is afgeleid uit de gegevens.
- Het afleidingssysteem kan op twee manieren werken.

- **Forward chaining:**
  - Kan in $O(g)$, met g de grootte van de regelbank.
  - Twee gegevensstructuren:
    - **LijstMetRegels(u)** die voor elke uitspraak u bijhoudt in welke regels ze in de premisse voorkomt.
    - **Premisseteller(r)** die voor elke regel r bijhoudt hoeveel uitspraken in de premisse nog niet gevalideerd zijn.
  - Werking:
    1. Initialiseer de gegevensverzameling en de doelverzameling.
    2. Voor elke regel r, initialiseer *Premisseteller(r)* op het aantal uitspraken in de premisse.
    3. Zolang de gegevensverzameling en het doel geen gemeenschappelijk element bevatten, en de gegevensverzameling niet leeg is:
       (a) Neem een uitspraak u uit de gegevensverzameling.
       (b) Voor elke regel r in *LijstMetRegels(r)*, verminder *Premisseteller(r)*. Als deze daardoor nul wordt, zet elke uitspraak uit de conclusie die nog niet behandeld is in de gegevensverzameling.
    4. Deel het resultaat mee aan de gebruiker.
  - Voorbeeld:
    - Stel volgende regelbank:
      1. Als X kwaakt en X eet vliegen - Dan X is een kikker.
      2. Als X kwettert en X zingt - Dan X is een kanarie.
      3. Als X is een kikker - Dan X is groen.
      4. Als X is een kanarie - Dan X is geel.
    - Stel nu dat we een huisdier hebben, Fritz, en er zijn twee dingen bekend over hem: hij kwaakt en eet vliegen. We willen nu de kleur weten.
    - Fritz wordt gesubstitueerd voor X in regel 1, zodat er besloten wordt dat hij een kikker is. Dit wordt dan toegevoegd aan de gegevens. We weten nu dat Fritz kwaakt, dat hij vliegen eet en dat hij een kikker is.
    - Fritz kan nu gesubstitueerd worden in regel 3, zodat er besloten wordt dat hij groen is.
  - Forward Chaining werkt van links naar rechts, startend vanaf de gekende feiten om tot een conclusie te komen.
  - ! In het algoritme kunnen regels toegepast worden die uiteindelijk niet bijdragen tot de oplossing. Men probeert dit efficiënter op te lossen met backwards chaining.
- **Backward chaining:**
  - Kan in $O(g)$, met g de grootte van de regelbank.

3.2 De constructie van een expertsysteem

- Twee soorten kennis nodig:
  - Kennis over het probleemgebied. Experten in het toepassingsgebied weten hoe ze problemen moeten oplossen, maar kunnen deze kennis niet vertalen in de vorm die een expertsysteem nodig heeft.
  - Kennis over expertsystemen. Een team die de kennis van de experten kan filteren en omvormen naar de juiste vorm.

3.3 Onzekerheid

- De relatie tussen premisse en conclusie van een regel is niet altijd zeker.
- Voorbeeld:
  - Een expertsysteem dat de oorzaak zoekt van problemen in een computer. Dit systeem kan een reparateur begeleiden die een defecte computer moet repareren.
  - De reparateur moet melden wat de symptomen zijn. Het systeem moet de actie bepalen die moet ondernomen worden om het probleem op te lossen. Mogelijke regels zijn bijvoorbeeld:

    als SYMPTOOM=computer doet niets bij opstarten
    dan PROBLEEM=voeding defect

    als SYMPTOOM=voeding defect
    dan PROBLEEM=vervang voeding

  - Maar soms kan een probleem meerdere oorzaken hebben, waarbij sommige meer waarschijnlijkheid hebben dan anderen:

    als PROBLEEM=bootloader hangt
    dan OORZAAK=schijf defect **of** OORZAAK=bootloderinstallatie **of** ...

  - Er wordt dan een numerieke factor toegekend, die de waarschijnlijkheid aanduidt.
  - **Hoe wordt de numerieke waarde bepaalt?**
    1. Vage verzamelingen: **Niet te kennen**.
    2. Bayesiaans redeneren:
       - Stel volgende regels:

         als het regent op dag x
         dan is de kans dat het op dag x + 1 regent 0,8

         als het niet regent op dag x
         dan is de kans dat het op dag x + 1 regent 0,3

       - Statistisch redeneren kan toegepast worden in expertsystemen als er voldoende statistische informatie is:
         - De kans dat het morgen en overmorgen regent: $0,8 \times 0,8 = 0,64$.
         - Da kans dat het morgen niet regent, maar overmorgen wel: $0,2 \times 0,3 = 0,06$.
       - Stel dat we nu twee uitspraken a en b hebben met $p(a) = 0,8$ en $p(b) = 0,2$.
       - Wat is de kans op $p(a$ **en** $b)$? Deze waarde kan tussen 0 en 0,2 liggen.
         Als a = het regent morgen en b = het regent niet morgen, dan is $p(a$ **en** $b) = 0$.
         Als a = het regent morgen en b = het regent morgenochtend, dan is $p(a$ **en** $b) = 0,2$
       - ! Bayesiaans redeneren is geen aangewezen methode.
    3. Theorie van zekerheidsfactoren:
       - Er wordt een zekerheidsfactor berekent op basis van woorden die een expert uitdrukt:
       - Voor een uitspraak a wordt de zekerheidsfactor van die uitspraak $z(a)$. Hierbij geldt $z(a) = -z(\neg a)$.

Term	zekerheidsfactor
zeker niet	-1,0
bijna zeker niet	-0,8
waarschijnlijk niet	-0,6
misschien niet	-0,2 tot +0,2
misschien	+0,4
waarschijnlijk	+0,6
bijna zeker	+0,8
zeker	+1,0

- Hoe berekenen van de zekerheid van een conclusie als de premisse van een regel zelf onzeker is?

als b
dan a (x)

$$z(a) = z(b) \times x$$

als b **en** c **en** d
dan a (x)

$$z(b \textbf{ en } c \textbf{ en } d) = \min(z(b), z(c), z(d)) \times x$$

als e
dan a (x)

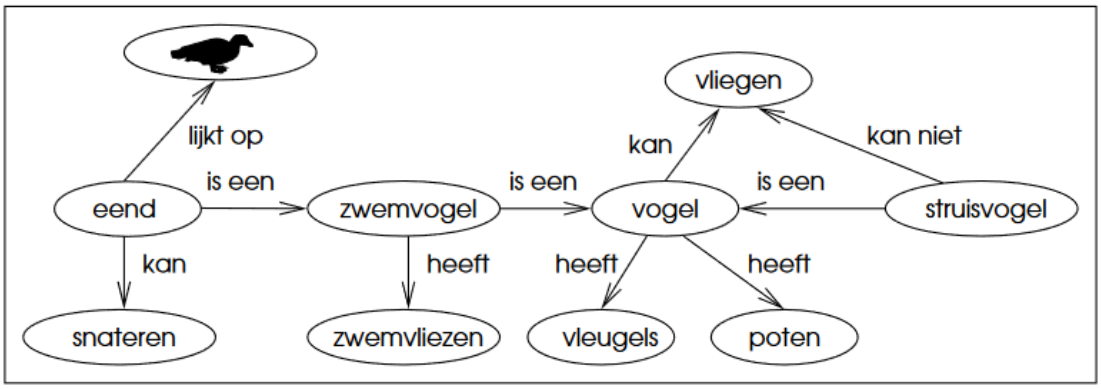
als f
dan a (y)

$$z_1 = z(e) \times x, \qquad z_2 = z(f) \times y$$

$$z(a) = \frac{z_1 + z_2}{1 - \min(|z_1|, |z_2|)}$$

3.4 Frames en regelschema's

- Noodzaak om kennis in te delen.
- Een **frame** kan vergeleken worden met een klasse uit object georiënteerde programeertalen.
- Komt uit de theorie van **semantische netten**.
  - Een semantisch netwerk bestaat uit een gelabelde gerichte multigraaf.
  - Elke knoop is een begrip, en elk begrip is verbonden met andere begrippen.



Figuur 3.1: Een semantisch net.

Hoofdstuk 4

De regel van Bayes en Markovketens

4.1 Probabiliteit

- Een universum U met een eindig aantal toestanden.
- De kans op een uitspraak a wordt genoteerd als $p(a)$ en is gelijk aan:

$$p(a) = \frac{\#(a)}{\#U}$$

met $\#U$ het aantal toestanden in het universum en $\#(a)$ het aantal toestanden waarin a waar is binnen dit universum.

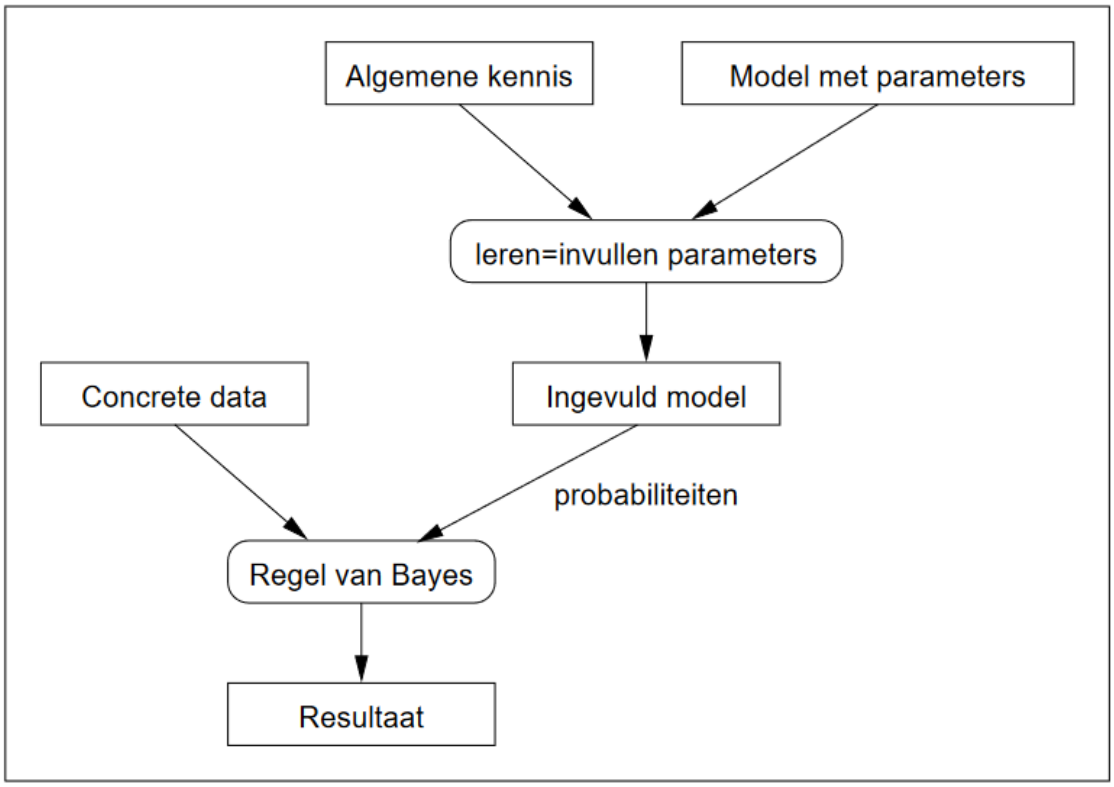
! Niet bruikbaar: het systeem kan enkel de kans bepalen van reeds bestaande gevallen.

1. ◇ Stel 83-jarige patiënt met hoofdpijn.
   ◇ In het universum zit er een 83-jarige patiënt met hoofdpijn.
   ◇ De patiënt kan dus niet bestaan.
2. ◇ Stel 43-jarige patiënt die niet rookt en metselaar is.
   ◇ In het universum zit één metselaar die 43 jaar is, niet rookte en psoriasis heeft.
   ◇ De patiënt heeft dus psoriasis.

- Er is een **zinvolle veralgemening** nodig.
- Het lerend programma moet een **model** gebruiken waarbij een aantal parameters moet worden ingevuld.
- De waarden van de parameters van dit model kan een zinnige veralgemening geven van probailiteit, via **de regel van Bayes**.

4.2 De regel van Bayes

- De voorwaardelijke kans $p(a|b)$ is de kans dat a waar is gegeven dat b waar is.
- Stel $\#U$ het aantal toestanden in het universum en $\#(a)$ het aantal toestanden waarin a waar is:



Figuur 4.1: De regel van Bayes bij artificiële intelligentie.

$$p(a) = \frac{\#(a)}{\#U}, \qquad p(a|b) = \frac{p(a\&b)}{p(b)} = \frac{\#(a\&b)}{\#(b)}$$

- Dit kan herschreven worden:

$$p(b)p(a|b) = p(a\&b)$$

zodat

$$p(a\&b) = p(b\&a)$$

of

$$p(b)p(a|b) = p(a)p(b|a)$$

- Voorbeeld:
  - Universum = 10.000 vogels
  - 1.000 van deze vogels zijn raven: $p(raaf) = 0,1$
  - 2 van deze vogels zijn wit: $p(wit) = 0,0002$
  - Er is slechts 1 witte raaf: $p(raaf|wit) = 0,5$ en $p(wit|raaf) = 0,001$
  - **Dus:** $p(wit)p(raaf|wit) = p(raaf)p(wit|raaf) = 0,0001$
- Dit kan veralgemeend worden met **de regel van Bayes**:

$$p(a|b) = \frac{p(a)p(b|a)}{p(b)}$$

- Veronderstel hypothetische uitspraken $h_i, i = 1, 2, ...$
  - Voor een willekeurig item is exact één hypothetische uitspraak waar.
  - Het resultaat van een aantal metingen op dat item resulteren in een uitspraak d.
  - Wat is de kans dat een hypothese h_i waar is gegeven de data d?

$$p(h_i|d) = \frac{p(h_i)p(d|h_i)}{p(d)}$$

- Voorbeeld:
  - Er is een groot aantal ziektes.
  - Deze leiden tot de lijst van hypothesen, waarbij h_i staat voor de uitspraak "De patiënt heeft de $i-$de ziektepatroon".
  - Is er een groot aantal mogelijke symptomen.
  - Bij een patiënt kunnen de symptomen vastgesteld worden, die resulteren in een uitspraak d.
  - Hoe bepalen welk ziektepatroon het meest waarschijnlijk is?
    - $p(h_i)$ is de kans dat ziektepatroon i voorkomt in het universum. Dit is verondersteld gekend te zijn.
    - $p(d|h_i)$ is de kans dat ziektepatroon i de opgegeven symptomen veroorzaakt. Dit is verondersteld gekend te zijn.
    - $p(d)$ kan berekend worden uit de vorige gegevens, als er verondersteld wordt dat er bij elke patiënt juist één i is waarvoor h_i waar is:

$$p(d) = \sum_i p(h_i \& d) = \sum_i p(h_i)p(d|h_i)$$

    - De expliciete waarde van $p(d)$ is eigenlijk niet nodig.

4.3 Markovketens en -modellen

- Wordt gebruikt om tijdsafhankelijke processen te modelleren.
- Formele definitie van een **Markov model**:
  1. Een eindige verzameling van **staten** $S = \{s_1, ..., s_n\}$. Er is een beginstaat (s_1) en een eindstaat (s_n).
  2. Een **transitiematrix** T. Dit is een $n \times n$ matrix die de probabiliteit van een toestand s_i naar een andere toestand s_j bevat.
  3. Een **uitvoeralfabet** $A = \{a_1, ..., a_{k-1}\} \cup \{a_k\}$. Hier is a_k het lege symbool.
  4. Een **uitvoermatrix** U. Een $a \times k$ matrix.
- Een Markovketen kent een probabiliteit toe aan elke string van karakters in A.
- Een Markovketen produceert een string, maar niet noodzakelijk altijd dezelfde. Het is **niet-deterministisch**, maar de kans op een bepaalde string kan wel berekent worden.
- Op elk ogenblik is de Markovketen in een bepaalde staat. Voor $t = 0$ is dit s_1.
- De keten activeren levert een functie $i : \mathcal{N} \to \{1, ..., n\}$ op die voor elk moment t de index van de staat op dat moment geeft. Op een ogenblik t is de staat $s_{i(t)}$.

- De probabiliteiten om van één staat naar een andere te gaan worden in T beschreven:

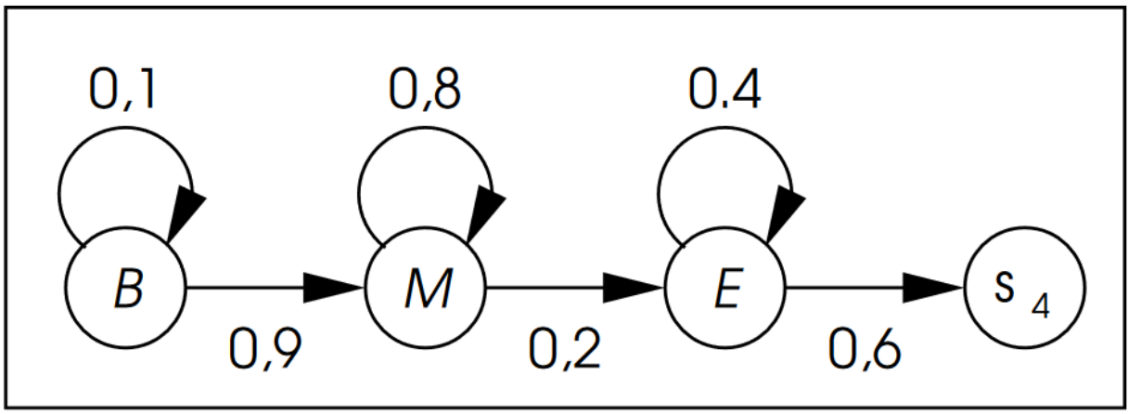
$$T_{kj} = p(i(t+1) = j | i(t) = k)$$

  Voor een willekeurige k geldt (som der kansen):

$$\sum_j T_{kj} = 1$$

- De probabiliteit is onafhankelijk van de tijd t.
- De probabiliteit om op tijdstip $t+1$ een bepaalde staat te bereiken hangt alleen af van de staat op tijdstip t.
- De uitvoer van een Markovketen wordt geschreven als $a_{j(t)}$.
- Als op tijdstip t de staat s_i is, dan is de waarschijnlijkheid dat a_j de uitvoer is gelijk aan U_{ij}.
- Er bestaat dus een één-op-één relatie tussen staat en uitvoer. Hieruit volgt $k = n$, $U_{ii} = 1$ en $U_{ij} = 0$ als $i \neq j$.
- Bij een **Hidden Markov Model (HMM)** kan een staat verschillende mogelijke uitvoeren hebben, en kan een uitvoer horen bij verschillende staten.
- Voorbeeld: Een HMM toegepast op spraakherkenning.
  - Er bestaat een HMM op drie niveaus:
    1. Voor elk *foneem*. Dit zijn betekenisvolle klankenreeksen die soms met een letter overeenkomen en soms met een lettergroep.
    2. Voor elk *woord*. Een woord is een opeenvolging van fonemen.
    3. VOor alle mogelijke zinnen samen.
  - Een aantal veronderstellingen:
    - Sommige woorden kunnen uitgesproken worden met verschillende fonemen. Hier worden deze als verschillende woorden beschouwd.
    - Homoniemen worden als hetzelfde woord beschouwd: meid, mijd, mijdt, mijt, ...
    - Hetzelfde foneem kan traag of snel uitgesproken worden. Deze variatie van lengte wordt aangegeven als de probabiliteit om in dezelfde staat te blijven.
  - Het meest gebruikelijke model maakt gebruik van drie staten: B, M en E (Begin, Midden en Eind), gevolgd door een eindstaat s_4. Figuur 4.2 toont het HMM voor het foneem 't'.
  - Het HMM voor een woord wordt gevormd door de HMM's voor zijn fonemen aaneen te schakelen. Als een woord w_i de fonemen f_i heeft, dan heeft het HMM 3 f_i staten
  - Er moet ook nog een model opgebouwd worden om de waarschijnlijkheid van zinnen te modelleren. Dit is echter zeer complex en hierbij wordt er gebruik gemaakt van ***n*-grammodellen**.
    - Een n-gram is een sequentie van n woorden.
    - Als de probabiliteit van alle n-grammen gekent is, dan kan dit gebruikt worden om bij $n-1$ woorden het volgende woord te voorspellen.
    - Een **unigram** is een 1-gram dat dan de relatieve frequentie van alle woorden bevat.
    - Een **bigram** is een 2-gram bevat alle mogelijke combinaties van twee opeenvolgende woorden.

4.4 Leren van het model



Figuur 4.2: Het HMM voor het foneem 't'.

Hoofdstuk 5

Actieve Systemen

- Klassieke regelbanken worden gebruikt voor expertsystemen, die toelaten om conclusies te trekken uit bepaalde gegevens.
- Er kan ook een systeem ontwikkeld worden waar het geheel

5.1 Eenvoudige systemen

- Bij een eenvoudig systeem is het mogelijk een opsomming te maken van alle visies, acties en hun combinaties.
- Bij elke tijdsstap analyseert de actieve eenheid de visie en kiest op basis daarvan een actie. De buitenwereld reageert op deze actie en genereert een nieuwe visie.

! Een visie geeft slechts gedeeltelijke beschrijving van de wereld. Een **Hidden Markov Decision Model (HMDM)**, wat een uitbreiding is op een **HMM**, tracht dit te modelleren en bestaat uit het volgende:

  - Een eindige verzameling van **staten** $S = \{s_1, ..., s_n\}$. Er is een beginstaat (s_1) en een eindstaat (s_n).
  - Een eindige verzameling $D = \{d_1, ..., d_l\}$ van **beslissingen**.
  - Voor elke $d \in D$ een **transitiematrix** $T(d)$.
  - Een **uitvoeralfabet** $A = \{a_1, ..., a_k\}$. Dit is het alfabet van visies.
  - Een **beloningsfunctie** $b : A \to \mathcal{R}$. Een beloning kan negatief zijn.
  - Een $n \times k$ **uitvoermatrix** U.

- Een **run** is het hele proces van begintoestand en eindtoestand.
- Een actieve eenheid moet een **strategie** τ hebben. Aan elke letter a van het uitvoeralfabet is er dan een beslissing $\tau(a)$.
- Actieve eenheden zonder onzekerheid
  - De staat komt overeen met de visie. $\tau(s) = d$.
  -

5.2 Complexe systemen

5.3 Genetische algoritmen

5.4 Optimalisatie van één strategie

5.5 Combinaties