

Advanced Programming & Maths

Tuur Vanhoutte

1 maart 2021

Inhoudsopgave

1 Basisfuncties in de wiskunde	1
1.1 Functies	1
1.2 Veelterm en veeltermfuncties	2
1.3 Bijzondere veeltermfuncties	2
1.3.1 Constante functie	3
1.3.2 Lineaire functie	3
1.3.3 Tweedegraadsfunctie	3
1.3.4 Derdegraadsfunctie	5
1.3.5 Exponentiële functie	5
2 Exponentiële verbanden in data	6
2.1 Lineaire groei	6
2.2 Exponentiële groei	7
2.3 Van groeipercentage naar groeifactor	7
2.3.1 Percentage naar factor	7
2.3.2 Factor naar percentage	8
2.4 Voorbeeld	8
2.5 Belangrijke maten voor exponentiële toename	9
2.5.1 Oefening: Combinatie van groeifactoren?	9
3 Belangrijke functies met betrekking tot machine learning	10
3.1 Logistische groei	10
3.1.1 Voorbeeld	10
3.1.2 De groei	10
3.1.3 Functievoorschrift	10
3.1.4 Voorbeeld	11
3.1.5 Algemene wiskundige notatie van een logistische functie	11
3.2 Regression analysis	12
3.2.1 Lineair regressiemodel	12
3.2.2 Logistisch regressiemodel	13
3.2.3 Lineair vs logistisch regressiemodel	14
3.2.4 Meerdere inputfactoren	14
3.3 Softmax functie	14
3.3.1 Kansen	15
3.3.2 Model	15
3.3.3 Wiskundig	16
3.4 Logistic regression cost function	16
3.4.1 Success meten	16
4 Pandas library	17
4.1 Inleiding	17
4.1.1 Welke data verwerken?	18
4.2 Pandas.core	18
4.3 Series	18
4.4 DataFrame	18
4.4.1 Select data from DataFrame	19
4.4.2 Veelgebruikte commandos bij dataframes	20
4.5 Loc vs iloc	21
4.5.1 iloc	21
4.5.2 loc	21

4.6	Plotten met pandas	22
4.6.1	Dataframe plotten	22
4.6.2	Series plotten	23
4.7	Demo: Iris Dataset	23
4.8	Complexe bewerkingen	25
5	Normaaldistributie	25
5.1	Doelstelling	25
5.2	Inleiding	25
5.3	Basisbegrippen	25
5.3.1	Kenmerken	26
5.3.2	Verdeling	27
5.3.3	Gevolg: Kansberekening	27
5.3.4	Wiskundig	27
5.4	Standaard normaalverdeling	28
5.4.1	Standardizeren van een normaalverdeling	28
5.5	Kansberekening	29
5.5.1	Via tabellen	30
5.6	Z-score	30
5.6.1	Wiskundig	30
5.6.2	Voorbeeld	30
5.6.3	Nut	31
5.7	Scheefheid	31
5.7.1	Berekening	31
5.8	Kurtosis	32
5.8.1	Wiskundig	32
5.9	Anomaly detection	32
5.9.1	Detectiemethode 1	33
5.9.2	Detectiemethode 2	33
5.9.3	Detectiemethode 3	34
6	Kansrekening	34
6.1	Basiskennis	34
6.1.1	Voorbeeld	34
6.2	Kansdefinitie van Laplace	35
6.2.1	Wiskundige notatie	35
6.2.2	Gecombineerd experiment	35
6.3	Somregel	36
6.3.1	Wiskundig	37
6.3.2	Algemene somregel	37
6.4	Complementregel	37
6.5	Samengesteld experiment (stochastisch exp)	38
6.5.1	Trekking met teruglegging	38
6.5.2	Trekking zonder teruglegging	39
6.6	Permutaties	39
6.6.1	Voorbeeld:	39
6.6.2	Oefening	40
6.7	Combinaties	40
6.7.1	Wiskundig	40
6.7.2	Voorbeeld	40
6.7.3	Oefening	40
6.8	Theoretische en empirische kansen	40

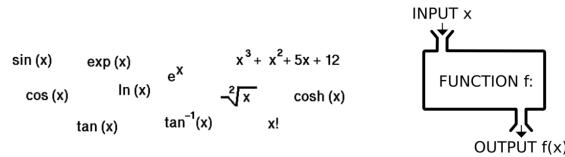
6.8.1	Theoretische kansen:	40
6.8.2	Empirische kansen	41
6.8.3	Voorbeeld	41
6.8.4	Voorbeeld 2	41
7	Voorwaardelijke kans	41
7.1	Doelstelling	41
7.2	Situering	42
7.3	Voorbeeld	42
7.4	Wiskundig	43
7.4.1	Gevolg: productregel	43
7.4.2	Algemene productregel	43
7.5	Oefening 1	44
7.5.1	Opgave	44
7.5.2	Oplossing	44
7.6	Oefening 2	44
7.6.1	Opgave	44
7.6.2	Oplossing	44
7.7	Onafhankelijke gebeurtenissen	44
7.7.1	Voorbeeld	44
7.7.2	Gevolg: Productregel voor onafhankelijke gebeurtenissen	45
7.8	Productregel algemeen	45
7.8.1	Oefening 1	46
7.8.2	Oefening 2	46
7.9	Regel van Bayes	46
7.9.1	Toepassingsdomein	46
7.9.2	Formule van Bayes (eenvoudige vorm)	46
7.9.3	Voorbeeld	47
7.10	Regel van Bayes (variant 1)	47
7.11	Regel van Bayes (variant 2)	48
7.12	Regel van Bayes (algemeen)	48
7.13	Bayesiaans leren: toepassing in machine learning	49
8	Logging	50
8.1	Inleiding	50
8.1.1	Waarom?	50
8.1.2	Opzet	50
8.1.3	Logging levels	50
8.2	Basis logging	50
8.2.1	Spelregels	50
8.2.2	Format logbericht	51
8.2.3	Naar een file loggen	51
8.2.4	Toegepast	51
9	Threading	52
9.1	Doelstelling	52
9.2	Wat zijn threads?	52
9.3	Process vs Thread	52
9.3.1	Process	53
9.3.2	Thread	53
9.4	De threading library	54
9.4.1	Thread aanmaken	54

9.4.2	Andere methods	54
9.4.3	Timer	55
9.5	Multithreading	55
9.5.1	Python Thread Management	55
9.6	Thread toestanden	56
9.7	Opgelet met threads!	56
9.8	Wanneer threads gebruiken?	56
9.9	Thread-class	56
9.9.1	Maken van een nieuwe thread-klasse	56
9.9.2	Opstarten van een thread	57
9.10	Threading-synchronisation	57
9.10.1	Join-methode	57
9.10.2	Thread locking	57
9.10.3	Condition	58
9.11	Daemon vs not-daemon threads	59
9.12	Multithreaded queue	59
9.12.1	Queue	59
9.12.2	Threading + queue	60
9.12.3	Queue programming	61

1 Basisfuncties in de wiskunde

1.1 Functies

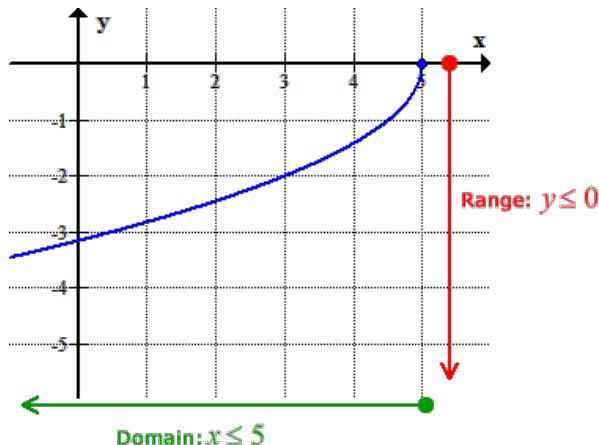
Definitie 1.1 (Reële functie) Een reële functie is een relatie in \mathbb{R} waarbij elke waarde x hoogstens één beeldwaarde $f(x)$ heeft



Figuur 1: Voorbeelden reële functies

Definitie 1.2 Voor elke functie geldt: er bestaat een ...

- (i) ... domein van de functie (domain)
- (ii) ... beeld van de functie (range)
- (iii) ... functievoorschrift van de functie

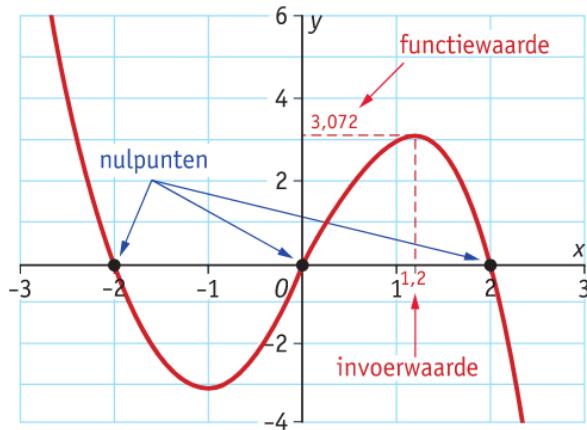


Figuur 2: Domein, bereik, functievoorschrift

$$f : \text{domein} \rightarrow \text{bereik} : x \rightarrow y = f(x)$$

$$f : \mathbb{R} \rightarrow \mathbb{R} : x \rightarrow y = x^3 - 4x$$

Definitie 1.3 Elke functie kan nulpunten hebben.



Figuur 3: $y = -x^3 + 4x$

Verloop van een functie wordt via een tekenschema verduidelijkt:

x		-2		0		2	
$f(x)$	+	0	-	0	+	0	-

Figuur 4: Tekenschema

1.2 Veelterm en veeltermfuncties

Definitie 1.4 (Veelterm)

$$A(x) = a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \dots + a_2 x^2 + a_1 x + a_0 \quad (a_n, a_{n-1}, \dots, a_2, a_1, a_0 \in \mathbb{R}) \quad (1)$$

Definitie 1.5 (Veeltermfunctie)

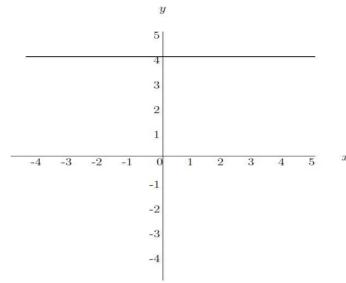
$$f(x) = a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \dots + a_2 x^2 + a_1 x + a_0 \quad (2)$$

Graad van veelterm = n als $a_n \neq 0$

1.3 Bijzondere veeltermfuncties

- Constante functie: $f(x) = 4$
- Lineaire functie: $f(x) = 4$
- Tweedegraadsfunctie: $f(x) = 3x^2 + 2x + 1$
- Derdegraadsfunctie: $f(x) = 5x^3 - 3x^2 + 2x - 1$
- Exponentiële functie: $f(x) = 2^x$
- Logaritmische functie: $(fx) = \log_2(x)$

1.3.1 Constante functie



Figuur 5: $y = 4$

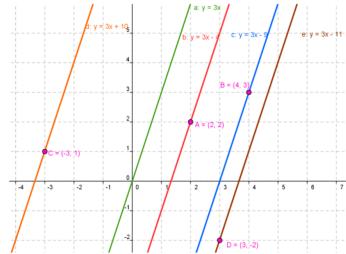
1.3.2 Lineaire functie

Definitie 1.6 (Lineaire functie)

$$f(x) = ax + b \quad (3)$$

Voorbeeld: $f(x) = 3x + 6$

- Betekenis van a : de richtingscoëfficiënt (rico)
- Betekenis van b : het snijpunt met de y -as
- Nulpunt: $f(x) = 0$
 $\Leftrightarrow 3x + 6 = 0$
 $\Leftrightarrow 3x = -6$
 $\Leftrightarrow x = -2$



Figuur 6: Meerdere evenwijdige lineaire functies

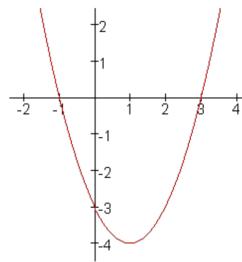
Evenwijdige rechten als: als $a_1 = a_2$

Loodrechte rechten als: als $a_1 \cdot a_2 = -1$

1.3.3 Tweedegraadsfunctie

Definitie 1.7

$$f(x) = ax^2 + bx + c, \quad (a \neq 0) \quad (4)$$



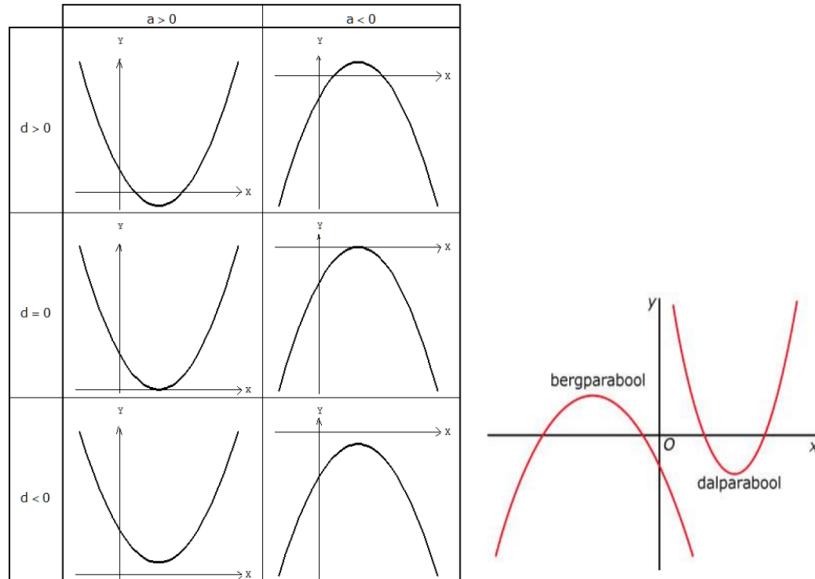
Figuur 7: $f(x) = x^2 - 2x - 3$

- Betekenis van a : positief \Rightarrow dalparabool, negatief \Rightarrow bergparabool
- Nulpunten: via de discriminant berekenen:

Definitie 1.8 (Discriminant) Bij een tweedegraadsvergelijking is de discriminant:

$$D = b^2 - 4ac \quad (5)$$

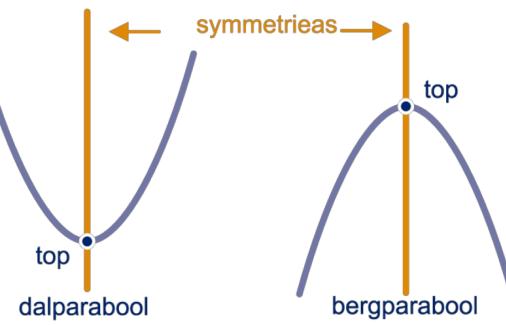
- Geval 1: $D > 0 \Rightarrow$ de functie heeft 2 nulpunten
- Geval 2: $D = 0 \Rightarrow$ de functie heeft 1 nulpunt
- Geval 3: $D < 0 \Rightarrow$ de functie heeft géén nulpunten



Figuur 8: De discriminant toont de nulpunten

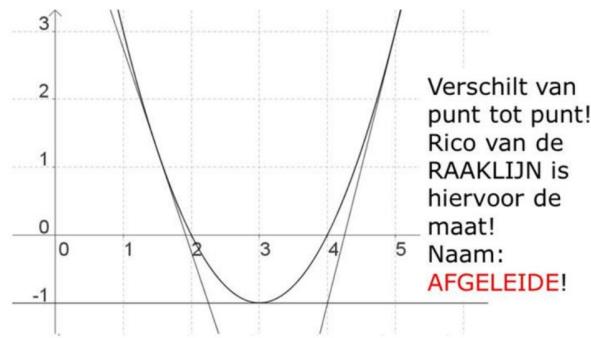
Nulpunten berekenen:

$$x_{1,2} = \frac{-b \pm \sqrt{D}}{2a} \quad (6)$$



Figuur 9: Symmetrieas: $x = \frac{-b}{2a}$

Voorbeeld:



Figuur 10: $y = x^2 - 6x + 8$

1.3.4 Derdegraadsfunctie

Definitie 1.9 (Derdegraadsfunctie)

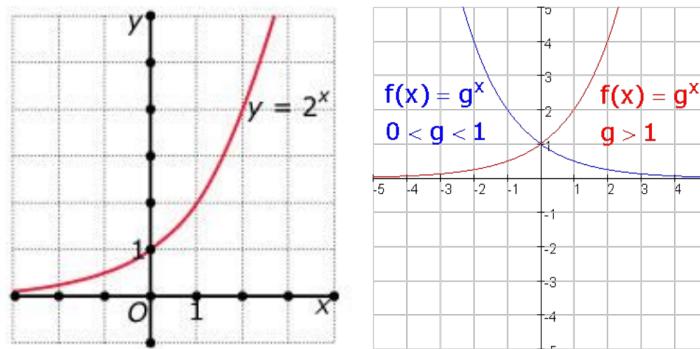
$$f(x) = ax^3 + bx^2 + cx + d (a \neq 0) \quad (7)$$

1.3.5 Exponentiële functie

Definitie 1.10 (Exponentiële functie)

$$f(x) = a^{g(x)} \quad (8)$$

Met grondtal $a \in \mathbb{R}_0^+ \setminus \{1\}$



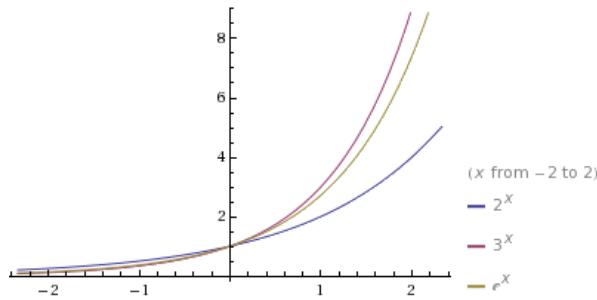
Figuur 11

- Betekenis van a : groefactor
- Wanneer stijgend?
- Wanneer dalend?
- Nulpunten:
- Vaststelling beeld functie

Definitie 1.11 (Constante van Euler)

$$e \approx 2.718281828 \dots \quad (9)$$

$f(x) = e^x$ is een bijzondere exponentiële functie



Figuur 12: Verschil tussen 2^x , 3^x en e^x

2 Exponentiële verbanden in data

2.1 Lineaire groei

Kenmerkend:

- Per tijdseenheid wordt hetzelfde getal **opgeteld**
- Grafiek is een rechte
- Algemene formule (N = aantal, t = tijd, b : beginhoeveelheid):

$$N = a \cdot t + b \quad (10)$$

t	0	1	2	3	4	5
N	750	780	810	840	870	900
	+30	+30	+30	+30	+30	+30

Figuur 13: Lineaire groei

2.2 Exponentiële groei

Kenmerkend:

- Per tijdseenheid wordt de hoeveelheid met hetzelfde getal **vermenigvuldigd**
- Grafiek is een exponentiële functie
- **Algemene formule:**

$$N = b \cdot g^t \quad (11)$$

t	0	1	2	3	4
N	1280	1600	2000	2500	3125
	x1,25	x1,25	x1,25	x1,25	x1,25

Figuur 14: Exponentiële groei

LENGTE FIETSPADEN IN NEDERLAND					
jaar	1998	2002	2006	2010	2014
aantal km	17 600	21 500	26 200	32 000	39 000

Figuur 15: Voorbeeld exponentiële groei met groeifactor ≈ 1.22

2.3 Van groeipercentage naar groeifactor

De toename/afname wordt vaak ook procentueel uitgedrukt

- Een jaarlijkse toename van 14.6%
- Een jaarlijkse afname van 14.6%

Definitie 2.1 (Groeifactor) *De groeifactor is de factor die per tijdseenheid wordt vermenigvuldigd met de vorige waarde.*

2.3.1 Percentage naar factor

$$g = \frac{p + 100}{100}\% \quad (12)$$

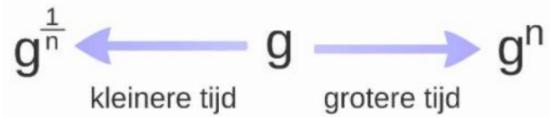
$$\begin{aligned} 100\% + 14,6\% &= 114,6\% = 1,146 \\ \times 1,146 \end{aligned} \qquad \begin{aligned} 100\% - 14,6\% &= 85,4\% = 0,854 \\ \times 0,854 \end{aligned}$$

Figuur 16: Van groeipercentage naar groeifactor

2.3.2 Factor naar percentage

$$0,765 = 76,5\% - 100\% = -23,5\%$$

Figuur 17: Van groeifactor naar groeipercentage



Voorbeeld

De groeifactor per uur is gelijk aan 3.

De groeifactor per 2 uur is gelijk aan $3^2 = 9$

De groeifactor per 20 minuten is gelijk aan $3^{\frac{1}{3}} = 1,44$.

Figuur 18: Let op: hier gebeuren vaak fouten bij het omrekenen

2.4 Voorbeeld

Een hoeveelheid groeit exponentieel. Na 5u is $N = 82$ en na 12u is $N = 246$.

Stel de formule van N op.

Oplossing

$$N = b \cdot g^t \quad (13)$$

Stap 1: groeifactor berekenen per tijdseenheid:

$$\left. \begin{array}{l} \text{Na 5u} \rightarrow N = 82 \\ \text{Na 12u} \rightarrow N = 246 \end{array} \right\} \Delta = 7u \rightarrow 164$$

Groeifactor voor 7 uren: $\frac{246}{82} = 3$

Groeifactor voor 1 uur: $3^{1/7} \approx 1.170$

Stap 2: 1 punt nemen waarvan we N weten:

Gekozen punt: (5, 82)

$$82 = b \cdot (1.170)^5$$

$$\Leftrightarrow b = \frac{82}{1.170}^5 \approx 37$$

$$\Leftrightarrow N = 37 \cdot 1.170^t$$

2.5 Belangrijke maten voor exponentiële toename

Definitie 2.2 (Verdubbelingstijd) *De verdubbelingstijd is de nodige tijd tot de hoeveelheid verdubbeld is.*

De verdubbelingstijd t kan je berekenen met:

$$g^t = 2 \quad (14)$$

Oefening

De populatie neemt toe met 8.3% per jaar. Bereken de verdubbelingstijd:

$$\begin{aligned} g^t &= 2 \\ \Leftrightarrow (1.083)^t &= 2 \\ \Leftrightarrow \log(1.083^t) &= \log(2) \\ \Leftrightarrow t \cdot \log(1.083) &= \log(2) \\ \Leftrightarrow t &= \frac{\log(2)}{\log(1.083)} \\ \Leftrightarrow t &= 8.69 \text{ jaar} \end{aligned}$$

Definitie 2.3 (Halveringstijd) *De halveringstijd is de nodige tijd tot de hoeveelheid gehalveerd is.*

De halveringstijd t kan je berekenen met:

$$g^t = 1/2 \quad (15)$$

2.5.1 Oefening: Combinatie van groeifactoren?

Een hoeveelheid neemt eerst 5 jaar lang met vast percentage (*) toe, om daarna nog 3 jaar met 10% per jaar toe te nemen. Na 8 jaar is de totale hoeveelheid verdubbeld.

(*) Bereken het jaarlijkse groeipercentage in de eerste 5 jaren.

Oplossing

We weten:

- Eerste 5 jaar: toename met vast percentage
- Volgende 3 jaar: toename met 10% (= factor van 1.1)
- Na 8 jaar: hoeveelheid verdubbeld (= factor van 2)

$$g^5 \cdot 1.1^3 = 2$$

We moeten g vinden:

$$\begin{aligned} \Leftrightarrow g^5 &= \frac{2}{1.1^3} \\ \Leftrightarrow g &= \sqrt[5]{\frac{2}{1.1^3}} \end{aligned}$$

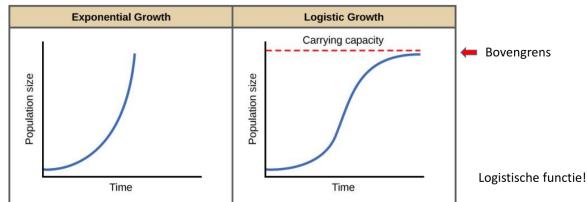
3 Belangrijke functies met betrekking tot machine learning

3.1 Logistische groei

3.1.1 Voorbeeld

Startsituatie: een bos (bv 10km²) waarin een konijnenepidemie uitbreekt. Boswachter houdt de populatie van de konijnen bij. Wat stelt hij vast?

De groei van de populatie verloopt volgens een typisch patroon (niet exponentieel):



Figuur 19: De rode lijn is de bovengrens

3.1.2 De groei

= de mate van toename

- Hangt af van hoeveel er al zijn tegenover hoeveel er nog bij kunnen
- Heel sterke verandering bij start, op het einde heel kleine verandering
- Hangt dus ook af van de tijd

Definitie 3.1 (De logistische groei) *De logistische groei is de mate van toename, afhankelijk van hoeveel er nog bij kan en hoeveel er al is*

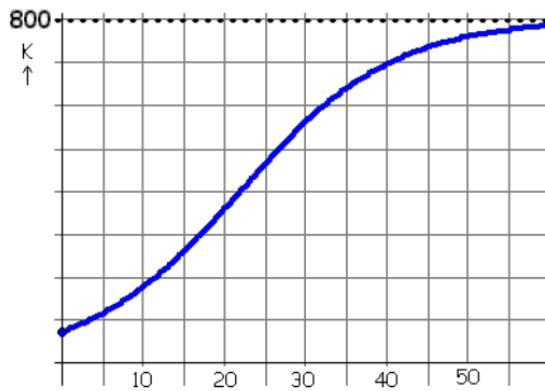
$$\frac{\text{Hoeveel er nog bij kan}}{\text{Hoeveel er al is}} = B \cdot g^t \quad (16)$$

- t = de tijd,
- B en g = constanten

3.1.3 Functievoorschrift

$$y = \frac{G}{1 + B \cdot g^t} \quad (17)$$

- t = de tijd
- B en constanten
- G = bovengrens



Figuur 20: Grafiek logistische groei met $G = 800$

3.1.4 Voorbeeld

Het aantal vissen in een meer is gegeven door:

$$N = \frac{2500}{1 + 5.5 \cdot 0.74^t}$$

waarbij N = aantal vissen, t = tijd

Beredeneer: Wanneer bereiken we het 'verzadigingsniveau'

Als t heel groot is:

- Dan wordt $0.74^t \approx 0$
- Dan wordt $5.5 \cdot 0.74^t \approx 0$
- Dan wordt $N \approx 2500$
- \Rightarrow Het meer is 'verzadigd'

3.1.5 Algemene wiskundige notatie van een logistische functie

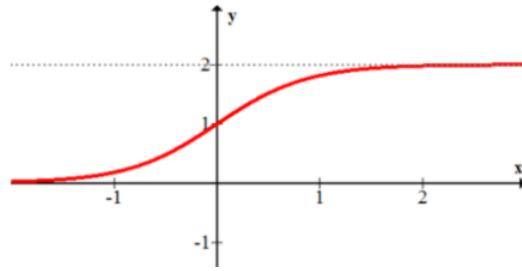
Definitie 3.2 (Logistische functie) De wiskundige notatie voor een logistische functie is:

$$f(x) = \frac{c}{1 + a \cdot b^x} \tag{18}$$

met a, b, c constanten waarbij de constante c de belangrijkste is:

c drukt uit wat de maximumwaarde kan zijn

$$f(x) = \frac{2}{1 + 0.1^x}$$



Figuur 21

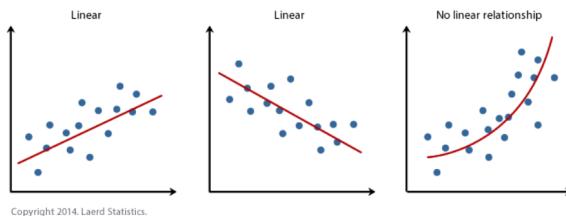
3.2 Regression analysis

Regressieanalyse:

- Is er een (voorspellend) verband tussen 2 variabelen
- Heeft de ene variabele een invloed op de andere variabele



Figuur 22: Regressieanalyse

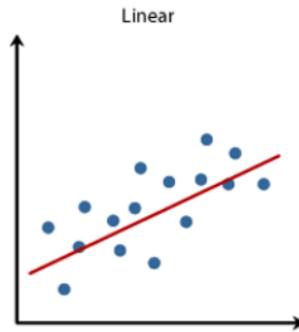


Figuur 23: Lineaire vs niet-lineaire samenhang

3.2.1 Lineair regressiemodel

Enkelvoudige vorm:

- 1 inputwaarde x
- via lineaire functie $h_\theta(x) = \theta_0 + \theta_1 x$



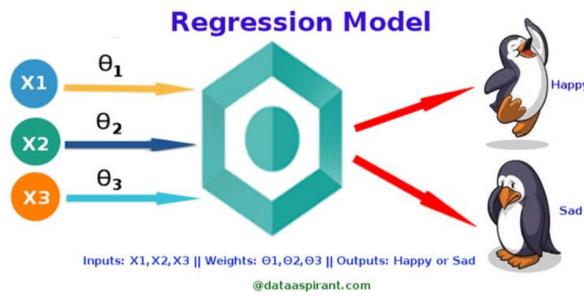
Figuur 24

- Aan de hand van de opgestelde functie doe je een voorspelling
- **Doel:** een zo goed mogelijke lineaire functie opstellen
- ⇒ zoektocht naar de beste θ_0 en θ_1

3.2.2 Logistisch regressiemodel

Logistische regressie = **Classificatie-algoritme**

Zoeken naar een model dat uitkomst (2 mogelijkheden) voorspelt mbv inputwaarden. Elke inputwaarde heeft een zeker belang (gewicht).



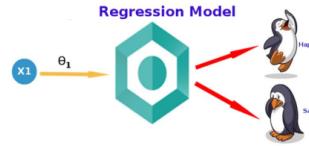
Figuur 25: 3 inputs met elk een bepaald gewicht, die een uitkomst zoekt (2 mogelijkheden)

Vereenvoudiging:

- 1 inputwaarde x
- Logistische functie $p = \frac{1}{1+e^{-(b_0+b_1x)}}$

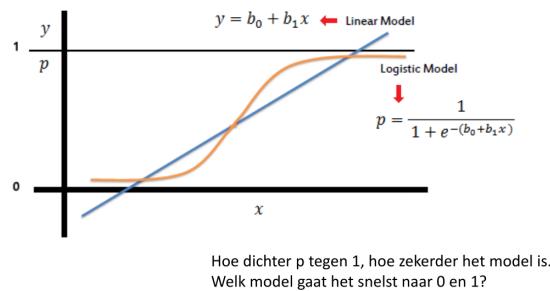
Uitkomst:

- de persoon slaagt als $h_\theta(x) \geq 0.5$
- de persoon slaagt niet als $h_\theta(x) < 0.5$



Figuur 26: 1 inputwaarde x , met twee uitkomsten

3.2.3 Lineair vs logistisch regressiemodel



Figuur 27: Hoe dichter p tegen 1, hoe zekerder het model is

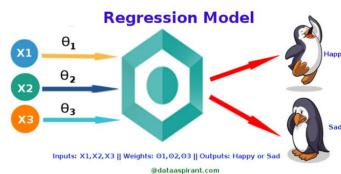
Welk model gaat het snelst naar 0 en 1?

- Het logistische model
- Daarom is het logistische model beter voor classificatie: je splitst de groep op in 2

3.2.4 Meerdere inputfactoren

Zelfde redenering:

- Meerdere inputwaarden x_1, x_2, \dots
- Gebruik $\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots$



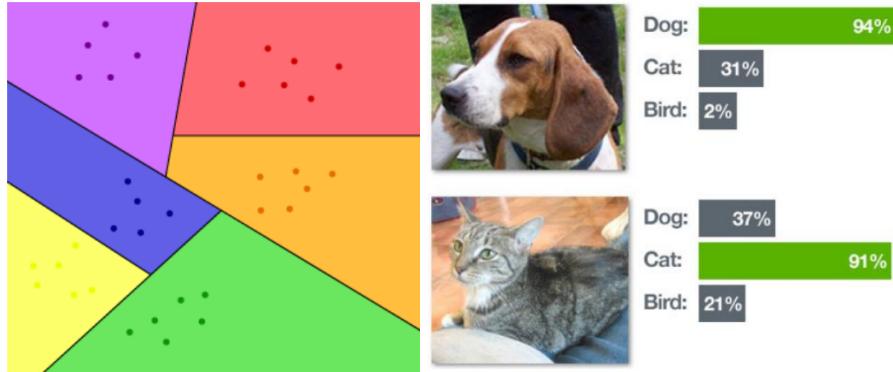
Figuur 28: Regressiemodel met meerdere inputfactoren

3.3 Softmax functie

Doelstelling:

- Model dat in staat is om data te gaan categoriseren
- Hoe?

- ⇒ Met behulp van verschillende inputvariabelen en bijhorende parameters



Figuur 29: Categoriseren met de softmax functie

3.3.1 Kansen

Kans dat de toestand tot groep A behoort:

- $\theta_{A,0} + \theta_{A,1}x_1 + \theta_{A,2}x_2$
- Voorbeeld: $0.01 + 0.1x_1 + 0.1x_2$

Kans dat de toestand tot groep B behoort:

- $\theta_{B,0} + \theta_{B,1}x_1 + \theta_{B,2}x_2$
- Voorbeeld: $0.1 + 0.2x_1 + 0.2x_2$

Kans dat de toestand tot groep C behoort:

- $\theta_{C,0} + \theta_{C,1}x_1 + \theta_{C,2}x_2$
- Voorbeeld: $0.1 + 0.3x_1 + 0.3x_2$

3.3.2 Model

Het softmax-model berekent de mate van zekerheid dat een toestand tot een bepaalde categorie behoort.

vb: volgende quotiënt drukt uit hoe zeker hij is dat (z_1, z_2) tot categorie A behoort:

$$\frac{e^{\theta_{A,0} + \theta_{A,1}z_1 + \theta_{A,2}z_2}}{e^{\theta_{A,0} + \theta_{A,1}z_1 + \theta_{A,2}z_2} + e^{\theta_{B,0} + \theta_{B,1}z_1 + \theta_{B,2}z_2} + e^{\theta_{C,0} + \theta_{C,1}z_1 + \theta_{C,2}z_2}}$$

(analoog voor categorie B en C: vervang de teller)

$$\frac{e^{0.01 + 0.1 \cdot 0.1 + 0.1 \cdot 0.5}}{e^{0.01 + 0.1 \cdot 0.1 + 0.1 \cdot 0.5} + e^{0.1 + 0.2 \cdot 0.1 + 0.2 \cdot 0.5} + e^{0.1 + 0.3 \cdot 0.1 + 0.3 \cdot 0.5}} = 0.2945$$

Figuur 30: Betekenis: het model is 29% zeker dat $(0.1, 0.5)$ tot categorie A behoort. Bereken zelf als oefening voor B en C

3.3.3 Wiskundig

Het gebruikte model wordt via volgende wiskundige formule algemeen beschreven:

$$\frac{e^{x_k}}{\sum_{i=1}^n e^{x_i}} \quad (19)$$

waarbij:

- $x_k = \theta_{k,0} + \theta_{k,1}x_1 + \theta_{k,2}x_2 + \cdots + \theta_{k,m}x_m$
- n = aantal groepen
- m = het aantal meetcriteria

3.4 Logistic regression cost function

Het model:

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^\top x}} \quad (20)$$

waarbij:

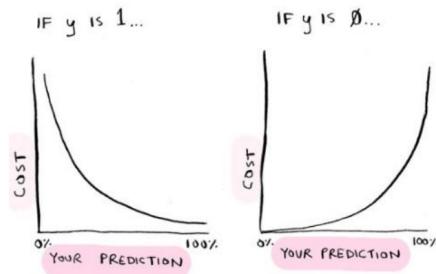
- $\theta^\top x = \theta_0 + \theta_1x_1 + \theta_2x_2$
- h_θ drukt uit wat de kans is dat voor opgegeven x_1 en x_2 de waarneming tot 1 groep behoort
- x_1 en x_2 zijn de inputwaardes
- θ_1 en θ_2 zijn gewichten (hoe belangrijk is de input)
- **Doel:** vinden van de beste gewichten zodat de voorspelling == de werkelijkheid

3.4.1 Success meten

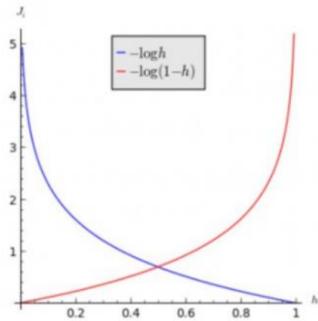
Stel: je maakt een logistisch regressiemodel die bepaalt of een object een groene appel of een tennisbal is.

- Bepalen van de kostenfunctie $J(\theta)$ met als doel deze zo laag mogelijk te brengen
- kost = afwijking tegenover de werkelijke situatie
- werkelijkheid kan 2 situaties zijn:
 - Indien de werkelijkheid een groene appel is $\Rightarrow y = 1$
 - Indien de werkelijkheid géén groene appel is $\Rightarrow y = 0$

Hoe ziet zo'n kostfunctie er dan uit?



Figuur 31: Als $y = 1$ en $y = 0$



Figuur 32

$$\begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Figuur 33

Hoe brengen we 2 mogelijke situaties in 1 functie samen?

(TODO: slide 24 - 32)

4 Pandas library

4.1 Inleiding

- Doelstelling:
 - Nut van de pandas library kunnen situeren
 - Data-analyse: basisbewerkingen
- Pandas = ‘Python Data Analysis Library’
- Pandas bouwt op de NumPy library
- Officiële website: <https://pandas.pydata.org/>
- Goede start: <http://pandas.pydata.org/pandas-docs/stable/10min.html>

4.1.1 Welke data verwerken?

- csv-files
- txt-files
- Excel-files
- Databases

4.2 Pandas.core

Beschikbare datastructuren:

- Series (1D)
- DataFrame (2D)
- Panel (3D)

4.3 Series

Bestemd voor 1-dimensionale data:

'a one-dimension labeled array capable of holding any data'

- Subklasse van numpy-ndarray
- Data: elk soort datatype
- Geordende index
- Duplicaten mag (maar niet optimaal)

	index	values
A	→	5
B	→	6
C	→	12
D	→	-5
E	→	6.7

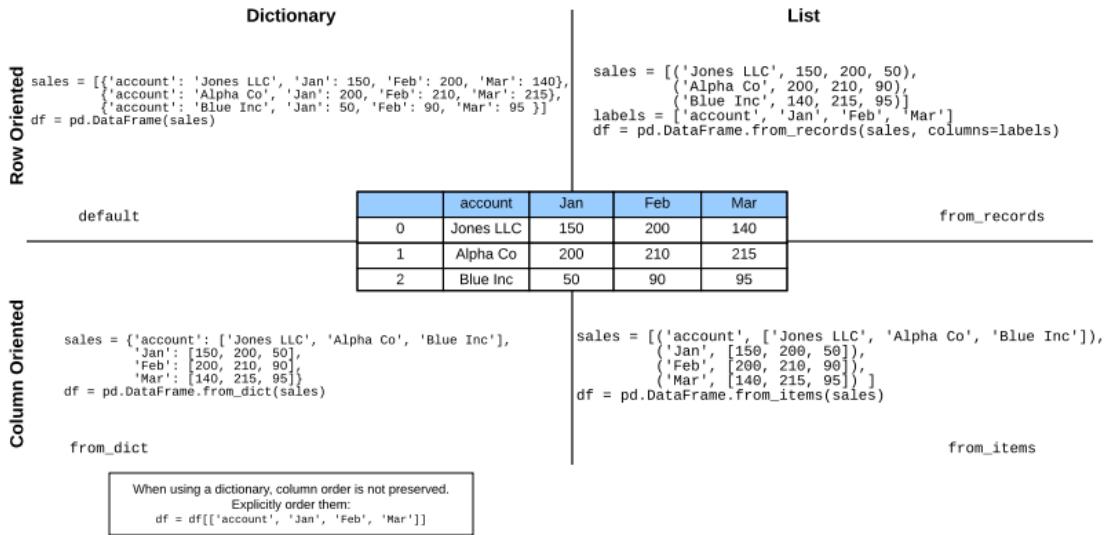
Figuur 34: Elk element heeft een index

4.4 DataFrame

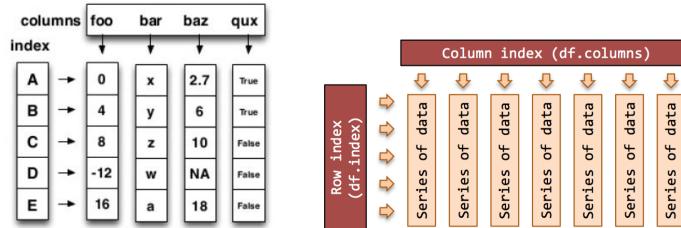
Bestemd voor meer-dimensionale data

- Subklasse van numpy-ndarray
- Elke kolom kan ander datatype hebben
- Rij en kolom index
- Grootte wijzigbaar (invoegen/verwijderen van rijen en kolommen)

Creating Pandas DataFrames from Python Lists and Dictionaries



Figuur 35: DataFrames maken uit Python lists en dictionaries



Figuur 36: Elk element heeft een rij en kolom

4.4.1 Select data from DataFrame

Via operator [] selecteer je een kolom:

	Feb	Jan	Mar	account
0	200	150	140	Jones LLC
1	210	200	215	Alpha Co
2	90	50	95	Blue Inc

Figuur 37: Voorbeeld DataFrame

```

1 # selecteer de kolom Jan uit dataframe
2 df['Jan']
3
4 # analoog: elke kolom is dus een attribuut van dataframe
5 df.Jan
6
7 # returnwaarde: Series-object

```

```

8    0    150
9    1    200
10   2    50
11  Name: Jan, dtype: int64

```

Via operator [] selecteer je een kolom & krijg je een dataframe terug:

```

1  >> df[['Jan']]
2  # returns:
3      Jan
4  0    150
5  1    200
6  2    50
7
8  >> df[['Jan', 'Mar']]
9  # returns:
10     Jan      Mar
11    0    150    140
12    1    200    215
13    2     50     95

```

Via de operator [] en met een conditie:

	Feb	Jan	Mar	account
0	200	150	140	Jones LLC
1	210	200	215	Alpha Co
2	90	50	95	Blue Inc

Figuur 38: Voorbeeld DataFrame

```

1  >> df[df.Jan > 60]
2  # returns:
3      Feb      Jan      Mar      account
4  0    200    150    140    Jones LLC
5  1    210    200    215    Alpha Co
6
7  >> df[np.logical_and(df.Jan > 100, df.Feb <= 200)]
8  # returns:
9      Feb      Jan      Mar      account
10   0    200    150    140    Jones LLC
11
12 >> df[df.account.str.startswith('Alpha')]
13 # test deze eens zelf uit als oefening :)

```

4.4.2 Veelgebruikte commandos bij dataframes

1 df.shape	# geeft de dimensie als een tuple terug
2 df.info()	# oplijsting van de aanwezige kolommen
3 df.head([aantal])	# eerste vijf/aantal rijen
4 df.tail([aantal])	# laatste vijf/aantal rijen
5 df.index	# geef de index-kolom weer

```

6 df.columns      # geef de kolomnamen weer
7 df.describe()  # geef snel overzicht van statistische data
8 df.T           # transponeer data (rij -> kol, kol -> rij)
9 df.sort_index() # sorteert op basis van index
10 df.sort_values() # sorteren op één of meerdere kolommen

```

4.5 Loc vs iloc

4.5.1 iloc

= Integer-location based indexing / selection by position

- Nut: selecteren van rijen en kolommen via rij/kolomnummer
- Syntax: data.iloc[<row>, <column>]
- Returnwaarde:
 - Indien 1 **rij** ⇒ series-object
 - Indien meerdere **rijen**: ⇒ dataframe-object
 - 1 of meerdere **kolommen**: ⇒ dataframe-object

iloc-voorbeelden:

```

1 # Rows:
2 data.iloc[0] # first row of data frame
3 data.iloc[1] # second row of data frame
4 data.iloc[-1] # last row of data frame
5
6 # Columns:
7 data.iloc[:,0] # first column of data frame
8 data.iloc[:,1] # second column of data frame
9 data.iloc[:, -1] # last column of data frame
10
11 data.iloc[0:5] # first five rows of dataframe
12
13 # first two columns of data frame with all rows
14 data.iloc[:, 0:2]
15
16 # 1st, 4th, 7th, 25th row + 1st 6th 7th columns.
17 data.iloc[[0,3,6,24], [0,5,6]]
18
19 # first 5 rows and 5th, 6th, 7th columns of data frame
20 data.iloc[0:5, 5:8]

```

4.5.2 loc

= label based indexing / selection

- Nut: selecteren van rijen en kolommen via label / via conditionele look-up
- Syntax: data.loc[<row>, <column>]
- Returnwaarde:

- Indien 1 rij/kol ⇒ series-object
- Indien meerdere rijen: ⇒ dataframe-object
- 1 of meerdere kolommen: ⇒ dataframe-object

loc-voorbeelden:

	cars_per_cap	country	drives_right
US	809	United States	True
AUS	731	Australia	False
JAP	588	Japan	False
IN	18	India	False
RU	200	Russia	True
MOR	70	Morocco	True
EG	45	Egypt	True

Figuur 39: Voorbeeld dataframe

```

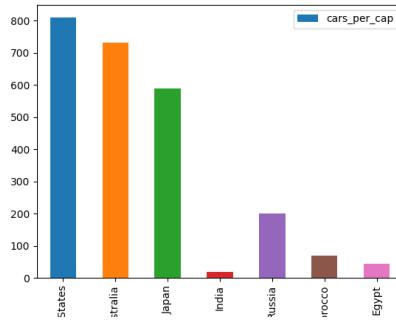
1 reviews.loc[:2, "score"] # return type =
2
3 reviews.loc[:2, ["score", "title"]] # return type =
4
5 # select column "score" where value of index <= 5
6 reviews.loc[:5, "score"]
7
8 # select columns "country" and "cars_per_cap" where rowindex is "US" or "RU"
9 cars.loc[ ["US","RU"] , ["country","cars_per_cap"]]
10
11 # select columns "country" and "cars_per_cap" where rowindex is from "US" to "RU"
12 cars.loc[ "US " : "RU " , ["country","cars_per_cap"]]
13
14 # selectie rijen hoeft niet altijd op basis van row-index te zijn
15 # select columns "country" and "drives_right", voor de landen 'Japan' en 'India'
16 cars.loc[ cars.country.isin( ['Japan', 'India'] ) , ['country','drives_right']]
```

4.6 Plotten met pandas

4.6.1 Dataframe plotten

```

1 # print(cars[['country', 'cars_per_cap']])
2 # werkwijze 1:
3 cars[['country', 'cars_per_cap']].plot(kind='bar', legend=True)
4 # werkwijze 2:
5 cars.plot(x='country', y='cars_per_cap', kind='bar', legend=True)
6
7 plt.show()
```



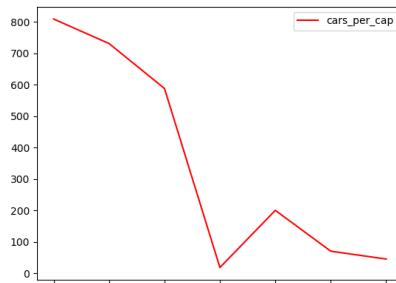
Figuur 40: Resultaat

4.6.2 Series plotten

```

1 # werkwijsje 1:
2 plt.plot(cars['cars_per_cap'])
3 # werkwijsje 2:
4 plt.plot(cars['cars_per_cap'].plot(color='r', legend=True))
5
6 plt.show()

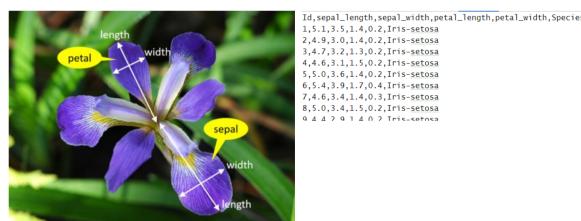
```



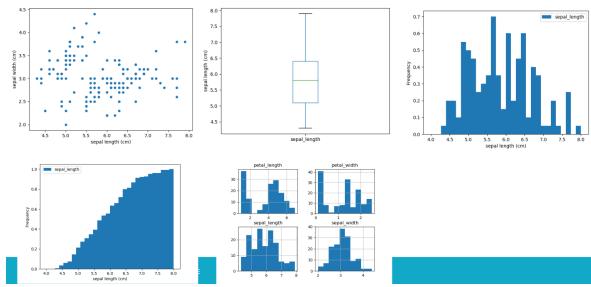
Figuur 41: Resultaat

4.7 Demo: Iris Dataset

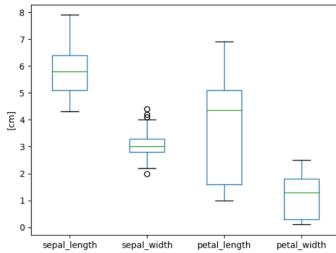
(Zie DemoPandas.zip op Leho voor de code)



Figuur 42: Een iris bestaat uit petals & sepals, met elk hun breedte en lengte



Figuur 43: Plotten van de beschikbare data (demo5.py)

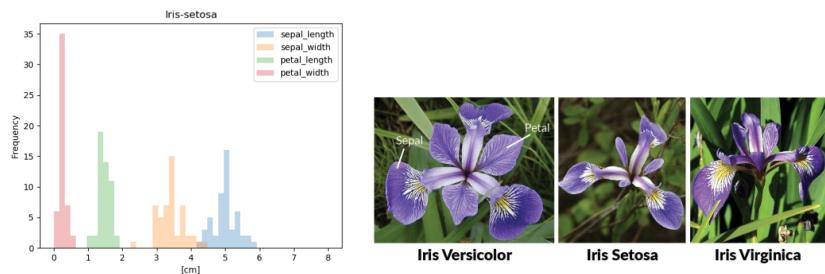


Figuur 44: Vergelijken van de lengtes en breedtes adhv box-plots (demo6.py)

```

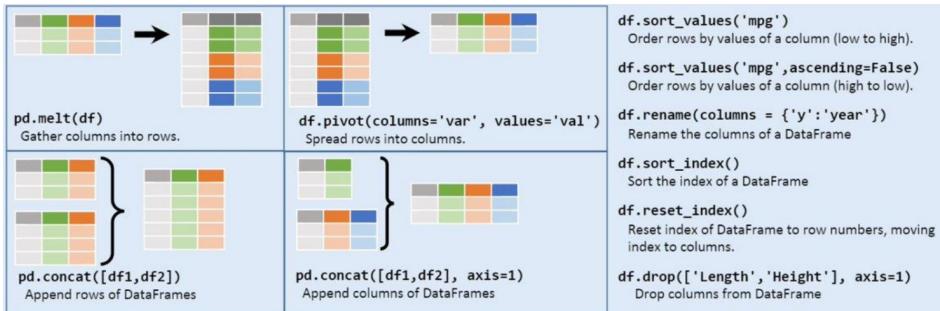
1 # filteren van data
2 result_check = iris['Species'] == 'Iris-setosa'
3 # print(type(result_check)) == TODO
4 filtered_setosa = iris.loc[result_check, :]
5 # of in 1 lijn:
6 filtered_setosa = iris.loc[iris['Species'] == 'Iris-setosa', :]

```



Figuur 45: Frequentiediagram voor de Iris-setosa soort (demo7.py)

4.8 Complexe bewerkingen



Figuur 46: Reshaping data: change the layout of a data set

(komen we later nog op terug)

5 Normaalverdeling

5.1 Doelstelling

- Het herkennen van de eigenschappen van de normaalverdeling
- Verband standaard normaalverdeling en Z-waarden
- Z-index, Z-score tabel

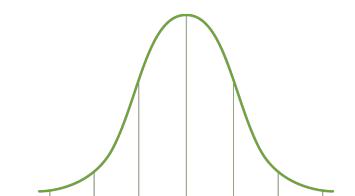
5.2 Inleiding

Wanneer komt een normaalverdeling voor?

- Grote aantallen onafhankelijke waarnemingen uit een willekeurige populatie
 - De lengte van personen
 - Productieprocessen die een bepaalde tijdsduur hebben
 - ...

5.3 Basisbegrippen

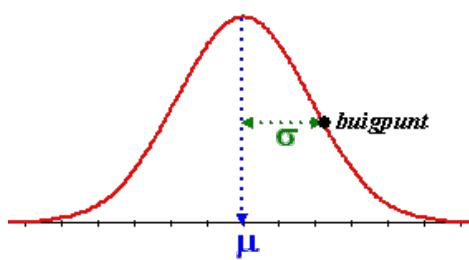
Definitie 5.1 (Normaalverdeling) *Een intervalwaarde dat afhankelijk is van oneindig aantal onafhankelijke factoren (die los van elkaar in werken) zal in de populatie een normaalverdeling vertonen (Gauss-curve)*



Figuur 47: Gauss-curve

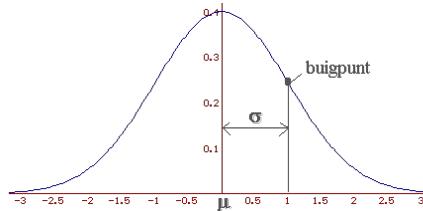
5.3.1 Kenmerken

- Klokvormig verloop
- 1 maximum = gemiddelde = mediaan (μ)
- Uitslagen vooral geconcentreerd rond gemiddelde
- Frequentie daalt naarmate scores afwijken
- Twee buigpunten
- Symmetrie



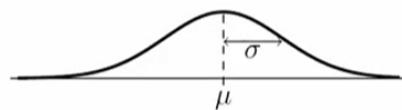
Figuur 48: Kenmerken Gauss-curve

- Standaarddeviatie (standaardafwijking) σ weerspiegelt mate van spreiding
- Wordt gemeten in de buigpunten



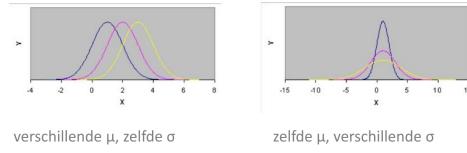
Figuur 49

- Functie is volledig te beschrijven met het gemiddelde μ en de Standaarddeviatie σ



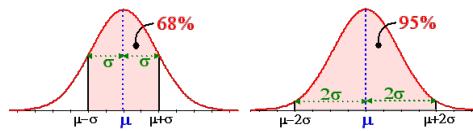
$$X \sim N(\mu, \sigma^2)$$

Figuur 50: Notatiewijze

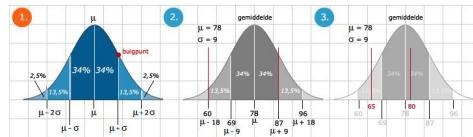


Figuur 51: Impact μ en σ

5.3.2 Verdeling



Figuur 52



Figuur 53

- Tussen beide buigpunten ($\mu - \sigma, \mu + \sigma$): $\approx 68\%$ van de populatie
- Tussen ($\mu - 2\sigma, \mu + 2\sigma$): $\approx 95\%$ van de populatie
- Tussen ($\mu - 3\sigma, \mu + 3\sigma$): $\approx 99\%$ van de populatie

5.3.3 Gevolg: Kansberekening

- Oppervlakte onder de grafiek = 100%
- Hiermee kunnen we nu uitdrukken wat de kans van een specifiek bereik kan zijn. Bv:
 - Met $\mu = 78, \sigma = 9 \Rightarrow$ Kans dat $x < 65$?
 - \Rightarrow bereken oppervlakte links van 65

5.3.4 Wiskundig

$$f(x) = \frac{1}{\sqrt{2 \cdot \pi} \cdot \sigma} \cdot e^{-\frac{(x-\mu)^2}{2 \cdot \sigma^2}} \quad (21)$$

Standaardafwijking berekenen:

$$\sigma = \sqrt{\frac{\sum_{i=1}^n (x_i - \mu)^2}{n-1}} \quad (22)$$

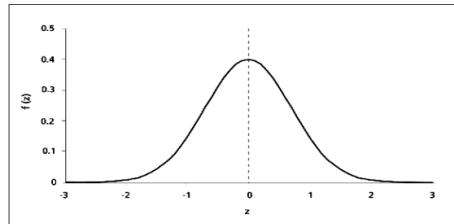
- $\sigma =$ Standaarddeviatie van getallenreeks x

- x_i = de waarde van getal i in de getallenreeks
- μ = het gemiddelde van de getallenreeks
- n = het aantal getallen in de proef

5.4 Standaard normaalverdeling

Definitie 5.2 (Standaard normaalverdeling) De standaardnormaalverdeling is een normaalverdeling waarbij:

- $\mu = 0$
- $\sigma = 1$



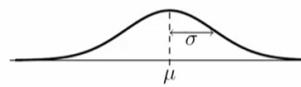
Figuur 54: Oppervlakte onder grafiek = 1

5.4.1 Standardizeren van een normaalverdeling

Doele: een normaalverdeling omzetten in een standaardnormaalverdeling waarbij $\mu = 0$ en $\sigma = 1$



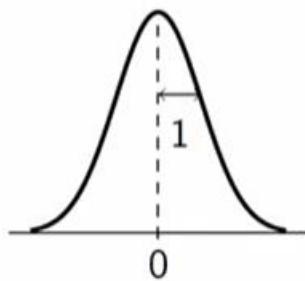
Figuur 55: Standardizeren



$$X \sim N(\mu, \sigma^2)$$

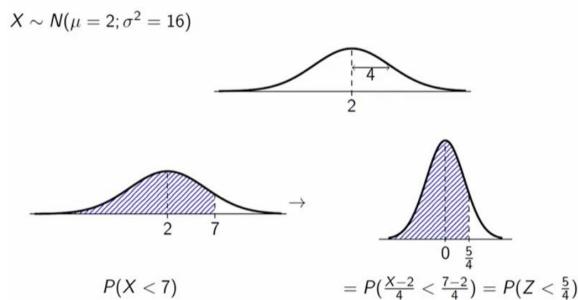
Figuur 56: Uitgangspunt: willekeurige normaalverdeling

- Verschuiven naar 0: $X - \mu$
- samendrukken/uitrekken tot $\sigma = 1$: $\frac{X-\mu}{\sigma}$



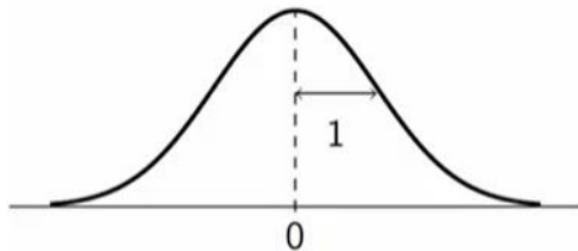
$$Z = \frac{X-\mu}{\sigma} \sim N(0, 1)$$

Figuur 57: Gestandardiseerd: alle eigenschappen blijven bewaard, enkel de hoogte blijft hetzelfde



Figuur 58: Standardizeren: voorbeeld

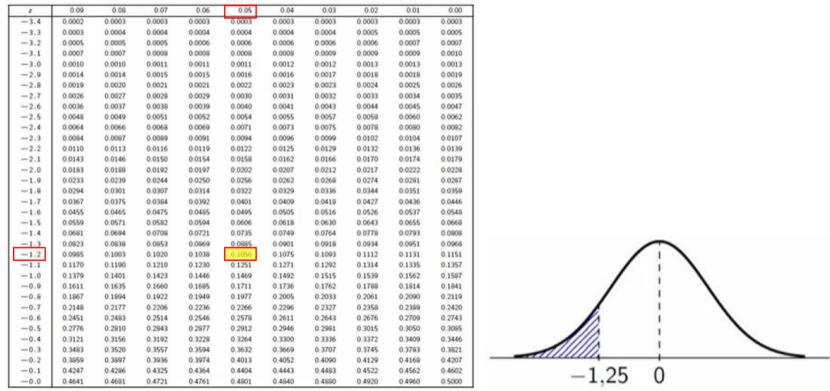
5.5 Kansberekening



Figuur 59: $Z \sim N(0, 1) \rightarrow \mu = 0$ en standaardafwijking $\sigma = 1$

- Grafiek beschrijft hoe waarschijnlijk uitkomsten zijn
- Uitkomsten dicht bij 0 waarschijnlijker dan in de staarten
- Kans = oppervlakte onder de grafiek, totale kans = 100% = 1

5.5.1 Via tabellen



Figuur 60: Kansberekening via tabellen: rijen = 1 getal na de komma, kolommen = 2de getal na de komma. $P(Z < 1.25) = 0.1056$

- Symmetrie: $P(Z < -1.25) = P(Z > 1.25) = 0.1056$
- Complement: $P(Z < -1.25) = 1 - P(Z < 1.25)$

5.6 Z-score

Definitie 5.3 (Z-score) Een Z-score geeft aan hoeveel standaardafwijkingen een observatie van het gemiddelde verwijderd is.

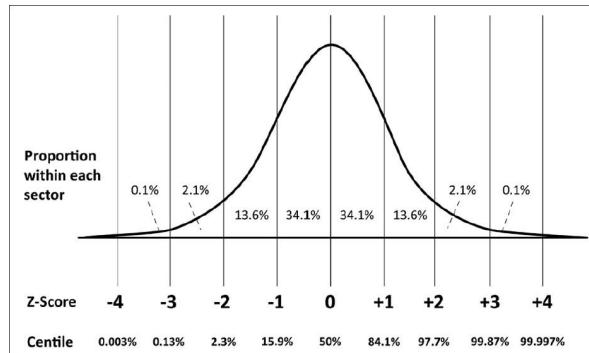
Je krijgt dus je plek ten opzichte van het gemiddelde, uitgedrukt in een standaard maat.

Voordeel: je weet direct hoe goed iemand scoort ten opzichte van de rest.

5.6.1 Wiskundig

$$z = \frac{x - \mu}{\sigma} \quad (23)$$

5.6.2 Voorbeeld



Figuur 61: Hoeveel standaarddeviations (σ) zit een score van het gemiddelde μ

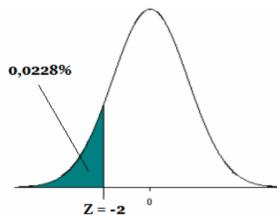
Voorbeeld: 20 studenten leggen een examen af.

- Gemiddelde $\mu = 7/10$
- Standaardafwijking $\sigma = 0.5$ (68% heeft score tussen [6.5, 7.5])
- Iemand heeft 6/10 $\Rightarrow z\text{-score} = -2 \Rightarrow 2 \cdot 0.5$ onder gemiddelde

5.6.3 Nut

Z-score is eveneens een uitdrukking van de kans:

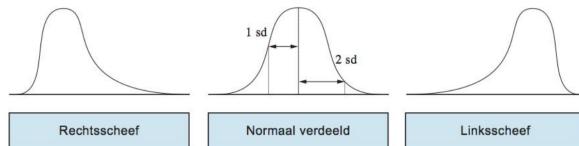
- Hoe (on)gebruikelijk is een score
- Bij vorig voorbeeld: 2.275% heeft een score $\leq 6/10$



Figuur 62

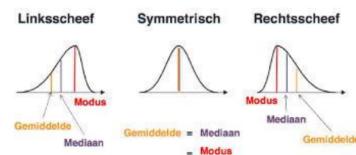
5.7 Scheefheid

Veel frequentieverdelingen hebben niet de vorm van de normaalverdeling. Ze zijn 'scheef'



Figuur 63: Scheefheid

- Linksscheef = gemiddelde is lager dan de mediaan
- Rechtsscheef = gemiddelde is hoger dan de mediaan



Figuur 64

5.7.1 Berekening

De maat voor scheefheid kan berekend worden:

- Linksscheef: negatieve waarde

- Rechtsscheef: positieve waarde

$$Scheefheid = \left(\frac{(\sum(x-\bar{x})^3)}{(\sum(x-\bar{x})^2)^{3/2}} \right) \quad (24)$$

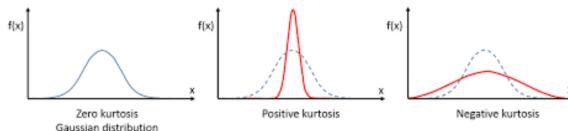
Interpretatie:

- Waarde tussen [-0.5, 0.5]: zeer goede normaalverdeling
- Waarde tussen [-1, 1]: redelijk goede normaalverdeling
- Waarde daarbuiten: geen goede normaalverdeling

5.8 Kurtosis

Definitie 5.4 (Kurtosis) *Kurtosis is de maat van de gepiektheid.*

Je gaat na of de verdeling een scherpe top heeft of een nogal vlakke top.



Figuur 65

5.8.1 Wiskundig

TODO (25)

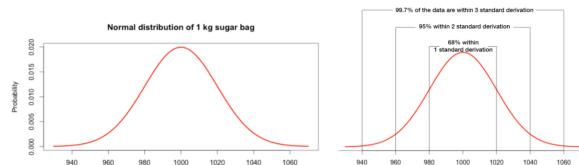
Interpretatie:

- 0 = normaalverdeling
- Minimum is -3: volledig plat
- Maximum is + inf

5.9 Anomaly detection

Voorbeeld: suikerfabriek produceert 1kg suikerzakken

- In werkelijkheid: nooit exact 1kg



Figuur 66: Verdeling 1kg suikerzak

Definitie 5.5 (Anomalie) Een anomalie in een normaalverdeling is een onregelmatigheid of afwijking
Soms verdacht, soms niet

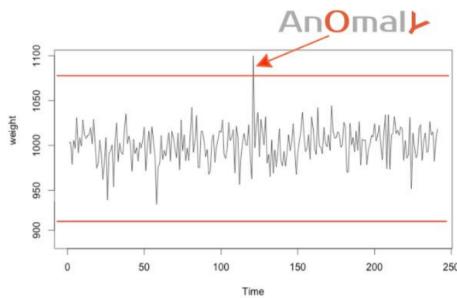
"Anomaly detection (also outlier detection) is the identification of items, events or observations which do not conform to an expected pattern or other items in a dataset."

5.9.1 Detectiemethode 1

= Minimum- en maximumgrens vastleggen

Bv:

- wanneer een waarde kleiner is dan $\mu - 4\sigma$ (920 gram)
- wanneer een waarde groter is dan $\mu + 4\sigma$ (1080 gram)



Figuur 67

5.9.2 Detectiemethode 2

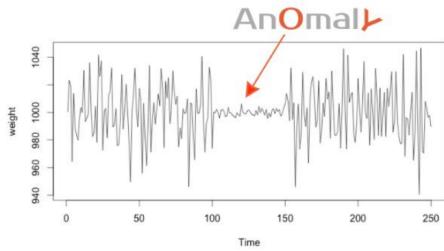
= detecteren van een zeldzame verdeling die afwijkt van de normaalverdeling

Bv:

- 4 opeenvolgende waarden die vallen in de buitengrenzen $\mu - 3\sigma$ en $\mu + 3\sigma$
- Kans voor 1 waarde: 0.3%
- Kans voor 4 opeenvolgende waardes: $(0.3\%)^4 = (0.003)^4 = 0.00000000081$

Bv:

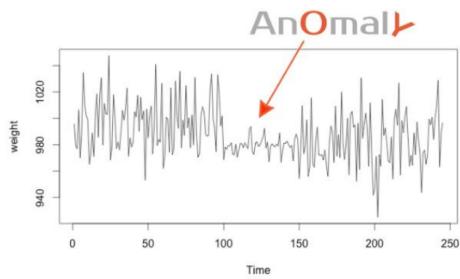
- 50 opeenvolgende waarden die vallen in de buitengrenzen $\mu - \sigma$ en $\mu + \sigma$
- Kans voor 1 waarde: 68%
- Kans 50 opeenvolgende waardes: $(0.68)^{50} = 4.221 \cdot 10^{-9}$



Figuur 68

5.9.3 Detectiemethode 3

Volledige verschuiving van de normaalverdeling



Figuur 69: Oplossing: gebruik van window

6 Kansrekening

Doelstelling:

- Basiskennis kansrekening
- Definitie van Laplace
- Somregel kunnen toepassen
- Wet van de grote getallen
- Wat is een empirische kans?

6.1 Basiskennis

Kansrekenen = waarschijnlijkheidsrekenen

- Vrij jonge tak in de wiskunde
- 'Rekenen over wat er nog niet is'

6.1.1 Voorbeeld

Wat is de kans dat je 6 ogen gooit met een dobbelsteen?

- Kans op gooien van 1 = 1/6

- Kans op gooien van 2 = 1/6
- ...
- Kans op gooien van x = 1/6 = 16.7%

1 = aantal gunstige uitkomsten

6 = aantal mogelijke uitkomsten

6.2 Kansdefinitie van Laplace

Definitie 6.1 (Kansdefinitie van Laplace) Bij een kansexperiment is de kans op een gebeurtenis G gelijk aan:

$$P(G) = \frac{\text{aantal gunstige uitkomsten}}{\text{aantal mogelijke uitkomsten}}$$

6.2.1 Wiskundige notatie

$$P(A) = \frac{\#A}{\#\Omega} \quad (26)$$

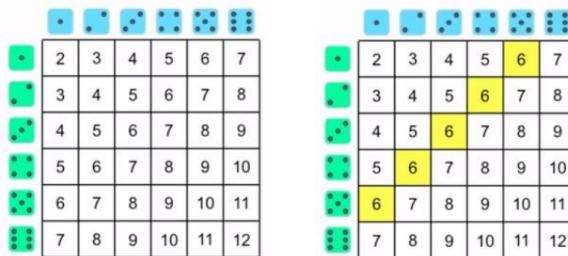
- Met A: een gebeurtenis
- Met Ω : alle mogelijkheden ('het universum') ($\Omega = \{w_1, w_2, \dots, w_n\}$)
- Met $\#A$: aantal elementen (uitkomsten) die voldoen aan deze gebeurtenis
- Met $\#\Omega$: aantal elementen uit universum
- Voorwaarde: alle mogelijke uitkomsten zijn even waarschijnlijk

6.2.2 Gecombineerd experiment

Bv: je gooit 2 dobbelstenen

Wat is de kans op de gebeurtenis 'de som is 6 ogen'?

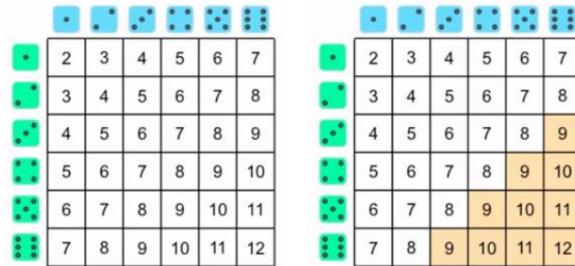
- $P(\text{som} == 2) = \frac{1}{36}$
- $P(\text{som} == 6) = \frac{5}{36}$



Figuur 70: Kans dat de som 6 ogen is

Wat is de kans op de gebeurtenis 'de som is minstens 9 ogen'?

- $P(\text{som} \geq 9) = \frac{10}{36} = \frac{5}{18}$



Figuur 71: Kans dat de som minstens 9 ogen is

Oefening: je gooit met een gewone dobbelsteen en een viervlaksdobbelsteen

- Wat is de kans op de gebeurtenis met beide dobbelstenen evenveel gooien?

4	5	6	7	8	9	10
3	4	5	6	7	8	9
2	3	4	5	6	7	8
1	2	3	4	5	6	7
	1	2	3	4	5	6

Figuur 72: Kansentabel opstellen

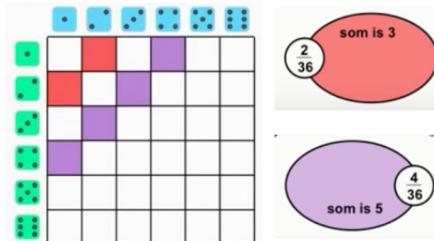
$$- P(\text{beide dobbelstenen evenveel}) = \frac{4}{24} = \frac{1}{6}$$

- Wat is de kans dat de som van de ogen minstens 8 is?
 - Opnieuw kansentabel opstellen $\Rightarrow \frac{15}{36}$

6.3 Somregel

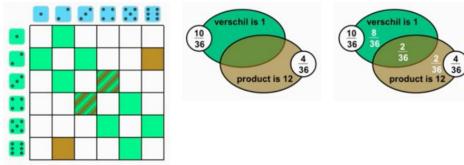
Bv: Je gooit met 2 dobbelstenen

- Wat is de kans op 'som is 3 **of** som is 5'



Figuur 73: Geen overlap:

- Wat is de kans op 'verschil is 1 **of** product is 12'



Figuur 74: Er is overlap: gevaar voor dubbeltelling \Rightarrow verschil aftrekken

6.3.1 Wiskundig

- De gebeurtenis dat A of B beide optreden = $A \cup B$ ‘vereniging’
- De gebeurtenis dat A of B tegelijkertijd optreden = $A \cap B$ ‘doorsnede’

Definitie 6.2 (Somregel) *De kans dat de vereniging zich voordoet ‘som-regel’:*

$$P(A \cup B) = P(A) + P(B) - P(A \cap B) \quad (27)$$

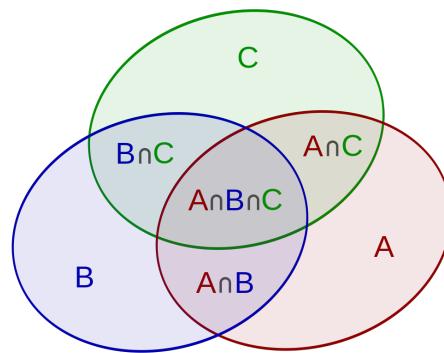
Als $P(A \cap B)$ leeg is, dan geldt:

$$P(A \cup B) = P(A) + P(B) \quad (28)$$

6.3.2 Algemene somregel

Voor 3 gebeurtenissen A,B,C

$$P(A \cup B \cup C) = P(A) + P(B) + P(C) - P(A \cap B) - P(A \cap C) - P(B \cap C) + P(A \cap B \cap C) \quad (29)$$



Figuur 75: $A \cap B \cap C$ moet terug opgeteld worden want hij werd te veel afgetrokken

6.4 Complementregel

1 dobbelsteen:

- Je gooit een 6
- Alle andere mogelijke gebeurtenissen = complement

Definitie 6.3 (Complementregel) Een gebeurtenis G en de complement-gebeurtenis $G_{complement}$ vormen samen alle mogelijkheden

$$\begin{aligned} P(G) + P(G_{complement}) &= 1 \\ P(G) &= 1 - P(G_{complement}) \end{aligned} \tag{30}$$

6.5 Samengesteld experiment (stochastisch exp)

Een samengesteld kansexperiment: hetzelfde experiment wordt een aantal keer herhaald

Verschillende oplossingsmethodes mogelijk:

- Kansboom
- Combinaties

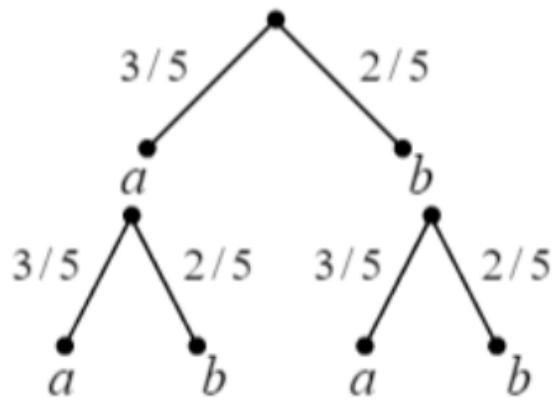
Voorbeeld: in een vaas liggen 5 briefjes waarop telkens 1 letter staat: 3 briefjes met de letter A en twee briefjes met de letter B

2 mogelijkheden:

- trekking met teruglegging
- trekking zonder teruglegging

6.5.1 Trekking met teruglegging

Uit de vaas wordt lukraak een letter getrokken. Deze wordt genoteerd en teruggelegd. Vervolgens trekken we opnieuw een letter.



Figuur 76: Verloop voorgesteld via een kansboom

- Uitkomstenverzameling: $U = \{AA, AB, BA, BB\}$
- Elke tak = bijhorende deelkans

Definitie 6.4 (Onafhankelijke deelexperimenten) Kansen bij de tweede trekking zijn onafhankelijk van het resultaat van de eerste trekking

Wat is $P(AB)$?

- N = groot aantal herhalingen

- $\frac{3}{5}N$ keer leveren een A op
- $\frac{2}{5}N$ keer leveren een B op
- **Gevolg:** $\frac{3}{5} \cdot \frac{2}{5}N$ keer volgen de weg AB
- $\Rightarrow P(AB) = \frac{6}{25}$

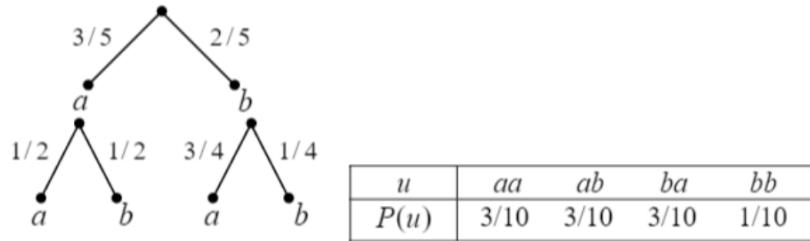
u	aa	ab	ba	bb
$P(u)$	$9/25$	$6/25$	$6/25$	$4/25$

Figuur 77

Productregel voor kansen: De kans op een weg is gelijk aan het product van de kansen langs die weg

6.5.2 Trekking zonder teruglegging

Uit de vaas wordt lukraak een letter genomen. Deze wordt genoteerd en niet teruggelegd. Vervolgens trekken we een nieuwe letter



Figuur 78: We vermenigvuldigen opnieuw de kansen

Definitie 6.5 (Afhankelijke deelexperimenten) Kansen bij de tweede trekking zijn afhankelijk zijn van het resultaat van de eerste trekking

6.6 Permutaties

Definitie 6.6 (Permutatie) Een permutatie van een verzameling elementen is een herschikking/herordening ervan.

\Rightarrow volgorde is van belang!

6.6.1 Voorbeeld:

Je organiseert een wedstrijd voor 10 hardlopers. Op hoeveel manieren kan ik de gouden/zilveren/bronzen medaille gaan verdelen?

- Volgorde van belang!
- Aantal permutaties van 3 uit 10: $10 \cdot 9 \cdot 8 = 720$

6.6.2 Oefening

Een reclameblok bestaat uit 9 items: 2 over eten, 3 over reizen, 4 over auto's

1. Op hoeveel manieren kunnen deze items gerangschikt worden?
2. Op hoeveel manieren kunnen deze items gerangschikt worden als de items over reizen direct na elkaar komen?

TODO: oplossing oefeningen

6.7 Combinaties

Definitie 6.7 (Combinaties) Een combinatie is een aantal manieren om k dingen uit n dingen te kiezen **zonder** dat de volgorde van belang is.

6.7.1 Wiskundig

'n boven k' of 'uit n kies ik er k'

$$\binom{n}{k} = \frac{n!}{(n-k)! \cdot k!} \quad (31)$$

6.7.2 Voorbeeld

Uit 63 studenten worden er 8 gekozen om deel te nemen aan de studentenparticipatie

$$\binom{63}{8} = \frac{63!}{(63-8)! \cdot 8!} = 3872894697$$

6.7.3 Oefening

We gooien met 5 identieke munstukken. Wat is de kans dat we 3 of 4 keer kop gooien?

- Gevraagd: $P(3$ keer kop of 4 keer kop)
- Totaal = $2 \cdot 2 \cdot 2 \cdot 2 \cdot 2 = 2^5 = 32$
- Aantal gunstige:
 - 3x kop: $\binom{5}{3} = 10$
 - 4x kop: $\binom{5}{4} = 5$
- Resultaat: somregel = $P(3$ keer kop of 4 keer kop) = $\frac{10}{32} + \frac{5}{32}$
- TODO

6.8 Theoretische en empirische kansen

6.8.1 Theoretische kansen:

- Eenvoudige kansberekeningen: gooien van een dobbelsteen, kans dat je 5 gooit
- Dit is een 'theoretische' of 'a priori' kans:
 - De kans op een gebeurtenis is vooraf berekenbaar

6.8.2 Empirische kansen

- Complexere kansberekening
- Uitkomst niet eenvoudig berekenbaar

6.8.3 Voorbeeld

We gooien 60 keer met een dobbelsteen, hoeveel vijfen verwacht je hierbij?

- Verwachting: $10x 1, 10x 2, 10x 3, \dots$
- Maar is de werkelijkheid = verwachting?
- Relatieve frequentie: $\frac{\text{frequentie aantal worpen}}{\text{aantal worpen}} = \frac{10}{60} = \frac{1}{6}$

Definitie 6.8 (De wet van de grote getallen) *Wanneer we een experiment heel vaak herhalen komt de relatieve frequentie steeds dichter in de buurt van de kans op de gebeurtenis.*



Figuur 79: De wet van grote getallen

- Nuttig wanneer we kans niet vooraf kunnen berekenen.
- Door het experiment heel vaak te herhalen, kunnen we de kans afleiden.
- Hierdoor mogelijkheid tot uitspraak over experiment in toekomst
- Dit heet **de empirische kans**: de kans gebaseerd op waarneming

6.8.4 Voorbeeld 2

Experiment met duimspijker: op hoeveel manieren kan hij bij een worp op de grond komen?

- We bekijken de kans $P(\text{punt omlaag})$ en $P(\text{punt omhoog})$
- Is dat wel 50/50?
- Theoretische kans niet te bepalen...
- Door experiment uit te voeren: de empirische kans $P(\text{punt omhoog}) \approx 0.58$

7 Voorwaardelijke kans

7.1 Doelstelling

- Voorwaardelijke kansen berekenen
- Onafhankelijke gebeurtenissen herkennen
- Regel van Bayes kennen en toepassen

7.2 Situering

Een patiënt heeft mogelijk last van griep, verkoudheid, of beide.

- Een verkouden patiënt heeft met 60% kans last van hoestbuien, met 10% kans op hoofdpijnklachten
- Een patiënt met griep heeft kans 20% op hoestbuien, met 70% kans op hoofdpijn

Vragen

1. De patient vertelt dat hij last heeft van hoestbuien. Hoe groot is de kans dat hij griep heeft?
2. Hij zegt dat hij ook last heeft van hoofdpijn. Hoe groot is de kans nu dat hij griep heeft?

7.3 Voorbeeld

Voorwaardelijke kans: kans die aan een voorwaarde voldoet

		geslacht		
		jongen	meisje	
sport	0	5	8	13
	1	22	17	39
	> 1	12	7	19
		39	32	71

Figuur 80

Bereken exact de kans dat een willekeurig gekozen...

1. ...leerling 1 sport beoefend?
 - $P(\text{leerling 1 sport}) = 39/71$
2. ...leerling een jongen is en niet sport?
 - $P(\text{jongen 1 sport}) = 5/71$
3. ...meisje meer dan 1 sport beoefend
 - $P(\text{meisje } > 1 \text{ sport}) = 7/32$
4. ...niet-sportende leerling een jongen is?
 - $P(\text{niet sportende jongen}) = 5/12$
5. ...meisje aan sport beoefend?
 - $P(\text{meisje doet sport}) = \frac{17+7}{32} = \frac{3}{4}$

Bij voorwaardelijke kansen bekijk je slechts een deel van de groep. Het aantal mogelijke uitkomsten waar je door deelt is dus een deelgroep

7.4 Wiskundig

De kans dat B onder de voorwaarde van A gebeurt:

$$P(B|A) = \frac{P(A \cap B)}{P(A)} = \frac{P(AB)}{P(A)} \quad (32)$$

Indien alle elementen in A en B evenwaardig zijn:

$$P(B|A) = \frac{P(A \cap B)}{P(A)} = \frac{\#(A \cap B)}{\#(A)} \quad (33)$$

7.4.1 Gevolg: productregel

We weten:

$$P(B|A) = \frac{P(AB)}{P(A)}$$

$$P(A|B) = \frac{P(AB)}{P(B)}$$

Daaruit volgt:

$$P(A \cap B) = P(AB) = P(B|A)P(A) = P(A|B)P(B) \quad (34)$$

De kans dat A en B samen voorkomen:

- = kans op B onder voorwaarde van A * kans op A
- = kans op A onder voorwaarde van B * kans op B

7.4.2 Algemene productregel

Met 3 verzamelingen:

$$\begin{aligned} P(A \cap B \cap C) &= P(ABC) \\ &= P(C|AB) \cdot P(AB) \\ &= P(C|AB) \cdot P(B|A) \cdot P(A) \end{aligned} \quad (35)$$

Algemeen:

$$P(ABC \dots Z) = P(Z|ABC \dots Y) \cdot \dots \cdot P(C|AB)P(B|A)P(A) \quad (36)$$

7.5 Oefening 1

7.5.1 Opgave

Drie machines M_1, M_2, M_3 produceren items:

	M_1	M_2	M_3
percentage productie	20	30	50
percentage defect	1	2	3

We kiezen willekeurig 1 item uit de totale productie.

Wat is de kans dat dit item kapot is?

Figuur 81: Opgave

7.5.2 Oplossing

Je vermenigvuldigt voor elke machine het productiepercentage met het defect percentage voor die machine. Die kansen tel je op:

$$P(D|M_1) \cdot P(M_1) + P(D|M_2) \cdot P(M_2) + P(D|M_3) \cdot P(M_3) \quad (37)$$

$$= 0.01 \cdot 0.02 + 0.02 \cdot 0.3 + 0.03 \cdot 0.5 \quad (38)$$

$$= 0.023 \quad (39)$$

7.6 Oefening 2

7.6.1 Opgave

Drie machines M_1, M_2, M_3 produceren items:

	M_1	M_2	M_3
percentage productie	20	30	50
percentage defect	1	2	3

We kiezen willekeurig 1 item uit de totale productie, en dit blijkt kapot te zijn.

Wat is de kans dat dit item door M_2 gemaakt is?

Figuur 82: Opgave

7.6.2 Oplossing

$$P(M_2|D) = \frac{P(M_2 \cap D)}{P(D)} = \frac{P(D|M_2) \cdot P(M_2)}{P(D)} \quad (40)$$

$$= \frac{0.02 \cdot 0.3}{TODO} \quad (41)$$

$$= \quad (42)$$

7.7 Onafhankelijke gebeurtenissen

7.7.1 Voorbeeld

In een groep van 65 jongeren is er onderzoek gedaan of ze wel of niet sporten

3 gebeurtenissen:

- A: een leerling sport
- B: een leerling is een jongen
- C: een leerling is een meisje

		geslacht		
		jongen	meisje	
sport	wel	11	44	55
	niet	2	8	10
totaal		13	52	65

Figuur 83

- Kans dat een leerling sport: $P(A) = \frac{55}{65} = \frac{11}{13}$
- Kans dat een leerling sport onder voorwaarde dat het een jongen is: $P(A|B) = \frac{11}{13}$
- **Vaststelling:** of een leerling sport beoefend is onafhankelijk of het een jongen is

Wanneer gebeurtenis A en B onafhankelijk zijn geldt $P(A \text{ onder voorwaarde } B) = P(A)$

- Kans dat leerling een jongen is: $P(B) = \frac{13}{65} = \frac{1}{5}$
- Kans dat leerling een jongen is onder de voorwaarde dat hij sport: $P(B|A) = \frac{11}{55} = \frac{1}{5}$

Wanneer gebeurtenis A en B onafhankelijk zijn geldt $P(A \text{ onder voorwaarde } B) = P(A)$ en dus ook $P(B \text{ voorwaarde } A) = P(B)$

7.7.2 Gevolg: Productregel voor onafhankelijke gebeurtenissen

Indien A en B onafhankelijk zijn, dan is de kans dat B onder voorwaarde van A gebeurt:

- $P(B|A) = P(B)$
- $P(A|B) = P(A)$

Productregel wordt dan:

$$P(A \cap B) = P(B) \cdot P(A) = P(A) \cdot P(B) \quad (43)$$

7.8 Productregel algemeen

Voor een n-tal willekeurige gebeurtenissen A_1, A_2, \dots, A_n geldt:

$$P(A_1 \cap A_2 \cap \dots \cap A_n) = P(A_1) \cdot P(A_2|A_1) \dots P(A_n|A_1 \cap A_2 \cap \dots \cap A_{n-1}) \quad (44)$$

Men schrijft dit kort als:

$$P(A_1 A_2 \dots A_n) = P(A_1) \cdot P(A_2|A_1) \dots P(A_n|A_1 A_2 \dots A_{n-1}) \quad (45)$$

7.8.1 Oefening 1

Opgave: controleer of het hebben van blauwe ogen al dan niet afhankelijk is van het geslacht

	Blauwe ogen	Geen blauwe ogen	
Jongen	15	5	20
Meisje	30	10	40
	45	15	

Figuur 84: Opgave

Oplossing:

7.8.2 Oefening 2

Opgave: controleer of kans op bloedgroep AB al dan niet onafhankelijk is van de waarde Rh+

	AB	O	
Rh+	15	155	170
Rh-	5	25	30
	20	180	

Figuur 85: Opgave

7.9 Regel van Bayes

7.9.1 Toepassingsdomein

- Belangrijkste deel uit kansrekening
- Toepassing in medische en genetische diagnostiek, AI, organisatie van dataverkeer, wetenschappelijk onderzoek

Een test op *HIV* is 90% betrouwbaar: als een persoon *HIV* heeft is de kans op een positieve uitslag 0.9, en als een persoon geen *HIV* heeft is de kans op een positieve uitslag 0.1. De kans op *HIV* is 0.05.

Een dame ondergaat de test en de uitslag is positief. Wat is de kans dat zij *HIV* heeft?

Figuur 86: Voorbeeld 1

Een test op *HIV* is 90% betrouwbaar: als een persoon *HIV* heeft is de kans op een positieve uitslag 0.9, en als een persoon geen *HIV* heeft is de kans op een positieve uitslag 0.1. De kans op *HIV* is 0.05.

Een dame ondergaat de test en de uitslag is positief. Wat is de kans dat zij *HIV* heeft?

Figuur 87: Voorbeeld 2

7.9.2 Formule van Bayes (eenvoudige vorm)

Uit:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} \text{ en } P(B|A) = \frac{P(A \cap B)}{P(A)}$$

volgt:

$$P(A \cap B) = P(A|B) \cdot P(B) = P(B|A) \cdot P(A)$$

dus:

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A) \cdot P(A)}{P(B)} \quad (46)$$

7.9.3 Voorbeeld

De kans dat een zekere kerncentrale oververhit raakt (O) is $1 \cdot 10^{-7}$, de kans dat een lek (L) ontstaat is $1 \cdot 10^{-8}$, maar als er eenmaal een lek is, is de kans groot dat de centrale oververhit raakt: 0.1

Wat is de kans dat de centrale gaat lekken als hij oververhit raakt?

$$P(L|O) = \frac{P(L \cap O)}{P(O)} = \frac{P(O|L) \cdot P(L)}{P(O)}$$

Dus:

$$P(L|O) = \frac{0.1 \cdot 1 \cdot 10^{-8}}{1 \cdot 10^{-7}} = 0.01$$

7.10 Regel van Bayes (variant 1)

We beschouwen van een gebeurtenis H ook zijn complement \bar{H} .

Ter herinnering:

- $P(B) + P(\bar{B}) = 1$
- $P(B|A) = \frac{P(B \cap A)}{P(A)}$
- $P(B|\bar{A}) = \frac{P(B \cap \bar{A})}{P(\bar{A})}$

Daaruit volgt:

$$P(B) = P(B \cap A) + P(B \cap \bar{A}) \quad (47)$$

$$= P(B|A) \cdot P(A) + P(B|\bar{A}) \cdot P(\bar{A}) \quad (48)$$

Een test op HIV is 90% betrouwbaar: als een persoon HIV heeft is de kans op een positieve uitslag 0.9, en als een persoon geen HIV heeft is de kans op een positieve uitslag 0.1. De kans op HIV is 0.05.

Een dame ondergaat de test en de uitslag is positief. Wat is de kans dat zij HIV heeft?

De gevraagde kans is $P(HIV|POS)$. In de notatie van de Stelling van Bayes wordt dat $H = HIV$ en $E = POS$.

Met de Stelling van Bayes:

$$P(HIV|POS) = \frac{P(POS|HIV)P(HIV)}{P(POS|HIV)P(HIV) + P(POS|\bar{HIV})P(\bar{HIV})}.$$

Daarmee

$$P(HIV|POS) = \frac{0.9 \cdot 0.05}{0.9 \cdot 0.05 + 0.1 \cdot 0.95} = 0.32.$$

Figuur 88: Voorbeeld 1

Een test op *HIV* is 90% betrouwbaar: als een persoon *HIV* heeft is de kans op een positieve uitslag 0.9, en als een persoon geen *HIV* heeft is de kans op een positieve uitslag 0.1. De kans op *HIV* is 0.05.

Een dame ondergaat de test en de uitslag is positief. Wat is de kans dat zij *HIV* heeft?

De gevraagde kans is $P(HIV|POS)$. In de notatie van de Stelling van Bayes wordt dat $H = HIV$ en $E = POS$.

Met de Stelling van Bayes:

$$P(HIV|POS) = \frac{P(POS|HIV)P(HIV)}{P(POS|HIV)P(HIV) + P(POS|\bar{HIV})P(\bar{HIV})}.$$

Daarmee

$$P(HIV|POS) = \frac{0.9 \cdot 0.05}{0.9 \cdot 0.05 + 0.1 \cdot 0.95} = 0.32.$$

Figuur 89: Voorbeeld 2

7.11 Regel van Bayes (variant 2)

We beschouwen van een gebeurtenis H ook zijn complement \bar{H} , waarbij nu geldt dat $P(H) = P(\bar{H})$

$$P(H|E) = \frac{P(E|H) \cdot P(H)}{P(E|H) \cdot P(H) + P(E|\bar{H}) \cdot P(\bar{H})} \quad (49)$$

$$= \frac{P(E|H)}{P(E|H) + P(E|\bar{H})} \quad (50)$$

7.12 Regel van Bayes (algemeen)

Verzameling B in verschillende gevallen onderverdeeld (B_1, B_2, \dots, B_n) die elkaar uitsluiten: $B_i \cap B_j = \emptyset$ als $i \neq j$

$$P(B_j|A) = \frac{P(A|B_j) \cdot P(B_j)}{\sum_{i=1}^n P(A|B_i) \cdot P(B_i)} \quad (51)$$

Vb. Een restaurant schenkt Bordeaux, Beaujolais en Merlot. Wanneer een gast geen keuze kan maken kiest de ober, kans 0.6 dat hij Bordeaux, 0.3 dat hij Beaujolais en 0.1 dat hij Merlot kiest. Bij elk van deze wijnen zijn er gasten die het niet smaakt. Stel dat de kans daarop voor Bordeaux, Beaujolais en Merlot respectievelijk 0.01, 0.04 en 0.2 is. Een gast krijgt een door de ober gekozen wijn en vindt die niet lekker.

Wat is de kans dat het Bordeaux is?

H_1 : de wijn is Bordeaux, H_2 : de wijn is Beaujolais, H_3 : de wijn is Merlot. E : de gast vindt de wijn niet lekker.

H_1 : de wijn is Bordeaux, H_2 : de wijn is Beaujolais, H_3 : de wijn is Merlot. E : de gast vindt de wijn niet lekker.

$$P(H_1|E) = \frac{P(E|H_1)P(H_1)}{\sum_{j=1}^3 P(E|H_j)P(H_j)} = \frac{0.01 \cdot 0.6}{0.01 \cdot 0.6 + 0.04 \cdot 0.3 + 0.2 \cdot 0.1} = 0.1578947.$$

Wat is de kans dat het Beaujolais is?

$$P(H_2|E) = \frac{P(E|H_2)P(H_2)}{\sum_{j=1}^3 P(E|H_j)P(H_j)} = \frac{0.04 \cdot 0.3}{0.01 \cdot 0.6 + 0.04 \cdot 0.3 + 0.2 \cdot 0.1} = 0.3157895. \quad P(H_3|E) = \frac{P(E|H_3)P(H_3)}{\sum_{j=1}^3 P(E|H_j)P(H_j)} = \frac{0.2 \cdot 0.1}{0.01 \cdot 0.6 + 0.04 \cdot 0.3 + 0.2 \cdot 0.1} = 0.5263158.$$

Figuur 90: Voorbeeld algemene regel van Bayes

7.13 Bayesiaans leren: toepassing in machine learning

Def. *Bayesiaans leren* heeft (in essentie) de volgende vorm:

Er zijn een aantal *hypotheses* H_1, \dots, H_n die samen de uitkomstenruimte vormen.

De hypotheses zijn meer of minder waarschijnlijk: de (initiële) bijbehorende verdeling is de *a-priori* verdeling, de kansen $P(H_i)$ zijn de *a-priori* kansen.

Na het verkrijgen van nieuwe informatie/data/gebeurtenis E worden de kansen van de hypotheses aangepast volgens de stelling van Bayes:

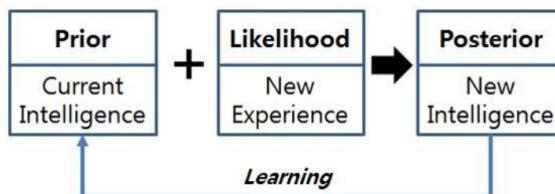
$$P(H_i | E) = \frac{P(E | H_i)P(H_i)}{\sum_{j=1}^n P(E | H_j)P(H_j)}.$$

De kansen $P(H_i | E)$ zijn de *a-posteriori* kansen. De kansen $P(E | H_i)$ zijn de *likelihoods* van E .

Na het verkrijgen van nieuwe informatie/data/gebeurtenis E worden de kansen van de hypotheses aangepast volgens de stelling van Bayes:

$$P(H_i | E) = \frac{P(E | H_i)P(H_i)}{\sum_{j=1}^n P(E | H_j)P(H_j)}.$$

De kansen $P(H_i | E)$ zijn de *a-posteriori* kansen. De kansen $P(E | H_i)$ zijn de *likelihoods* van E .



Na het verkrijgen van nieuwe informatie/data/gebeurtenis E worden de kansen van de hypotheses aangepast volgens de stelling van Bayes:

$$P(H_i | E) = \frac{P(E | H_i)P(H_i)}{\sum_{j=1}^n P(E | H_j)P(H_j)}.$$

De kansen $P(H_i | E)$ zijn de *a-posteriori* kansen. De kansen $P(E | H_i)$ zijn de *likelihoods* van E .

Leren:

Op grond van telkens nieuwe data E_1, E_2, \dots wordt de verdeling van de hypotheses voortdurend aangepast, P_0, P_1, P_2, \dots :

- P_0 is de *a-priori* verdeling, waarbij $P_0(H_i) = P(H_i)$.
- Na het verkrijgen van data E_1 wordt de nieuwe verdeling P_1 , waarbij $P_1(H_i) = P_0(H_i | E_1)$.
- Na het verkrijgen van data E_2 wordt de nieuwe verdeling P_2 , waarbij $P_2(H_i) = P_1(H_i | E_2)$.
- Etc.

Figuur 91: TODO: tekst ipv images

8 Logging

8.1 Inleiding

8.1.1 Waarom?

- Diagnose van een specifiek probleem
- Normale flow van een applicatie controleren
- Gebeurtenissen registreren
- Werken met verschillende niveau's: niet elk logbericht is even belangrijk
- Flexibere manier nodig dan enkel maar print-statements

8.1.2 Opzet

- Je wenst zelf niveau van loggen bepalen
- Formaat van logbericht vooraf definieren
- Locatie (scherm/bestand/...) vooraf vastleggen

8.1.3 Logging levels

Level	Wanneer te gebruiken?
DEBUG	Gedetailleerde info, typisch bestemd voor de diagnose van een probleem
INFO	Bevestiging van zaken die verondersteld werden te lukken
WARNING	Indicatie dat er iets onverwacht gebeurd is, of iets dat in de nabije toekomst zal plaatsvinden (bv: 'disk space low')
ERROR	Door een ernstig probleem was de software niet in staat om een bepaalde functie uit te voeren
CRITICAL	Zeer ernstig probleem dat het programma verhindert om verder te zetten

Figuur 92

8.2 Basis logging

```
1 # stap 1: niveau instellen (DEBUG, INFO, WARNING, ERROR, CRITICAL)
2 logging.basicConfig(level=logging.CRITICAL)
3
4 # stap 2: voorzie code van log-methodes op gewenste plaatsen
5 logging.debug("dit is een debug log")
6 logging.info("dit is een inf log")
7 ...
8 logging.critical("dit is een critical log")
```

8.2.1 Spelregels

- Er wordt pas gelogd als het niveau van de logmethode gelijk of groter is dan het ingesteld logniveau

```
1 logging.basicConfig(level=logging.ERROR) # stel loggingniveau in
2 logging.debug("...") # Niet gelogd
3 logging.info("...") # Niet gelogd
4 logging.warning("...") # Niet gelogd
```

```

5  logging.error("...")          # Wel gelogd
6  logging.critical("...")       # Wel gelogd

```

8.2.2 Format logbericht

```

1  # Voorbeeld 1
2  logging.basicConfig(level=logging.ERROR, format="*** %(message)s ***")
3  # Voorbeeld 2
4  logging.basicConfig(level=logging.DEBUG,
5      format="%(asctime)s\t%(levelname)s -- %(processName)s%(filename)s:%(lineno)s -- %(message)s")

```

8.2.3 Naar een file loggen

```

1  logging.basicConfig(
2      filename='logging.txt',
3      level=logging.DEBUG,
4      format="%(asctime)s - %(message)s"
5 )

```

8.2.4 Toegepast

(zie ook demos)

```

1 @studentennummer.setter
2  def studentennummer(self, value):
3      if isinstance(value, int):
4          self.__studentennummer = value
5      else:
6          logging.error("Geen geldig studentennummer: %s" % value)
7          raise ValueError("Geen geldig studentennummer!")

```



```

1  @staticmethod
2  def inlezen_studenten(bestandsnaam):
3      studenten = []
4      try:
5          logging.info("Bestand %s inlezen" % bestandsnaam)
6          fo = open(bestandsnaam)
7          lijn = fo.readline() # eerste lijn = niets mee doen
8          lijn = fo.readline()
9          while (lijn != ""):
10             lijn = lijn.rstrip('\n')
11             delen = lijn.split(';')
12             try:
13                 student = Student(delen[0], delen[1], delen[2], delen[3], int(delen[4]))
14                 studenten.append(student)
15             except:
16                 logging.debug("Ongeldig lijn bij inlezen file: %s" % lijn)
17                 lijn = fo.readline() # lees volgende lijn in
18             logging.info("Bestand volledig ingelezen")
19             fo.close()

```

```

20     except FileNotFoundError as ex:
21         logging.error("Bestand niet gevonden")
22         raise ex
23     return studenten
24
25
26
27 @staticmethod
28 def opslaan_studenten(bestandsnaam, studenten):
29     try:
30         logging.info("Start opslaan studenten naar bestand: %s" % bestandsnaam)
31         fo = open(bestandsnaam, "w")
32         fo.write("Naam;Voornaam;StudentenNummer;Opleiding;Geboortejaar\n")
33         for student in studenten:
34             fo.write(student.naam + ";" + student.voornaam + ";" + str(student.studentnummer) + ";")
35         fo.close()
36         logging.info("Opslaan studenten voltooid")
37     except FileNotFoundError as ex:
38         logging.error("Bestand niet gevonden")
39         raise ex
40     except Exception as ex:
41         logging.error("Onverwachte fout: %s" % ex)

```

Meer info: <https://docs.python.org/3.9/howto/logging.html#logging-basic-tutorial>

9 Threading

9.1 Doelstelling

- In python een thread-klasse kunnen programmeren
- Threads op elkaar kunnen afstemmen

9.2 Wat zijn threads?

Thread: 'flow of control through a program'

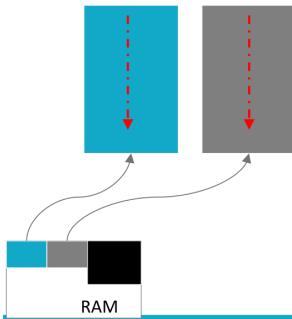
- Heeft geen gescheiden geheugengebruik
- Deelt deze met andere threads gecreëerd door hetzelfde programma
- Kan wel nog lokale variabelen hebben: elke thread heeft zijn eigen stack met lokale variabelen

9.3 Process vs Thread

PROCESSES	THREADS
Processes don't share memory	Threads share memory
Spawning/switching processes is expensive	Spawning/switching threads is less expensive
Processes require more resources	Threads require fewer resources (are sometimes called lightweight processes)
No memory synchronisation needed	You need to use synchronisation mechanisms to be sure you're correctly handling the data

Figuur 93: Overzicht

9.3.1 Process

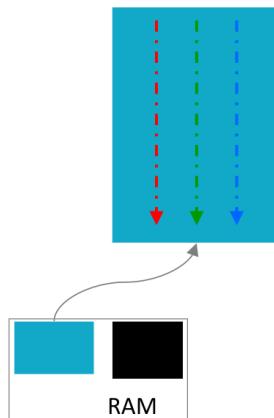


Figuur 94: Twee processen

Twee processen:

- Elk eigen geheugenruimte (virtual address space)
- Gescheiden code
- Data gescheiden
- CPU beslist welk proces uitgevoerd wordt

9.3.2 Thread



Figuur 95: 3 threads in 1 proces

- Zelfde geheugenruimte (virtual address space)
- Zelfde code wordt gedeeld
- Data kan gedeeld worden
- Voor elke thread: soort virtual CPU

2 types threads in python:

- User-level threads: eigen ontwikkelde threads

- Kernel-level threads: low-level threads veroorzaakt door operating system

9.4 De threading library

```

import threading
import time
from threading import Thread

def sleepMe(i):
    print(f"++Thread {i} started: {threading.current_thread()}")
    time.sleep(10)
    print(f"++Thread {i} is finished")

# main-thread afprinten
print(threading.main_thread())

for i in range(1,10):
    th = Thread(target=sleepMe, args=(i, ))
    th.start()
    print(f"\nCurrent Thread count: {threading.active_count()}.")

```

Figuur 96: De threading library

9.4.1 Thread aanmaken

```

1 # Via de constructor van de thread klasse
2 th = Thread(target=sleepMe, args=(i, ))

```

Mogelijke parameters:

- group: a special parameter which is reserved for future extension
- target: the callable object to be invoked by the run method(), if None: nothing will be started
- name: our threads name
- args: arguments tuple for target invocation, defaults to ()
- kwargs: dictionary keyword argument to invoke the base class constructor

9.4.2 Andere methods

- :
-
-

```

1 # huidige thread wordt in slaaptoestand gebracht (voor x seconden)
2 time.sleep(x)
3 # geeft het aantal active threads weer
4 # Vergeet ook de main-thread niet mee te tellen
5 threading.active_count()
6 # Main-thread opvragen:
7 threading.main_thread()
8 # Overlopen van huidige threads:
9 print("Active threads:")
10 for thread in threading.enumerate():
11     print(f">Thread name is {thread.getName()}")

```

9.4.3 Timer

Uitvoeren van een thread na een specifieke tijd laten opstarten

```
1 import threading
2
3 def delayed():
4     print("I am printed after 5 seconds")
5
6 print("Demo timer")
7 thread = threading.Timer(5, delayed)
8 thread.start()
9 print("thread is opgestart")
10
11 # output:
12 > "Demo timer"
13 > "thread is opgestart"
14 > "I am printed after 5 seconds!"
```

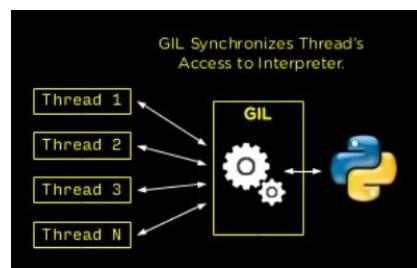
9.5 Multithreading

Veronderstelling dat meerdere threads 'gelijktijdig' uitgevoerd worden:

- Processing core switcht heel snel tussen de verschillende threads en geeft zo de impressie dat ze dat ze gelijktijdig plaatsvinden
- De keuze tussen de threads wordt bepaald door de context

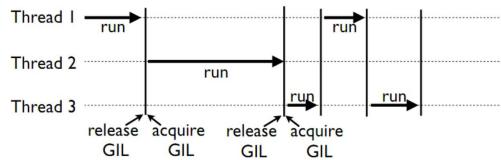
9.5.1 Python Thread Management

- Memory management is not thread-safe!
- Global Interpreter Lock (GIL):
 - Slechts 1 thread kan van de python interpreter gebruikmaken
 - Hierdoor kunnen python threads niet helemaal gelijktijdig uitgevoerd worden
 - GIL gaat dus de verschillende threads serialiseren naar de interpreter



Figuur 97: GIL

Elke thread dient een GIL-lock te verkrijgen:

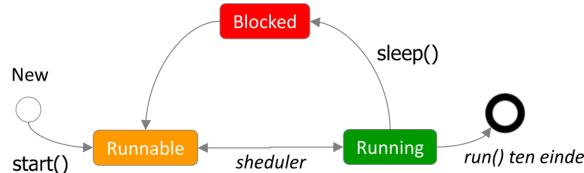


Figuur 98

9.6 Thread toestanden

Een thread kan in verschillende toestanden bevinden:

- Runnable state: thread is aangemaakt
- Running state: thread is gestart
- Blocked: sleeping, wachten tot een andere thread klaar is, wachten op data, ...



Figuur 99: Schema thread toestanden

9.7 Opgelet met threads!

- Een thread dat ten einde is, kan niet nogmaals uitgevoerd worden
- Python VM heeft geen thread management: het is niet mogelijk aan threads prioriteiten te hangen

9.8 Wanneer threads gebruiken?

- Bij I/O bound processing
- Beperkte CPU-intensieve operaties

9.9 Thread-class

9.9.1 Maken van een nieuwe thread-klasse

Werkwijze:

1. maak een klasse die erft van de `Thread` class
2. overschrijf de `__init__` methode: geef eventueel parameters langs deze weg door
3. overschrijf de `__run__` methode

```

1 class MyThread(threading.Thread):
2
3     def __init__(self, threadID, name, counter, delay):
4         threading.Thread.__init__(self)
5         self.threadID = threadID
6         self.name = name
7         self.counter = counter
8         self.delay = delay
9
10    def run(self):
11        print(f"Starting {self.name}")
12        while self.counter >= 0:
13            time.sleep(self.delay)
14            print(f"{self.name}: {time.ctime(time.time())} -> counter: {self.counter}")
15            self.counter -= 1
16        print(f"Exiting {self.name}")

```

Figuur 100: Voorbeeld MyThread-kLASSE

9.9.2 Opstarten van een thread

1. Maak een object van de thread-kLASSE
2. Voer de methode `start()` uit

```

1 # create threads
2 thread1 = MyThread(1, "Thread-1", 10, 1)
3 thread2 = MyThread(1, "Thread-1", 20, 2)
4
5 # start new threads
6 thread1.start()
7 thread2.start()

```

9.10 Threading-synchronisation

9.10.1 Join-methode

Laat toe op een bepaalde thread te wachten

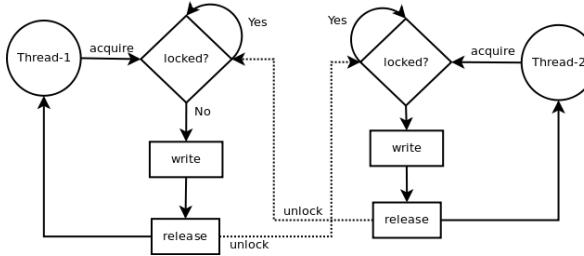
```

1 # create threads
2 thread1 = MyThread(1, "Thread-1", 10, 1)
3 thread2 = MyThread(1, "Thread-1", 20, 2)
4
5 # start new threads
6 thread1.start()
7 thread2.start()
8
9 thread1.join()
10 thread2.join()
11 print("Exiting Main thread")

```

9.10.2 Thread locking

Locking mechanisme: lock-objecten controleren de toegang tot een gemeenschappelijk data-object



Figuur 101: Schema 2 threads met locking

```

1 # Vooraf: thread-lock-object aanmaken
2 myThreadLock = threading.Lock()
3
4 # in de run-methode:
5 # - lock aanvragen
6 # - lock vrijgeven
7
8 def run(self):
9     print(f"Starting {self.name}")
10    while MyThread.global_counter >= 0:
11        myThreadLock.acquire()
12        print(f"{self.name}: {time.ctime(time.time())} -> counter: {MyThread.global_counter}")
13        MyThread.global_counter -= 1
14        myThreadLock.release()
15        time.sleep(self.delay)
16    print(f"Exiting {self.name}")

```

9.10.3 Condition

Condition mechanisme:

- Analoog als lock, maar nu verwittigen we ook andere threads
- Gebruikt in pattern: Producer/Consumer pattern

```

1 # condition object
2 condition = threading.Condition()
3
4 # gebruik op de juiste plaats:
5 def consumer(cond):
6     """ Wait for the condition and use the resource """
7     logging.debug("Starting consumer thread")
8     with cond:
9         cond.wait()
10        logging.debug("Resource is available to consumer")
11
12 def producer(cond):
13     """ Set up the resource to be used by the consumer """
14     logging.debug("Starting producer thread")
15     with cond:
16         logging.debug("Making resources available")

```

17 cond.notifyAll()

Uitvoering:

```
1 condition = threading.Condition()  
2  
3 c1 = threading.Thread(name='c1', target=consumer, args=(condition, ))  
4 c2 = threading.Thread(name='c2', target=consumer, args=(condition, ))  
5 p = threading.Thread(name='p', target=producer, args=(condition, ))  
6  
7 c1.start()  
8 time.sleep(0.2)  
9 c2.start()  
10 time.sleep(0.2)  
11 p.start()
```

Output:

```
1 2018-03-13 09:32:16,235 (c1) Starting consumer thread  
2 2018-03-13 09:32:16,435 (c2) Starting consumer thread  
3 2018-03-13 09:32:16,635 (p ) Starting producer thread  
4 2018-03-13 09:32:16,643 (p ) Making resources available  
5 2018-03-13 09:32:16,644 (c1) Resource is available to consumer  
6 2018-03-13 09:32:16,644 (c2) Resource is available to consumer
```

9.11 Daemon vs not-daemon threads

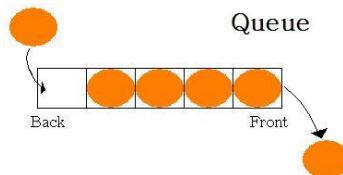
Tot nu toe: elke thread heeft dezelfde prioriteit

- Dit is niet steeds nodig: sommige threads mogen/kunnen onderbroken worden (zonder gevaar op verlies van data)
- Werkwijze: via methode set_daemon() op true plaatsen

9.12 Multithreaded queue

Een queue en threads kunnen heel nauw samenwerken met elkaar

9.12.1 Queue



Figuur 102: Queue

Method	Description
get()	It will give an item from the queue
put()	It adds an item to the queue.
qsize()	This method will return a total number of elements in the queue.
Empty()	If the queue is empty it will return true else false.
Full()	If the queue is full the method will return true else false.

Figuur 103

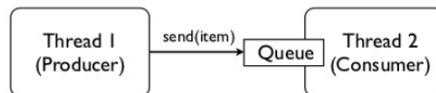
```

1  from Queue import Queue
2
3  q = Queue([maxsize])          # create a queue
4  q.put(item)                  # put an item on the queue
5  q.get()                      # get an item from the queue
6  q.empty()                    # check if empty
7  q.full()                     # check if full

```

9.12.2 Threading + queue

Threading is soms eenvoudiger indien ze in een producer/consumer model geplaatst worden. Ze geven data via een queue door aan elkaar.



- Voordeel: een queue heeft geen locks nodig.
- Heeft bovendien zijn eigen signalisatiemechanisme:

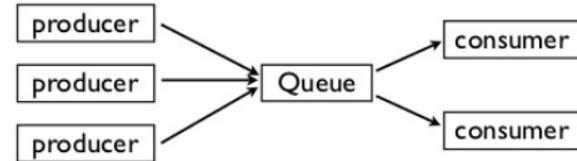
```

1  q.task_done()                # signal that work is done
2  q.join()                     # wait for all work to be done
3
4  # in producer thread:
5  for item in produce_items():
6      q.put(item)
7  q.join()                     # wait for consumer
8
9
10 # in consumer thread:
11 while True:
12     item = q.get()
13     consume_item(item)
14     q.task_done()

```

9.12.3 Queue programming

Er zijn heel veel verschillende werkwijzen om een queue te integreren. Geen beperking in het aantal producers/consumers:



Figuur 104: Meerdere producers/consumers met een queue