



**Mathematics for Machine Learning**  
**Lesnota's**

Stijn Lievens

Professionele Bachelor in de Toegepaste Informatica

# Inhoudsopgave

<b>Inhoudsopgave</b>	<b>i</b>
<b>Voorwoord</b>	<b>v</b>
<b>I Lineaire Algebra</b>	<b>1</b>
<b>1 Vectoren</b>	<b>2</b>
1.1 Basisbewerkingen op vectoren . . . . .	2
1.2 Lineaire combinaties van vectoren . . . . .	6
1.3 Lineaire deelruimte . . . . .	10
1.4 Lengte en inwendig product van vectoren . . . . .	14
1.4.1 Lengte van vectoren . . . . .	14
1.4.2 Afstand tussen twee vectoren . . . . .	15
1.4.3 Orthogonale vectoren . . . . .	17
1.4.4 Betekenis van het inwendig product . . . . .	18
1.5 Projectie op een vector . . . . .	20
1.6 Oefeningen . . . . .	21
<b>2 Matrices</b>	<b>24</b>
2.1 Lineaire transformaties en matrixproduct . . . . .	24
2.1.1 Matrix-vector product . . . . .	27
2.1.2 Samenstelling van lineaire transformaties . . . . .	28
2.1.3 Het matrixproduct . . . . .	30
2.2 Transponeren van een matrix . . . . .	33
2.2.1 Inwendig product als matrixproduct . . . . .	34
2.3 Bewerkingen en eigenschappen van matrices . . . . .	35
2.4 Inverse van een matrix . . . . .	37
2.5 Loodrechte projectie op een deelruimte . . . . .	41
2.6 Oefeningen . . . . .	45

<b>3</b>	<b>Stelsels lineaire vergelijkingen</b>	<b>48</b>
3.1	Stelsels lineaire vergelijkingen . . . . .	48
3.1.1	Oefeningen . . . . .	50
3.2	Gaussische eliminatie . . . . .	50
3.2.1	Oefeningen . . . . .	54
3.3	De LU matrixdecompositie . . . . .	55
3.3.1	LU matrixdecompositie zonder rijverwisselingen . . . . .	55
3.3.2	LU decompositie met rijverwisselingen . . . . .	57
3.3.3	Oefeningen . . . . .	59
3.4	Toepassingen van de LU-decompositie . . . . .	60
3.4.1	Oplossen van stelsels lineaire vergelijkingen . . . . .	60
3.4.2	Berekenen van de inverse matrix . . . . .	63
3.4.3	Berekenen van de determinant . . . . .	64
3.4.4	Oefeningen . . . . .	66
3.5	Oplossen van strijdige stelsels . . . . .	66
3.5.1	Toepassing: kleinste kwadraten methode . . . . .	68
3.5.2	Oefeningen . . . . .	73
<b>4</b>	<b>Orthogonale matrices</b>	<b>74</b>
4.1	Orthonormale vectoren . . . . .	74
4.1.1	Oefeningen . . . . .	76
4.2	Orthogonale matrices . . . . .	76
4.2.1	Oefeningen . . . . .	79
4.3	Projectie op orthonormale basis . . . . .	80
4.3.1	Projectie op bestaande basisvectoren . . . . .	80
4.3.2	Oefeningen . . . . .	81
4.4	Creëren van een orthonormale basis . . . . .	81
4.5	De QR-decompositie . . . . .	84
4.5.1	Oefeningen . . . . .	85
4.6	Lineaire stelsels en de QR-decompositie . . . . .	85
<b>5</b>	<b>De singuliere waarden ontbinding</b>	<b>88</b>
5.1	De singuliere waarden ontbinding . . . . .	88
5.1.1	De volledige singuliere waarden ontbinding . . . . .	88
5.1.2	De zuinige SVD . . . . .	89
5.1.3	Lage rang benadering m.b.v. de SVD . . . . .	91
5.2	Principale Componenten Analyse . . . . .	95
5.2.1	Voorbeeld: de MNIST dataset . . . . .	98
5.2.2	Verklaarde variabiliteit . . . . .	101

5.3	Moore-Penrose pseudoinverse . . . . .	105
5.4	Stelsels oplossen met de pseudoinverse . . . . .	109
5.5	Oefeningen . . . . .	113
<b>II</b>	<b>Analyse</b>	<b>116</b>
<b>6</b>	<b>Reële functies in één veranderlijke</b>	<b>117</b>
6.1	Inleiding reële functies . . . . .	117
6.1.1	Limieten . . . . .	120
6.1.2	Continuïteit . . . . .	123
6.1.3	Oefeningen . . . . .	123
6.2	Afgeleiden . . . . .	124
6.2.1	Oefeningen . . . . .	128
6.3	Afgeleide van som en product . . . . .	129
6.4	De kettingregel voor afgeleiden . . . . .	131
6.4.1	Oefeningen . . . . .	134
6.5	Het belang van de afgeleide . . . . .	135
6.5.1	Oefeningen . . . . .	137
6.6	De exponentiële en logaritmische functie . . . . .	138
6.6.1	Exponentiële groei . . . . .	138
6.6.2	De exponentiële functie . . . . .	141
6.6.3	De natuurlijke logaritmische functie . . . . .	142
6.6.4	Andere grondtallen . . . . .	144
6.6.5	De logistische functie . . . . .	144
6.6.6	Oefeningen . . . . .	147
6.7	Newton-Raphson methode voor nulpunten . . . . .	148
6.7.1	Oefeningen . . . . .	150
<b>7</b>	<b>Reële functies in meerdere veranderlijken</b>	<b>152</b>
7.1	Definitie en voorstelling functies . . . . .	152
7.2	Partiële afgeleiden . . . . .	155
7.2.1	Oefeningen . . . . .	158
7.3	De gradiënt . . . . .	158
7.4	Gradient descent . . . . .	162
7.4.1	Lineaire regressie met gradient descent . . . . .	166
7.4.2	Oefeningen . . . . .	171
7.5	Kettingregel in meerdere veranderlijken . . . . .	172
7.5.1	Oefeningen . . . . .	175

7.6	Exacte numerieke bepaling gradiënt . . . . .	176
7.6.1	Oefeningen . . . . .	181
<b>A</b>	<b>Inleiding tot numpy</b>	<b>189</b>
A.1	Basisdatastructuur . . . . .	189
A.2	Random numpy arrays genereren . . . . .	192
A.3	Rekenkundige bewerkingen . . . . .	192
A.3.1	Broadcasting . . . . .	193
A.4	Indexeren van arrays . . . . .	195
A.4.1	Indexeren van ééndimensionale arrays . . . . .	195
A.4.2	Indexeren van meerdimensionale arrays . . . . .	197
A.5	Aggregatiemethoden in numpy . . . . .	199
A.6	Lineaire algebra . . . . .	200
A.7	Vergelijken van arrays . . . . .	201

# Voorwoord

Dit is de cursustekst voor het opleidingsonderdeel “Mathematics for Machine Learning”, uit het tweede jaar professionele bachelor toegepaste informatica voor de studenten in het keuzetraject “AI & Data Engineering”.

Dit opleidingsonderdeel heeft als bedoeling om je de nodige wiskundige vaardigheden bij te brengen zodat je in latere opleidingsonderdelen zoals “Machine Learning” en “Deep Learning” beter begrijpt wat er achter de schermen van de algoritmen die daarin worden gebruikt gebeurt. Tegelijkertijd maak je een aantal oefeningen m.b.v. de Python-bibliotheek `numpy` die een zeer belangrijke rol speelt in het Python-ecosysteem om aan datawetenschap te doen. Het zal je ook helpen om later vlot met andere Python-bibliotheken zoals `tensorflow` of `pytorch` aan de slag te kunnen gaan.

De wiskunde die aan de grondslag ligt van de meeste algoritmen in machine learning en deep learning is enerzijds lineaire algebra (voor het efficiënt opstellen van de modellen) en anderzijds optimalisatie van een (kost)functie in meerdere veranderlijken. De cursus is dan ook opgebouwd uit twee onderdelen. Het eerste deel, bestaande uit de eerste 5 hoofdstukken, geeft een inleiding tot een aantal belangrijke concepten binnen het domein van de lineaire algebra. Het tweede deel, bestaande uit de laatste twee hoofdstukken, geeft aan hoe reële functies in meerdere veranderlijken kunnen worden geminimaliseerd. Het eerste van deze twee hoofdstukken geeft kort een inleiding tot een aantal belangrijke begrippen uit de reële analyse in één veranderlijke. Het laatste hoofdstuk gaat dan over reële functies in meerdere veranderlijken.

We overlopen nu kort de inhoud van elk hoofdstuk.

In Hoofdstuk 1 introduceren we het concept vector en leggen we uit hoe je kan rekenen met vectoren. Verder komen in dit hoofdstuk de uiterst belangrijke concepten van lineaire (on)afhankelijkheid van vectoren en een basis

van een vectorruimte aan bod. M.b.v. het inwendig product van vectoren kunnen we bepalen wanneer twee vectoren orthogonaal zijn en meer algemeen kunnen we op die manier bepalen wat de ingesloten hoek is tussen twee vectoren. Tenslotte illustreren we hoe je m.b.v. het inwendig product de (loodrechte) projectie van een vector op (de deelruimte opgespannen door) een andere vector kan bepalen.

Hoofdstuk 2 introduceert het begrip matrix aan de hand van het concept van een lineaire transformatie. Het matrix-vector product, een lineaire combinatie van de kolommen van de matrix, bepaalt het beeld van een vector onder een lineaire transformatie. De samenstelling van lineaire transformaties leidt op een natuurlijke manier tot het product van twee matrices. We definiëren wat de inverse matrix is van een vierkante matrix en zien dat deze inverse matrix enkel bestaat wanneer de matrix de grootst mogelijke rang heeft. Dit hoofdstuk eindigt met een uitbreiding van het resultaat uit het vorige hoofdstuk, nl. het vinden van de loodrechte projectie van een vector op de deelruimte opgespannen door een aantal vectoren.

Hoofdstuk 3 bekijkt stelsels lineaire vergelijkingen. We zien hoe stelsels lineaire vergelijkingen compact voor te stellen zijn a.d.h.v. matrices en hoe Gaussische eliminatie in combinatie met achterwaartse substitutie een manier geeft om zo'n stelsels op te lossen. Gaussische eliminatie kan gebruikt worden om de LU-decompositie van een matrix te bepalen. De LU-decompositie heeft een aantal toepassingen zoals het oplossen van stelsels, het bepalen van de inverse matrix of het berekenen van de determinant. In de laatste sectie van dit hoofdstuk bekijken we strijdige stelsels, dit zijn stelsels zonder oplossing. In bepaalde gevallen kunnen we de "beste" benaderende oplossing vinden door het rechterlid te projecteren op de vectorruimte opgespannen door de kolommen van de coëfficiëntenmatrix van het stelsel. Als toepassing hierop bepalen we hoe we de best passende rechte kunnen vinden door een aantal gegeven datapunten. Dit is het basisidee achter lineaire regressie.

Het volgende en korte hoofdstuk gaat dieper in op de begrippen orthogonale en orthonormale vectoren en bekijkt hoe orthonormale vectoren verzameld kunnen worden in een orthogonale matrix. Een orthogonale matrix is zeer interessant want zijn inverse is gelijk aan de getransponeerde matrix en is bijgevolg zeer eenvoudig (en snel) te berekenen. We zien hoe de Gram-Schmidt methode toelaat om een verzameling lineair onafhankelijke

vectoren te transformeren in een verzameling orthonormale vectoren die dezelfde vectorruimte opspannen. Hieruit volgt onmiddellijk dat een matrix met lineair onafhankelijke kolommen steeds kan geschreven worden als het product van een orthogonale matrix en een bovendriehoeksmatrix: de zogenaamde QR-decompositie.

Het laatste hoofdstuk over lineaire algebra bespreekt enkele toepassingen van de allerbelangrijkste matrixdecompositie: de singuliere waarden ontbinding (SVD) die van toepassing is op *elke* matrix. De SVD kan gebruikt worden om de beste lage rang benadering te vinden voor een matrix. We zien hoe de SVD kan gebruikt worden om aan principale componenten analyse te doen. Bij principale componenten analyse (PCA) trachten we een gegeven verzameling datapunten voor te stellen met minder attributen op een zodanige manier dat zoveel mogelijk van de informatie behouden blijft. We zien ook hoe de SVD kan gebruikt worden om de Moore-Penrose pseudoinverse te definiëren: dit is een matrix die zich zoveel mogelijk gedraagt als de “echte” inverse (als die zou bestaan). Deze pseudoinverse kan dan gebruikt worden om alle lineaire stelsels “op te lossen”, en men bekomt steeds oplossingen met zeer interessante eigenschappen.

De volgende twee hoofdstukken vormen samen het deel “Analyse” van de cursus. In Hoofdstuk 6 starten we met de definitie van een reële functie en zien hoe men reële functies kan creëren door gebruik te maken van eenvoudigere reële functies. Verder geven we kort wat intuïtie mee m.b.t. het begrip limiet. We besteden redelijk wat aandacht aan het begrip afgeleide, hoe deze te bepalen en wat het belang ervan is. We herhalen ook de begrippen exponentiële en logaritmische functie die je in principe al hebt gezien in het eerste jaar. We eindigen met de methode van Newton-Raphson die kan gebruikt worden om de nulpunten van een reële functie te benaderen.

In het laatste hoofdstuk van de cursus bekijken we functies in meerdere veranderlijken, met een focus op functies met veel variabelen maar slechts één reëel getal als uitkomst. We zien hoe gradient descent kan gebruikt worden om een (lokaal) minimum van zo’n functie te vinden. Hiervoor worden de begrippen partiële afgeleide en gradiënt geïntroduceerd. Opdat gradient descent vlot zou werken is het nodig dat men de gradiënt efficiënt kan bepalen, ook wanneer er vele duizenden variabelen zijn. In de laatste sectie zien we hoe men de gradiënt (in een punt) exact (op afrondingsfouten door de floating-point voorstelling na) kan bepalen door één voorwaartse



en achterwaartse berekening te doen in de berekeningsgraaf. Het is onder andere dit algoritme dat aan de grondslag ligt van het huidige succes van deep learning algoritmen.

## Studieadvies

Je zal vast wel gemerkt hebben aan de voorgaande beschrijving dat **de inhoud van deze cursus uitdagend is**. Het is daarom zeer belangrijk om het studeren van deze cursus op een goede manier aan te pakken.

Ten eerste zou ik aanraden om steeds naar de les te komen: sommige concepten lijken misschien moeilijk op papier maar zijn in essentie eenvoudig. Dit kan hopelijk naar voor komen tijdens de les. Ten tweede is het belangrijk om de cursus bij te houden: op die manier zullen de lessen ook (veel) zinvoller zijn. Tot slot is het ook belangrijk om oefeningen te maken: deze zullen helpen om de theoretische concepten duidelijker te maken. De berekeningen van de oefeningen kunnen hoofdzakelijk m.b.v. Python worden gedaan: het is niet de bedoeling om lange en saaie berekeningen “met de hand” te doen.

## Bedanking en oproep

Dit is het tweede jaar dat dit opleidingsonderdeel wordt gegeven en deze cursus werd volledig “from scratch” geschreven. Ik bedank mijn collega’s Koen Mertens, Jan Willem en Mario Verstraeten voor het nalezen van verschillende draftversies van deze cursus. Johan Decorte ben ik dankbaar voor het aanbrengen van een oefening i.v.m. stelsels lineaire vergelijkingen. Enkele studenten merkten ook nog wat foutjes op die werden aangepast in de eerste versie van de cursus en hebben die gemeld via het forum, waarvoor dank!

Ondanks de zorgvuldigheid waarmee deze tekst werd geschreven en nagelezen ben ik er zeker van dat er nog veel zaken voor verbetering vatbaar zijn (en de verantwoordelijkheid hiervoor ligt volledig bij mij). Vandaar ook mijn oproep aan jullie, de studenten, om “alles” wat beter kan (zoals bv. spellings- en grammaticale fouten, onduidelijkheden, inconsistente notaties, ...) te melden. Ik zal een forum in Chamilo aanmaken om deze errata en suggesties te verzamelen. Ik gebruik deze errata en suggesties om de cursus aan te passen zodat die beter wordt voor jullie opvolgers.

Tot slot wens ik jullie veel lees-, leer- en oefenplezier.

Stijn Lievens, september 2023

# Deel I

# Lineaire Algebra

# Vectoren

## 1.1 Basisbewerkingen op vectoren

Vectoren kunnen ruwweg op drie verschillende manieren worden bekeken:

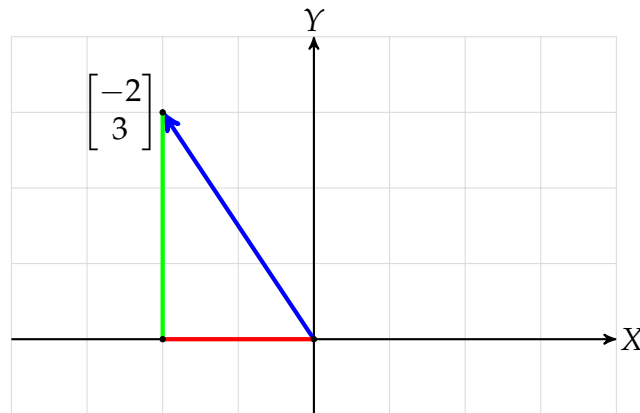
- vanuit het *fysica* perspectief;
- vanuit het *informatica* perspectief;
- vanuit het *wiskunde* perspectief.

In het **fysica** perspectief zijn vectoren *pijlen in de ruimte*. Een vector wordt bepaald door zijn lengte en de richting waarin deze pijl wijst. In het fysica-perspectief maakt het niet uit waar de kop en de staart van de pijl zich bevinden. Als de lengte en richting gelijk zijn dan worden de vectoren als gelijk beschouwd.

In het **informatica** perspectief bestaat een vector uit een *geordende lijst van getallen*. In zekere zin is een vector dus een ééndimensionale array.

Het **wiskundig** perspectief definieert vectoren als *objecten* die je op een betekenisvolle manier kan *optellen* bij elkaar en waar het mogelijk is om een vector *met een getal te vermenigvuldigen*. Dit perspectief veralgemeent de twee vorige perspectieven.

Om te kunnen wisselen tussen het fysica en het informatica perspectief introduceren we een coördinatensysteem. In twee dimensies betekent dit dat we twee assen kiezen, nl. een horizontale X-as en een verticale Y-as. De



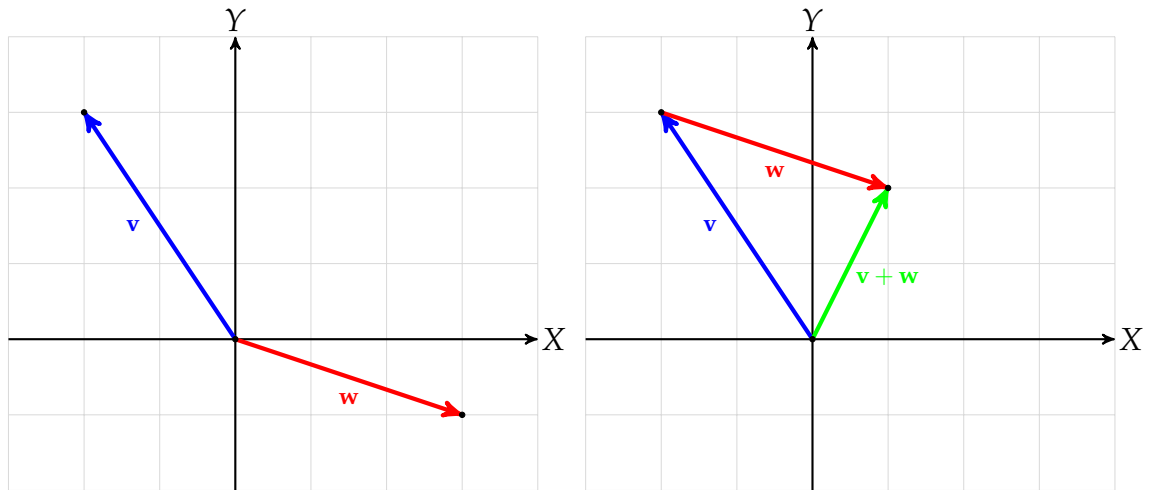
**Figuur 1.1:** Verband tussen fysica en informatica perspectief a.d.h.v. een coördinatensysteem.

plaats waar de  $X$ -as en de  $Y$ -as elkaar snijden noemen we de oorsprong. De oorsprong stelt de plaats voor waar alle vectoren starten. Verder kiezen we een bepaalde afstand om de lengte 1 voor te stellen. Op die manier krijgt elk punt van het tweedimensionale vlak een unieke coördinaat.

Bekijk de vector in Figuur 1.1. De staart van de vector is de oorsprong. De coördinaten van de kop van de vector geven je de instructies om van de oorsprong de kop te bereiken. In dit geval zijn de coördinaten  $-2$  en  $3$ . Dit betekent dat je, startend in de oorsprong, eerst twee eenheden naar links (want de  $X$ -coördinaat is negatief) moet bewegen en vervolgens drie eenheden naar boven. Deze vector wordt dan als volgt genoteerd als een (verticale) lijst van twee geordende getallen:

$$\begin{bmatrix} -2 \\ 3 \end{bmatrix}.$$

Om twee vectoren  $\mathbf{v}$  en  $\mathbf{w}$  op te tellen, stellen we ons eerst voor dat de staart van beide vectoren zich in de oorsprong bevinden. Daarna verplaatsen we de staart van de vector  $\mathbf{w}$  naar de kop van de vector  $\mathbf{v}$ , en we kijken waar de kop van deze verschoven vector zich bevindt. Deze positie bepaalt de kop van de vector die de som van  $\mathbf{v}$  en  $\mathbf{w}$  voorstelt. Dit proces wordt geïllustreerd in Figuur 1.2. Algebraïsch komt dit proces van de optelling neer op het optellen van de overeenkomende  $X$  resp.  $Y$  componenten van de vecto-



**Figuur 1.2:** De optelling van twee vectoren  $\mathbf{v}$  en  $\mathbf{w}$ .

ren:

$$\begin{bmatrix} v_1 \\ v_2 \end{bmatrix} + \begin{bmatrix} w_1 \\ w_2 \end{bmatrix} = \begin{bmatrix} v_1 + w_1 \\ v_2 + w_2 \end{bmatrix}.$$

Deze definitie is zinvol als men deze bekijkt vanuit het standpunt van opeenvolgende bewegingen:  $\mathbf{v} + \mathbf{w}$  staat dan voor de beweging die men verkrijgt door eerst volgens  $\mathbf{v}$  te bewegen en vervolgens volgens  $\mathbf{w}$ . Dit proces wordt geïllustreerd in Figuur 1.3.

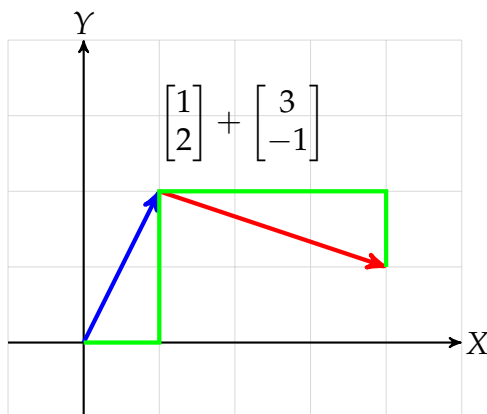
Naast optellen van twee vectoren ligt het ook voor de hand om een bepaalde vector te bekijken (in het fysica perspectief) en deze vector te (her)schalen, i.e. om de lengte van de vector te wijzigen terwijl de richting ongewijzigd blijft. Figuur 1.4 toont een illustratie waarbij een vector  $\mathbf{v}$  herschaald wordt met een factor 2 en één waarbij dezelfde vector herschaald wordt met een factor  $-1/3$ . Uit deze figuur zou moeten duidelijk zijn dat

$$2 \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} 2v_1 \\ 2v_2 \end{bmatrix},$$

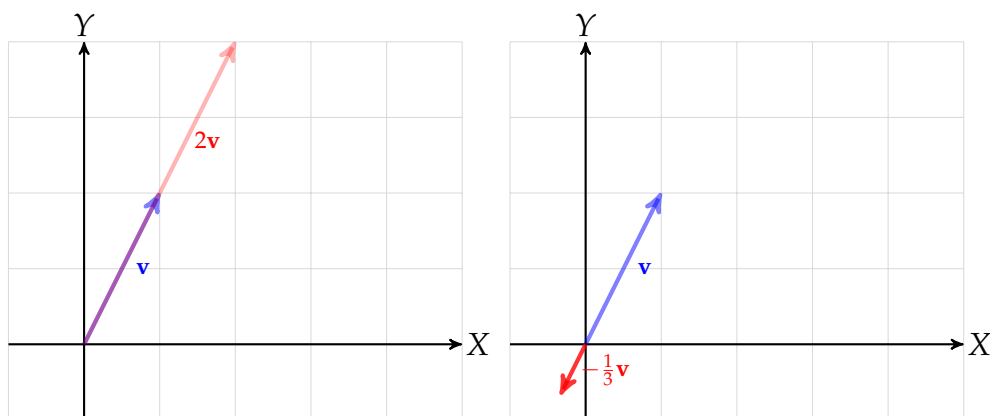
en in het algemeen

$$\alpha \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} \alpha v_1 \\ \alpha v_2 \end{bmatrix}.$$

Merk op dat als  $\alpha$  een negatief getal is, dat de resulterende vector dan de andere kant op wijst.



**Figuur 1.3:** De optelling van twee vectoren als achtereenvolgende beweging:  $\begin{bmatrix} 1 \\ 2 \end{bmatrix} + \begin{bmatrix} 3 \\ -1 \end{bmatrix}$  betekent dat men in totaal 4 eenheden beweegt in de horizontale richting en men 1 eenheid beweegt in de verticale richting, wat overeenkomt met de vector  $\begin{bmatrix} 4 \\ 1 \end{bmatrix}$ .



**Figuur 1.4:** Herschalen van de vector  $\mathbf{v}$  met factoren 2 en  $-1/3$  respectievelijk.

In de voorbeelden tot hier toe hadden de vectoren steeds twee componenten. Hetgeen we verteld hebben kan echter op een eenvoudige manier uitgebreid worden tot vectoren met een willekeurig aantal componenten. De verzameling  $\mathbb{R}^n$  stelt alle vectoren voor met  $n$  reële componenten waarbij  $n$  een strikt positief natuurlijk getal is.

**Definitie 1.1** De SOM VAN TWEE VECTOREN  $\mathbf{v}$  en  $\mathbf{w}$  van  $\mathbb{R}^n$  wordt gedefinieerd als de vector van  $\mathbb{R}^n$  waarvan elke component de som is van de overeenkomstige componenten van  $\mathbf{v}$  en  $\mathbf{w}$ :

$$\begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} + \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix} = \begin{bmatrix} v_1 + w_1 \\ v_2 + w_2 \\ \vdots \\ v_n + w_n \end{bmatrix}.$$

Het SCALAIR PRODUCT van een reëel getal  $\alpha$  en een vector  $\mathbf{v}$  wordt als volgt gedefinieerd

$$\alpha \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} \alpha v_1 \\ \alpha v_2 \\ \vdots \\ \alpha v_n \end{bmatrix}.$$

■

**Opmerking 1.2** De som van twee vectoren  $\mathbf{v}$  en  $\mathbf{w}$  is niet gedefinieerd wanneer  $\mathbf{v}$  en  $\mathbf{w}$  een verschillend aantal componenten hebben. ■

## 1.2 Lineaire combinaties van vectoren

Een vector als

$$\begin{bmatrix} 3 \\ -2 \end{bmatrix}$$

kunnen we ook als volgt bekijken

$$3 \begin{bmatrix} 1 \\ 0 \end{bmatrix} + (-2) \begin{bmatrix} 0 \\ 1 \end{bmatrix} = 3\mathbf{i} + (-2)\mathbf{j},$$

waarbij

$$\mathbf{i} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{en} \quad \mathbf{j} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$



We zeggen dat  $\mathbf{i}$  en  $\mathbf{j}$  een “basis”<sup>1</sup> vormen voor het coördinatensysteem. De coördinaten van een vector zijn dan scalair die deze basisvectoren schalen en de som is de resulterende vector.

Op zichzelf is er *niets bijzonders* aan de vectoren  $\mathbf{i}$  en  $\mathbf{j}$ , en we kunnen ook twee andere vectoren kiezen om als basis te dienen voor ons coördinatensysteem. Veronderstel dat we de vectoren

$$\mathbf{v} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \text{en} \quad \mathbf{w} = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$$

als basis kiezen. We kunnen ons dan bv. de vraag stellen welke  $\alpha$  en  $\beta$  we moeten kiezen opdat

$$\alpha \mathbf{v} + \beta \mathbf{w} = 5\mathbf{i} - \frac{1}{2}\mathbf{j}?$$

Het is niet zo moeilijk om te verifiëren dat  $\alpha = \frac{1}{2}$  en  $\beta = \frac{3}{2}$  ervoor zorgen dat de vergelijking opgaat. Omgekeerd, stel dat we  $\alpha$  en  $\beta$  vrij kunnen laten variëren, *welke vectoren kunnen we dan allemaal bereiken?* Overtuig jezelf, bv. door het maken van een tekening, dat je *alle* vectoren van  $\mathbb{R}^2$  op deze manier kan bereiken.

Veronderstel dat we nu twee andere vectoren beschouwen, bv.

$$\mathbf{a} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \text{en} \quad \mathbf{b} = \begin{bmatrix} -2 \\ -4 \end{bmatrix}.$$

Bekijk nu alle vectoren die je kan bereiken door  $\alpha$  en  $\beta$  in de formule hieronder te laten variëren:

$$\alpha \mathbf{a} + \beta \mathbf{b}.$$

In dit geval is het *niet* mogelijk om alle vectoren van  $\mathbb{R}^2$  te bereiken. Het is bv. onmogelijk om de vector

$$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

te bereiken. Maak een figuur om jezelf hiervan te overtuigen.

**Definitie 1.3** Een LINEAIRE COMBINATIE van  $m$  vectoren  $\mathbf{v}_1$  t.e.m.  $\mathbf{v}_m$  is een som van de vorm

$$\alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \cdots + \alpha_m \mathbf{v}_m,$$

---

<sup>1</sup>Formele definitie volgt later.

waarbij  $\alpha_1$  t.e.m.  $\alpha_m$  reële getallen zijn. De verzameling van alle lineaire combinaties van de vectoren  $\mathbf{v}_1$  t.e.m.  $\mathbf{v}_m$  wordt hun OMHULSEL (Eng. *span*) genoemd en noteren we met

$$\text{span}\{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_m\} = \text{alle lineaire combinaties van } \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_m\}. \quad \blacksquare$$

**Voorbeeld 1.4 (Omhulsel/Span)** Voor de vectoren

$$\mathbf{v} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \text{en} \quad \mathbf{w} = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$$

geldt dat

$$\text{span}\{\mathbf{v}, \mathbf{w}\} = \mathbb{R}^2.$$

Voor de vectoren

$$\mathbf{a} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \text{en} \quad \mathbf{b} = \begin{bmatrix} -2 \\ -4 \end{bmatrix}$$

geldt echter dat

$$\text{span}\{\mathbf{a}, \mathbf{b}\} = \left\{ \begin{bmatrix} \alpha \\ 2\alpha \end{bmatrix} \mid \alpha \in \mathbb{R} \right\}.$$

Bovendien zien we dat

$$\text{span}\{\mathbf{a}, \mathbf{b}\} = \text{span}\{\mathbf{a}\} = \text{span}\{\mathbf{b}\}. \quad \blacksquare$$

In het vorige voorbeeld is er een duidelijk verschil tussen de verzamelingen van vectoren  $\{\mathbf{v}, \mathbf{w}\}$  en  $\{\mathbf{a}, \mathbf{b}\}$ . In het eerste geval verkleint het omhulsel wanneer je een vector uit de verzameling weglaat, in het tweede geval kan je een vector uit de verzameling verwijderen zonder het omhulsel te wijzigen. We zeggen dat de verzameling  $\{\mathbf{v}, \mathbf{w}\}$  bestaat uit lineair onafhankelijke vectoren, terwijl de verzameling  $\{\mathbf{a}, \mathbf{b}\}$  uit lineair afhankelijke vectoren bestaat.

De formele definitie van lineaire (on)afhankelijkheid bekijkt of het mogelijk is om de nulvector te bekomen m.b.v. een lineaire combinatie waarbij minstens één coëfficiënt verschillend is van nul.

**Definitie 1.5** Een verzameling vectoren  $\mathbf{v}_1$  t.e.m.  $\mathbf{v}_m$  is LINEAIR ONAFHANKELIJK als en slechts als de enige manier om de nulvector te bekomen erin bestaat om een lineaire combinatie te nemen waarin alle coëfficiënten gelijk zijn aan nul. In symbolen:

$$\alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \dots + \alpha_m \mathbf{v}_m = \mathbf{0} \iff \alpha_1 = \alpha_2 = \dots = \alpha_m = 0.$$

Een verzameling vectoren die niet lineair onafhankelijk is, wordt LINEAIR AFHANKELIJK genoemd. ■

**Voorbeeld 1.6** De verzameling vectoren  $\{\mathbf{a}, \mathbf{b}\}$  met

$$\mathbf{a} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \text{en} \quad \mathbf{b} = \begin{bmatrix} -2 \\ -4 \end{bmatrix}$$

is lineair afhankelijk omdat

$$2\mathbf{a} + \mathbf{b} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = \mathbf{0}.$$

Dit is een voorbeeld van een lineaire combinatie waarbij sommige coëfficiënten verschillend zijn van nul en toch de nulvector wordt bekomen.

De verzameling vectoren

$$\mathbf{v} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \text{en} \quad \mathbf{w} = \begin{bmatrix} 3 \\ -1 \end{bmatrix}$$

is lineair onafhankelijk. Proberen we immers een lineaire combinatie te vinden die gelijk is aan de nulvector, nl.

$$\alpha \begin{bmatrix} 1 \\ 2 \end{bmatrix} + \beta \begin{bmatrix} 3 \\ -1 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

dan volgt dat

$$\begin{cases} \alpha + 3\beta = 0 \\ 2\alpha - \beta = 0. \end{cases}$$

Dit is een stelsel lineaire vergelijkingen (zie Hoofdstuk 3), waarvan de unieke oplossing gegeven wordt door  $\alpha = \beta = 0$ . Dit toont aan dat de vectoren  $\mathbf{v}$  en  $\mathbf{w}$  lineair onafhankelijk zijn. ■

**Opmerking 1.7** Hieronder geven we enkele observaties die interessant kunnen zijn wanneer men moet bepalen of een verzameling vectoren lineair (on)afhankelijk is.

1. Een verzameling bestaande uit 1 vector is steeds lineair onafhankelijk tenzij het de nulvector is.

2. Een verzameling bestaande uit 2 vectoren (verschillend van de nulvector) is lineair afhankelijk als en slechts als de vectoren veelvouden zijn van elkaar.
3. Als men aan een lineair afhankelijke verzameling vectoren extra vectoren toevoegt dan is deze grotere verzameling vectoren ook steeds lineair afhankelijk. ■

### 1.3 Lineaire deelruimte

De deelverzameling  $V$  van  $\mathbb{R}^2$ , die als volgt wordt gedefinieerd

$$V = \left\{ \begin{bmatrix} m \\ 2m \end{bmatrix} \mid m \in \mathbb{R} \right\}$$

heeft een aantal bijzondere eigenschappen:

- Een willekeurig veelvoud van een vector uit  $V$  behoort opnieuw tot  $V$ . Bijvoorbeeld, de vector

$$\begin{bmatrix} 3 \\ 6 \end{bmatrix}$$

behoort tot  $V$ , maar ook

$$2 \begin{bmatrix} 3 \\ 6 \end{bmatrix} = \begin{bmatrix} 6 \\ 12 \end{bmatrix} \quad \text{en} \quad - \begin{bmatrix} 3 \\ 6 \end{bmatrix} = \begin{bmatrix} -3 \\ -6 \end{bmatrix} \quad \text{en} \quad 0 \begin{bmatrix} 3 \\ 6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

behoren tot  $V$ .

- De som van twee willekeurige vectoren uit  $V$  behoort opnieuw tot  $V$ . Bijvoorbeeld:

$$\begin{bmatrix} 6 \\ 12 \end{bmatrix} \quad \text{en} \quad \begin{bmatrix} -3 \\ -6 \end{bmatrix}$$

behoren tot  $V$ , en hun som

$$\begin{bmatrix} 3 \\ 6 \end{bmatrix}$$

behoort ook tot  $V$ .

We kunnen deze twee eigenschappen ook korter samenvatten en zeggen dat de lineaire combinatie van om het even welke twee vectoren van  $V$  opnieuw tot  $V$  behoort. Een verzameling van vectoren met deze eigenschappen noemen we een lineaire deelruimte.

**Definitie 1.8** Een LINEAIRE DEELRUIMTE (Eng. *linear subspace*)  $V$  van  $\mathbb{R}^n$  is een verzameling van vectoren zodanig dat elke lineaire combinatie van twee vectoren uit  $V$  opnieuw tot  $V$  behoort. In symbolen:

$$\mathbf{v} \in V \text{ en } \mathbf{w} \in V \implies \alpha \mathbf{v} + \beta \mathbf{w} \in V,$$

voor alle reële getallen  $\alpha$  en  $\beta$ . ■

**Opmerking 1.9** Door  $\alpha$  en  $\beta$  allebei gelijk aan nul te stellen is het onmiddellijk duidelijk dat de nulvector tot elke lineaire deelruimte behoort. Dit betekent ook onmiddellijk dat een verzameling vectoren die de nulvector niet bevat nooit een lineaire deelruimte kan zijn. ■

Het is heel eenvoudig om lineaire deelruimten te vinden:

- De verzameling bestaande uit enkel de nulvector  $V = \{\mathbf{0}\}$  is een lineaire deelruimte.
- De verzameling  $\mathbb{R}^n$  is een lineaire deelruimte.
- Het omhulsel van een aantal vectoren  $\mathbf{v}_1$  t.e.m.  $\mathbf{v}_m$  is steeds een lineaire deelruimte.

**Voorbeeld 1.10 (Deelverzameling die geen lineaire deelruimte is.)** De verzameling van vectoren  $V$  van de vorm

$$\begin{bmatrix} \alpha \\ \alpha^2 \end{bmatrix}$$

met  $\alpha \in \mathbb{R}$  is geen lineaire deelruimte. Inderdaad,

$$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

behoort tot  $V$  maar

$$\begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

wat een veelvoud is van de eerste vector behoort *niet* tot  $V$ . ■

Bekijk de deelruimte  $V$  die we reeds voordien bespraken:

$$V = \left\{ \begin{bmatrix} m \\ 2m \end{bmatrix} \mid m \in \mathbb{R} \right\}$$

Binnen deze verzameling  $V$  is het mogelijk om een verzameling (bijzondere) vectoren te kiezen zodanig dat

1. elke vector in  $V$  te schrijven is als lineaire combinatie van deze vectoren, en
2. deze verzameling vectoren lineair onafhankelijk is.

Voor de verzameling  $V$  kan je bv. de verzameling

$$\left\{ \begin{bmatrix} 1 \\ 2 \end{bmatrix} \right\}$$

kiezen. Een verzameling vectoren die aan de twee bovenstaande voorwaarden voldoet noemen we een basis voor de deelruimte.

**Voorbeeld 1.11** Beschouw de deelverzameling  $V$  van  $\mathbb{R}^3$  bestaande uit alle vectoren waarbij de eerste en laatste component aan elkaar gelijk zijn, en waarbij de tweede component om het even wat kan zijn. Vectoren van  $V$  zijn dus van de vorm

$$\begin{bmatrix} \alpha \\ \beta \\ \alpha \end{bmatrix} \quad \text{met } \alpha, \beta \in \mathbb{R}.$$

Overtuig jezelf dat  $V$  inderdaad een lineaire deelruimte is van  $\mathbb{R}^3$ .

De vectoren

$$\begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \quad \text{en} \quad \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

vormen samen een basis voor  $V$  want deze vectoren zijn lineair onafhankelijk van elkaar en

$$\begin{bmatrix} \alpha \\ \beta \\ \alpha \end{bmatrix} = \alpha \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} + \beta \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}.$$

Dit is echter niet de enige mogelijke basis. We kunnen bv. ook

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \text{en} \quad \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

kiezen. Deze vectoren zijn eveneens lineair onafhankelijk en er geldt

$$\begin{bmatrix} \alpha \\ \beta \\ \alpha \end{bmatrix} = \alpha \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} + (\beta - \alpha) \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}.$$

**Definitie 1.12** Een BASIS voor een lineaire deelruimte  $V$  bestaat uit een verzameling vectoren  $\mathbf{v}_1$  t.e.m.  $\mathbf{v}_m$  van  $V$  die

- lineair onafhankelijk is en
- zodanig is dat elke vector uit  $V$  kan geschreven als lineaire combinatie van  $\mathbf{v}_1$  t.e.m.  $\mathbf{v}_m$ .

**Eigenschap 1.13** Als  $\mathbf{v}_1$  t.e.m.  $\mathbf{v}_m$  een basis is voor  $V$  dan geldt dat elke vector  $\mathbf{v}$  op een *unieke* manier kan geschreven worden als lineaire combinatie van  $\mathbf{v}_1$  t.e.m.  $\mathbf{v}_m$ . Deze unieke combinatie noemen we ook de COÖRDINATEN van  $\mathbf{v}$  t.o.v. deze basis.

*Bewijs* We weten uit de definitie van een basis reeds dat elke vector kan geschreven worden als *een* lineaire combinatie van de basisvectoren. We moeten enkel nog aantonen dat er geen twee verschillende lineaire combinaties bestaan voor dezelfde vector  $\mathbf{v}$ .

Veronderstel dat we een vector  $\mathbf{v}$  op twee manieren kunnen schrijven als lineaire combinatie van  $\mathbf{v}_1$  t.e.m.  $\mathbf{v}_m$ , dan moeten we aantonen dat deze twee manieren identiek zijn. M.a.w. stel dat

$$\mathbf{v} = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 + \cdots + \alpha_m \mathbf{v}_m$$

en

$$\mathbf{v} = \beta_1 \mathbf{v}_1 + \beta_2 \mathbf{v}_2 + \cdots + \beta_m \mathbf{v}_m$$

dan volgt onmiddellijk dat

$$\mathbf{0} = (\alpha_1 - \beta_1) \mathbf{v}_1 + (\alpha_2 - \beta_2) \mathbf{v}_2 + \cdots + (\alpha_m - \beta_m) \mathbf{v}_m.$$

Uit het feit dat de verzameling vectoren  $\mathbf{v}_1$  t.e.m.  $\mathbf{v}_m$  lineair onafhankelijk is volgt nu onmiddellijk dat

$$\alpha_1 = \beta_1, \alpha_2 = \beta_2, \dots, \alpha_m = \beta_m.$$

Dit is precies wat we wilden aantonen.  $\diamond$

We hebben al gezien dat een basis voor een lineaire deelruimte niet uniek is, maar het zal wel steeds zo zijn dat alle basissen hetzelfde aantal elementen hebben.

**Eigenschap 1.14** Als  $\mathbf{v}_1$  t.e.m.  $\mathbf{v}_m$  een basis is van  $V$  en  $\mathbf{w}_1$  t.e.m.  $\mathbf{w}_k$  een andere basis is van  $V$ , dan zal steeds gelden dat  $m = k$ . Het aantal elementen van een basis noemen we de DIMENSIE van de deelruimte. ■

*Bewijs* Zonder bewijs.  $\diamond$

## 1.4 Lengte en inwendig product van vectoren

### 1.4.1 Lengte van vectoren

In sectie 1.1 zijn we gestart met te vermelden dat vectoren een lengte hebben. In twee dimensies is het gemakkelijk om de lengte van een vector te bepalen a.d.h.v. de stelling van Pythagoras. Als we kijken naar Figuur 1.1 dan zien we dat

$$\text{lengte} \begin{bmatrix} -2 \\ 3 \end{bmatrix} = \sqrt{(-2)^2 + 3^2} = \sqrt{4 + 9} = \sqrt{13}.$$

De lengte van de vector  $\mathbf{v}$  wordt op een meer wiskundige manier aangeduid met

$$\|\mathbf{v}\| \quad \text{of} \quad \|\mathbf{v}\|_2.$$

De tweede notatie (met de subscript 2) wordt gebruikt wanneer men heel expliciet wil zijn over de manier waarop de lengte werd bepaald.

We kunnen de formule voor lengte uitbreiden tot vectoren met een willekeurig aantal componenten.

**Definitie 1.15** De LENGTE van een vector in  $\mathbb{R}^n$  wordt gedefinieerd als de vierkantswortel van de som van de kwadraten van de componenten van de



vector:

$$\left\| \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \right\| = \sqrt{v_1^2 + v_2^2 + \cdots + v_n^2}. \quad (1.1)$$

**Opmerking 1.16** In sommige toepassingen, ook binnen machinaal leren, wordt de “lengte” van een vector soms op een andere manier bepaald. Zo is bv.

$$\|\mathbf{v}\|_1 = \left\| \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \right\|_1 = |v_1| + |v_2| + \cdots + |v_n|$$

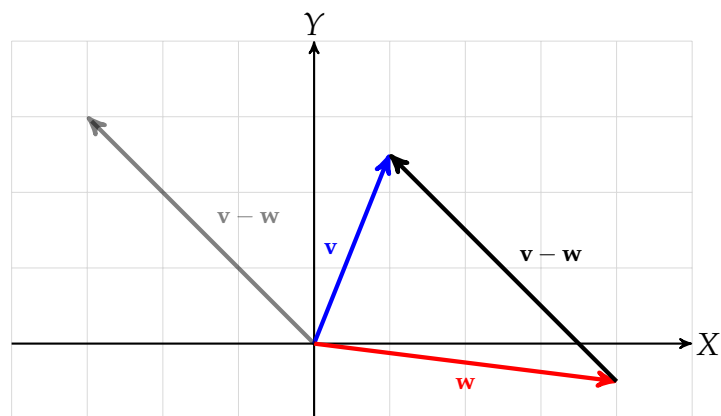
gelijk aan de som van de absolute waarden van de componenten. In dit geval gebruikt men een subscript 1 om deze manier van berekenen aan te geven.

### 1.4.2 Afstand tussen twee vectoren

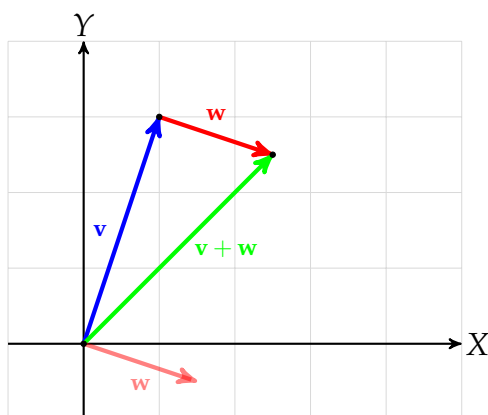
De lengte van een vector is de afstand van de oorsprong tot de kop van de vector. In het algemeen definiëren we de afstand tussen twee vectoren  $\mathbf{v}$  en  $\mathbf{w}$  als de afstand tussen de kop van de vector  $\mathbf{v}$  en de kop van de vector  $\mathbf{w}$ . In Figuur 1.5 zie je dat dit overeenkomt met de lengte van het verschil  $\mathbf{v} - \mathbf{w}$ .

In formulevorm:

$$\begin{aligned} \text{afstand}(\mathbf{v}, \mathbf{w}) &= \|\mathbf{v} - \mathbf{w}\| \\ &= \left\| \begin{bmatrix} v_1 - w_1 \\ v_2 - w_2 \\ \vdots \\ v_n - w_n \end{bmatrix} \right\| \\ &= \sqrt{(v_1 - w_1)^2 + (v_2 - w_2)^2 + \cdots + (v_n - w_n)^2}. \end{aligned}$$



**Figuur 1.5:** De afstand tussen de vectoren  $\mathbf{v}$  en  $\mathbf{w}$  wordt gegeven door de lengte van vector  $\mathbf{v} - \mathbf{w}$ .



**Figuur 1.6:** Een rechthoekige driehoek waarvan de lengtes van de zijden gegeven worden door  $\|\mathbf{v}\|$ ,  $\|\mathbf{w}\|$  en  $\|\mathbf{v} + \mathbf{w}\|$ . In deze driehoek is de stelling van Pythagoras geldig.

### 1.4.3 Orthogonale vectoren

We trachten nu uit te drukken dat twee vectoren orthogonaal zijn, i.e. dat ze loodrecht op elkaar staan. Bekijken we twee vectoren  $\mathbf{v}$  en  $\mathbf{w}$  dan kunnen we een driehoek herkennen waarvan de lengte van de drie zijden gegeven wordt door

$$\|\mathbf{v}\|, \quad \|\mathbf{w}\| \quad \text{en} \quad \|\mathbf{v} + \mathbf{w}\|.$$

Als de vectoren  $\mathbf{v}$  en  $\mathbf{w}$  loodrecht op elkaar staan dan zal deze driehoek een *rechthoekige* driehoek zijn waarin de stelling van Pythagoras geldig is, zie Figuur 1.6. Dit betekent dat

$$\|\mathbf{v} + \mathbf{w}\|^2 = \|\mathbf{v}\|^2 + \|\mathbf{w}\|^2.$$

Als we dit nu uitwerken m.b.v. formule (1.1) en onthouden dat

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} \quad \text{en} \quad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix}$$

dan vinden we

$$\begin{aligned} (v_1 + w_1)^2 + (v_2 + w_2)^2 &= (v_1^2 + v_2^2) + (w_1^2 + w_2^2) \\ \iff (v_1^2 + 2v_1w_1 + w_1^2) + (v_2^2 + 2v_2w_2 + w_2^2) &= (v_1^2 + v_2^2) + (w_1^2 + w_2^2) \\ \iff v_1w_1 + v_2w_2 &= 0. \end{aligned}$$

Dezelfde berekening kan gedaan worden voor vectoren met  $n$  componenten. Twee vectoren zullen dan orthogonaal zijn als

$$v_1w_1 + v_2w_2 + \cdots + v_nw_n = 0.$$

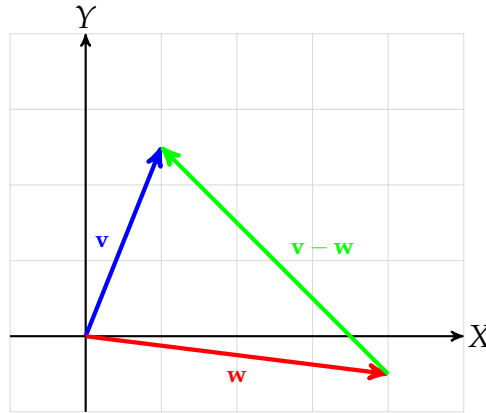
Dit suggereert dat het zinvol is om een nieuw “product” van vectoren als volgt te definiëren:

$$\mathbf{v} \cdot \mathbf{w} = v_1w_1 + v_2w_2 + \cdots + v_nw_n,$$

met

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} \quad \text{en} \quad \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}.$$

Dit INWENDIG PRODUCT neemt twee vectoren als invoer en produceert een reëel getal als uitvoer.



**Figuur 1.7:** Een willekeurige driehoek waarvan de lengtes van de zijden gegeven worden door  $\|\mathbf{v}\|$ ,  $\|\mathbf{w}\|$  en  $\|\mathbf{v} - \mathbf{w}\|$ .

#### 1.4.4 Betekenis van het inwendig product

Daarnet bekeken we een rechthoekige driehoek, nu bekijken we een willekeurige driehoek met zijden

$$\|\mathbf{v}\|, \quad \|\mathbf{w}\| \quad \text{en} \quad \|\mathbf{v} - \mathbf{w}\|.$$

Zo'n driehoek wordt geïllustreerd in Figuur 1.7.

**Opmerking 1.17 (Cosinusregel)** De stelling van Pythagoras is geldig in rechthoekige driehoeken en zegt dat

$$C^2 = A^2 + B^2,$$

waarbij  $C$  de lengte van de schuine zijde en  $A$  en  $B$  de lengtes van de rechthoekszijden voorstellen. Merk op dat de overstaande hoek van de schuine zijde de rechte hoek is.

In een willekeurige driehoek geldt echter dat

$$C^2 = A^2 + B^2 - 2AB \cos(\hat{c})$$

waarbij  $\hat{c}$  de overstaande hoek van de zijde (met lengte)  $C$  voorstelt. Deze regel wordt de COSINUSREGEL genoemd. Wanneer  $\hat{c}$  gelijk is aan 90 graden, dan reduceert de cosinusregel tot de stelling van Pythagoras. ■

We passen de cosinusregel toe op de driehoek in Figuur 1.7 en we krijgen

$$\|\mathbf{v} - \mathbf{w}\|^2 = \|\mathbf{v}\|^2 + \|\mathbf{w}\|^2 - 2\|\mathbf{v}\|\|\mathbf{w}\|\cos(\theta), \quad (1.2)$$

waarbij  $\theta$  de ingesloten hoek tussen  $\mathbf{v}$  en  $\mathbf{w}$  voorstelt. Aan de andere kant kunnen we  $\|\mathbf{v} - \mathbf{w}\|^2$  ook berekenen met de definitie van lengte van vectoren:

$$\begin{aligned}\|\mathbf{v} - \mathbf{w}\|^2 &= (v_1 - w_1)^2 + (v_2 - w_2)^2 \\ &= v_1^2 - 2v_1w_1 + w_1^2 + v_2^2 - 2v_2w_2 + w_2^2 \\ &= (v_1^2 + v_2^2) + (w_1^2 + w_2^2) - 2(v_1w_1 + v_2w_2) \\ &= \|\mathbf{v}\|^2 + \|\mathbf{w}\|^2 - 2\mathbf{v} \cdot \mathbf{w}.\end{aligned}\tag{1.3}$$

Door deze laatste uitdrukking gelijk te stellen aan het rechterlid van formule (1.2) vinden we dat

$$\mathbf{v} \cdot \mathbf{w} = \|\mathbf{v}\| \|\mathbf{w}\| \cos(\theta).\tag{1.4}$$

**Opmerking 1.18** Wanneer  $\mathbf{v}$  en  $\mathbf{w}$  gelijk zijn, dan is  $\theta$  gelijk aan 0 graden en  $\cos(\theta) = 1$ , zodat

$$\mathbf{v} \cdot \mathbf{v} = \|\mathbf{v}\|^2.$$

Wanneer  $\mathbf{v}$  en  $\mathbf{w}$  loodrecht staan, dan is  $\theta$  een rechte hoek waarvoor de cosinus gelijk is aan nul. In dit geval is dus

$$\mathbf{v} \cdot \mathbf{w} = 0.$$

Wanneer  $\theta$  een hoek maakt tussen 90 en 270 graden, i.e. wanneer  $\mathbf{v}$  en  $\mathbf{w}$  in (min of meer) tegenstelde richtingen wijzen dan zal het inwendig product negatief zijn. ■

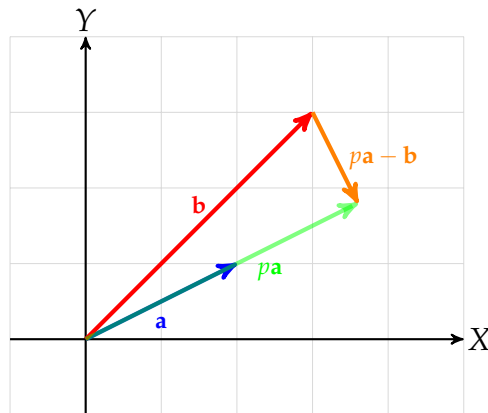
**Eigenschap 1.19** Het is niet moeilijk om te verifiëren dat het inwendig product aan de volgende eigenschappen voldoet.

1. Het inwendig product is commutatief:  $\mathbf{v} \cdot \mathbf{w} = \mathbf{w} \cdot \mathbf{v}$  voor alle vectoren  $\mathbf{v}$  en  $\mathbf{w}$ .
2. Het inwendig product is lineair in de tweede component:

$$\mathbf{u} \cdot (\alpha \mathbf{v} + \beta \mathbf{w}) = \alpha(\mathbf{u} \cdot \mathbf{v}) + \beta(\mathbf{u} \cdot \mathbf{w})$$

voor alle vectoren  $\mathbf{u}$ ,  $\mathbf{v}$  en  $\mathbf{w}$  en alle reële getallen  $\alpha$  en  $\beta$ . ■

*Bewijs* Dit volgt uit een eenvoudige berekening. ◇



**Figuur 1.8:** Projectie van de vector  $\mathbf{b}$  op de vector  $\mathbf{a}$ , of juister op de lineaire deelruimte opgespannen door  $\mathbf{a}$ .

**Opmerking 1.20** Het inwendig product is *niet* associatief

$$\mathbf{u} \cdot (\mathbf{v} \cdot \mathbf{w}) \neq (\mathbf{u} \cdot \mathbf{v}) \cdot \mathbf{w}.$$

Het linkerlid is immers een vector in de richting van  $\mathbf{u}$ , terwijl het rechterlid een vector is in de richting van  $\mathbf{w}$ . ■

## 1.5 Projectie op een vector

Veronderstel dat twee vectoren  $\mathbf{a}$  en  $\mathbf{b}$  gegeven zijn. We wensen een vector te vinden in de richting van  $\mathbf{a}$  waarvoor de afstand tot  $\mathbf{b}$ , zoals gedefinieerd in sectie 1.4.2, zo klein mogelijk is. Het is intuïtief duidelijk dat deze vector de loodrechte projectie moet zijn van  $\mathbf{b}$  op de richting bepaald door  $\mathbf{a}$ . M.a.w. als we de projectie aanduiden als

$$p\mathbf{a},$$

waarbij  $p \in \mathbb{R}$  het onbekende getal is dat we moeten vinden, dan wordt het getal  $p$  bepaald door het feit dat

$$(p\mathbf{a} - \mathbf{b}) \text{ en } \mathbf{a}$$

orthogonaal zijn, zie Figuur 1.8 Als we dit uitdrukken m.b.v. het inwendig product dan krijgen we

$$0 = \mathbf{a} \cdot (p\mathbf{a} - \mathbf{b}) = p\mathbf{a} \cdot \mathbf{a} - \mathbf{a} \cdot \mathbf{b},$$

of dus

$$p = \frac{\mathbf{a} \cdot \mathbf{b}}{\mathbf{a} \cdot \mathbf{a}}.$$

We hebben hierbij de volgende eigenschap bewezen.

**Eigenschap 1.21** De loodrechte projectie van  $\mathbf{b}$  op  $\mathbf{a}$  wordt gegeven door de vector

$$\frac{\mathbf{a} \cdot \mathbf{b}}{\mathbf{a} \cdot \mathbf{a}} \mathbf{a}. \quad \blacksquare$$

**Opmerking 1.22 (Bijzondere gevallen loodrechte projectie)** Als  $\mathbf{b}$  loodrecht staat op  $\mathbf{a}$  dan is de projectie de nulvector. Aan de andere kant, als  $\mathbf{b}$  reeds een veelvoud was van  $\mathbf{a}$  dan is de projectie van  $\mathbf{b}$  gelijk aan  $\mathbf{b}$  zelf.  $\blacksquare$

**Voorbeeld 1.23 (Voorbeeld loodrechte projectie)** Veronderstel dat de vectoren  $\mathbf{a}$  en  $\mathbf{b}$  gegeven worden door:

$$\mathbf{a} = \begin{bmatrix} 2 \\ 1 \end{bmatrix} \quad \text{en} \quad \mathbf{b} = \begin{bmatrix} 3 \\ 3 \end{bmatrix}.$$

De loodrechte projectie van  $\mathbf{b}$  op de vectorruimte opgespannen door  $\mathbf{a}$  wordt volgens Eigenschap 1.21 gegeven door

$$\frac{\mathbf{a} \cdot \mathbf{b}}{\mathbf{a} \cdot \mathbf{a}} \mathbf{a} = \frac{6+3}{4+1} \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 18/5 \\ 9/5 \end{bmatrix}.$$

Deze situatie wordt geïllustreerd in Figuur 1.8.  $\blacksquare$

## 1.6 Oefeningen

1. Welke verzamelingen vectoren zijn lineair onafhankelijk?

a)

$$\mathbf{a} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \text{en} \quad \mathbf{b} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

b)

$$\mathbf{a} = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \text{en} \quad \mathbf{b} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

c)

$$\mathbf{a} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \quad \text{en} \quad \mathbf{b} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \quad \text{en} \quad \mathbf{c} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

d)

$$\mathbf{a} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \quad \text{en} \quad \mathbf{b} = \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix} \quad \text{en} \quad \mathbf{c} = \begin{bmatrix} -3 \\ 1 \\ -2 \end{bmatrix}$$

2. Gegeven de volgende drie vectoren met coördinaten t.o.v. de standaardbasis.

$$\mathbf{v} = \begin{bmatrix} 5 \\ -1 \end{bmatrix} \quad \text{en} \quad \mathbf{b}_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \text{en} \quad \mathbf{b}_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Gevraagd: geef de coördinaten van  $\mathbf{v}$  t.o.v. de basis bepaald door  $\mathbf{b}_1$  en  $\mathbf{b}_2$ .

3. Vormt de verzameling vectoren van de vorm

$$\begin{bmatrix} \alpha \\ 1 \\ \alpha \end{bmatrix} \quad \text{met} \quad \alpha \in \mathbb{R}$$

een lineaire deelruimte van  $\mathbb{R}^3$ ? Verklaar je antwoord.

4. Geef de loodrechte projectie van de volgende vectoren  $\mathbf{b}$  op  $\mathbf{a}$ .

a)  $\mathbf{a} = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}$  en  $\mathbf{b} = \begin{bmatrix} 3 \\ 4 \\ -1 \end{bmatrix}$

b)  $\mathbf{a} = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}$  en  $\mathbf{b} = \begin{bmatrix} -2 \\ 0 \\ -4 \end{bmatrix}$

c)  $\mathbf{a} = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}$  en  $\mathbf{b} = \begin{bmatrix} 4 \\ -3 \\ -2 \end{bmatrix}$



5. `Numpy` In machinaal leren, bv. in de context van aanbevelingsystemen (Eng. *recommendation systems*) zoekt men dikwijls *gelijkaardige vectoren*. Een manier om dit te doen is te kijken naar de hoek die twee vectoren maken, en meer bepaald naar hun cosinus. Dit noemt men dan de COSINUS SIMILARITEIT (Eng. *cosine similarity*) van de twee vectoren, waarbij een waarde van  $+1$  betekent dat de twee vectoren maximaal gelijkaardig zijn, terwijl een waarde van  $-1$  betekent dat ze maximaal ongelijkaardig zijn.

Schrijf een methode `cosinus_similariteit` die twee (even lange) vectoren als invoer heeft en hun cosinus similariteit berekent. Maak gebruik van ingebouwde `numpy` methoden om het inwendig product en de lengte van vectoren te berekenen.

Pas je methode toe op een aantal vectoren om de correcte werking te verifiëren.

6. `Numpy` Gebruik `numpy` om de eigenschappen van het inwendig product (commutativiteit en lineair in de tweede component) te verifiëren voor een aantal willekeurig gegenereerde vectoren.
7. Veronderstel dat  $\mathbf{u}$  en  $\mathbf{v}$  twee orthogonale vectoren zijn (bv. in  $\mathbb{R}^n$ ) en dat  $\mathbf{u}$  en  $\mathbf{v}$  allebei lengte 1 hebben. Toon aan dat de vectoren  $\mathbf{u} + \mathbf{v}$  en  $\mathbf{u} - \mathbf{v}$  orthogonaal zijn. Wat is hun lengte?

## Matrices

In dit hoofdstuk introduceren we het begrip matrix. Om de definitie van het matrixproduct te onderbouwen bespreken we eerst kort het begrip lineaire transformatie, waaruit de definitie van het matrixproduct dan volgt.

### 2.1 Lineaire transformaties en matrixproduct

Een *transformatie* is een ander woord voor een functie. Een transformatie neemt m.a.w. een argument als invoer en produceert een overeenkomstige uitvoer. In de context van lineaire algebra is de invoer een vector en is de uitvoer eveneens een vector. Zo kan een transformatie  $L$  de vector

$$\begin{bmatrix} 5 \\ 7 \end{bmatrix}$$

afbeelden op

$$\begin{bmatrix} 2 \\ 3 \end{bmatrix}.$$

Symbolisch schrijven we

$$L\left(\begin{bmatrix} 5 \\ 7 \end{bmatrix}\right) = \begin{bmatrix} 2 \\ 3 \end{bmatrix}.$$

Een transformatie is een LINEAIRE TRANSFORMATIE wanneer die rechte lijnen transformeert in (andere) rechte lijnen. Wiskundig komt dit neer op de volgende twee eigenschappen:

- Het beeld van een som is de som van de beelden:

$$L(\mathbf{v} + \mathbf{w}) = L(\mathbf{v}) + L(\mathbf{w}).$$

- Wanneer men een vector schaalt, dan schaalt het beeld op precies dezelfde manier.

$$L(\alpha \mathbf{v}) = \alpha L(\mathbf{v}).$$

De vector

$$\mathbf{v} = \begin{bmatrix} 5 \\ 7 \end{bmatrix}$$

kunnen we ook schrijven als de volgende (unieke) lineaire combinatie van  $\mathbf{i}$  en  $\mathbf{j}$ :

$$\mathbf{v} = 5\mathbf{i} + 7\mathbf{j},$$

waarbij  $\mathbf{i}$  en  $\mathbf{j}$ , zoals voorheen, gegeven worden door:

$$\mathbf{i} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \quad \text{en} \quad \mathbf{j} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

Als de transformatie  $L$  lineair is, dan is het beeld van  $\mathbf{v}$  onder  $L$  *dezelfde lineaire combinatie* van de beelden van  $\mathbf{i}$  en  $\mathbf{j}$ ! In symbolen:

$$L(\mathbf{v}) = L(5\mathbf{i} + 7\mathbf{j}) = L(5\mathbf{i}) + L(7\mathbf{j}) = 5L(\mathbf{i}) + 7L(\mathbf{j}).$$

M.a.w. als je weet wat het beeld is van  $\mathbf{i}$  en  $\mathbf{j}$ , dan ken je het beeld van *alle* vectoren onder deze lineaire transformatie.

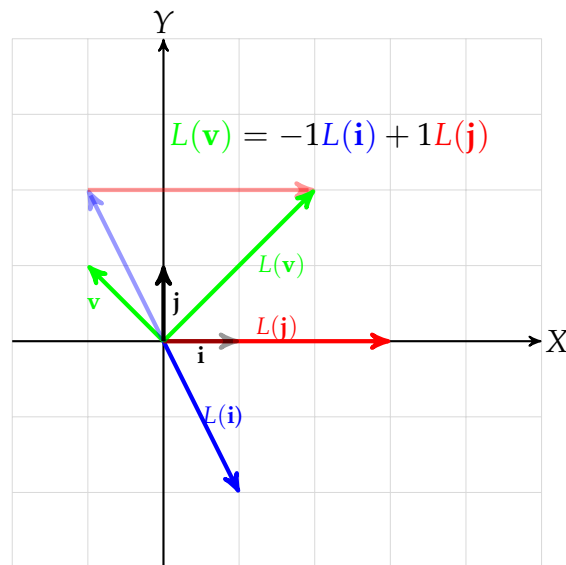
Als we (computer)berekeningen willen uitvoeren met lineaire transformaties, dan moeten we een manier hebben om zo'n lineaire transformatie voor te stellen m.b.v. getallen. De zonet gemaakte observatie dat het voldoende is om te weten wat het beeld is van de basisvectoren  $\mathbf{i}$  en  $\mathbf{j}$  suggereert om enkel deze beelden op te slaan.

Bekijk de lineaire transformatie zoals voorgesteld in Figuur 2.1. Uit deze figuur lezen we af dat

$$L(\mathbf{i}) = \begin{bmatrix} 1 \\ -2 \end{bmatrix} \quad \text{en} \quad L(\mathbf{j}) = \begin{bmatrix} 3 \\ 0 \end{bmatrix}.$$

M.b.v. deze informatie (i.e. met behulp van deze 4 getallen) kunnen we berekenen wat het beeld is van de vector

$$\mathbf{v} = \begin{bmatrix} -1 \\ 1 \end{bmatrix}.$$



**Figuur 2.1:** Voorbeeld van een lineaire transformatie.

Dit gaat als volgt:

$$\begin{aligned}
 L(\mathbf{v}) &= L(-1 \mathbf{i} + 1 \mathbf{j}) && \mathbf{v} \text{ schrijven als lineaire combinatie} \\
 &= -1L(\mathbf{i}) + 1L(\mathbf{j}) && L \text{ is een lineaire transformatie} \\
 &= -1 \begin{bmatrix} 1 \\ -2 \end{bmatrix} + 1 \begin{bmatrix} 3 \\ 0 \end{bmatrix} && \text{invullen gekende beelden} \\
 &= \begin{bmatrix} 2 \\ 2 \end{bmatrix}.
 \end{aligned}$$

Beschouw nu een willekeurige vector

$$\begin{bmatrix} x \\ y \end{bmatrix}.$$

Wat is het beeld van deze vector onder de transformatie  $L$ ? We volgen het-

zelfde stramien als bij het voorbeeld:

$$\begin{aligned}
 L\left(\begin{bmatrix} x \\ y \end{bmatrix}\right) &= L(x\mathbf{i} + y\mathbf{j}) && \text{vector schrijven als lineaire combinatie} \\
 &= xL(\mathbf{i}) + yL(\mathbf{j}) && L \text{ is een lineaire transformatie} \\
 &= x\begin{bmatrix} 1 \\ -2 \end{bmatrix} + y\begin{bmatrix} 3 \\ 0 \end{bmatrix} && \text{invullen gekende beelden} \\
 &= \begin{bmatrix} 1x + 3y \\ -2x + 0y \end{bmatrix}.
 \end{aligned}$$

### 2.1.1 Matrix-vector product

We kunnen de beelden van  $\mathbf{i}$  en  $\mathbf{j}$  schrijven als de kolommen van een 2 bij 2 matrix:

$$\begin{bmatrix} 1 & 3 \\ -2 & 0 \end{bmatrix}.$$

Om dan het beeld te berekenen van de vector

$$\begin{bmatrix} x \\ y \end{bmatrix}$$

onder de lineaire transformatie nemen we  $x$  keer de eerste kolom van deze matrix en tellen daar  $y$  keer de tweede kolom van deze matrix bij op, zoals hierboven geïllustreerd. Dit beeld noteren we als het matrix-vector product

$$\begin{bmatrix} 1 & 3 \\ -2 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = x \begin{bmatrix} 1 \\ -2 \end{bmatrix} + y \begin{bmatrix} 3 \\ 0 \end{bmatrix}.$$

In het algemeen kunnen we dus schrijven

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = x \begin{bmatrix} a \\ c \end{bmatrix} + y \begin{bmatrix} b \\ d \end{bmatrix} = \begin{bmatrix} ax + by \\ cx + dy \end{bmatrix}.$$

In dit voorbeeld ging de lineaire transformatie van  $\mathbb{R}^2$  naar  $\mathbb{R}^2$ , maar in het algemeen kan een lineaire transformatie van  $\mathbb{R}^n$  naar  $\mathbb{R}^m$  gaan.

Beschouw bv. een lineaire transformatie  $L$  van  $\mathbb{R}^3$  naar  $\mathbb{R}^2$  die de drie basisvectoren  $\mathbf{i}$ ,  $\mathbf{j}$  en  $\mathbf{k}$  afbeeldt op:

$$L(\mathbf{i}) = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, \quad L(\mathbf{j}) = \begin{bmatrix} 1 \\ 1 \end{bmatrix} \quad \text{en} \quad L(\mathbf{k}) = \begin{bmatrix} 0 \\ -1 \end{bmatrix}.$$

Als we dan willen weten wat het beeld is van een willekeurige vector

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

van  $\mathbb{R}^3$  onder  $L$ , dan berekenen we het volgende matrix-vector product

$$\begin{bmatrix} 2 & 1 & 0 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = x \begin{bmatrix} 2 \\ 0 \end{bmatrix} + y \begin{bmatrix} 1 \\ 1 \end{bmatrix} + z \begin{bmatrix} 0 \\ -1 \end{bmatrix} = \begin{bmatrix} 2x + 1y + 0z \\ 0x + 1y - z \end{bmatrix}. \quad (2.1)$$

**Definitie 2.1** Een MATRIX-VECTORPRODUCT tussen een matrix  $\mathbf{A}$  en een vector  $\mathbf{v}$  kan berekend worden als het aantal kolommen van  $\mathbf{A}$  gelijk is aan het aantal componenten van  $\mathbf{v}$  en is een vector die gelijk is aan *de lineaire combinatie van de kolommen van  $\mathbf{A}$*  waarbij de coëfficiënten van de lineaire combinatie gegeven worden door de componenten van  $\mathbf{v}$ . In symbolen:

$$\mathbf{A}\mathbf{v} = \begin{bmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \\ | & | & & | \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix} = v_1 \begin{bmatrix} | \\ \mathbf{a}_1 \\ | \end{bmatrix} + v_2 \begin{bmatrix} | \\ \mathbf{a}_2 \\ | \end{bmatrix} + \cdots + v_n \begin{bmatrix} | \\ \mathbf{a}_n \\ | \end{bmatrix}. \quad (2.2) \quad \blacksquare$$

### 2.1.2 Samenstelling van lineaire transformaties

In de wiskunde en informatica worden functies of transformaties vaak *samengesteld*, d.w.z. dat de uitvoer van de ene transformatie de invoer van de andere wordt.

Beschouw bv. de volgende twee transformaties. Een rotatie  $R$  over 90 graden in tegenwijzerzin, die de vector  $\mathbf{i}$  afbeeldt op  $\mathbf{j}$ , terwijl  $\mathbf{j}$  wordt afgebeeld op  $-\mathbf{i}$ . De matrix die hiermee overeenstemt is m.a.w.

$$\mathbf{R} = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}.$$

De tweede transformatie die we beschouwen is een “scheeftrekking”  $S$  die  $\mathbf{i}$  afbeeldt op zichzelf en  $\mathbf{j}$  afbeeldt op  $\mathbf{i} + \mathbf{j}$ . De matrix die hiermee overeenstemt is

$$\mathbf{S} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}.$$

Stel dat we nu willen uitvissen waar de vector

$$\begin{bmatrix} x \\ y \end{bmatrix}$$

terecht komt als we eerst de rotatie  $R$  uitvoeren en daarna de scheeftrekking  $S$ ? Na uitvoeren van de rotatie krijgen we:

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} -y \\ x \end{bmatrix}.$$

Als we hierop nu de scheeftrekking  $S$  toepassen, dan krijgen we

$$\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} -y \\ x \end{bmatrix} = \begin{bmatrix} x - y \\ x + 0y \end{bmatrix} = x \begin{bmatrix} 1 \\ 1 \end{bmatrix} + y \begin{bmatrix} -1 \\ 0 \end{bmatrix}.$$

We kunnen dit laatste nu herkennen als een lineaire transformatie die  $\mathbf{i}$  afbeeldt op  $\mathbf{i} + \mathbf{j}$  en die  $\mathbf{j}$  afbeeldt op  $-\mathbf{i}$ .

We schrijven dit als

$$\mathbf{SR} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 1 & 0 \end{bmatrix}.$$

Deze manier om de nieuwe matrix te bepalen die de lineaire transformatie voorstelt noemen we het matrixproduct en is in feite niets anders dan het tweemaal toepassen van het matrix-vectorproduct, één keer voor elke kolom van de tweede matrix. Elk matrix-vectorproduct wordt een kolom in de resulterende matrix.

**Voorbeeld 2.2** Veronderstel dat  $\mathbf{M}_1$  en  $\mathbf{M}_2$  twee matrices zijn die elk overeen komen met een lineaire transformatie:

$$\mathbf{M}_1 = \begin{bmatrix} 1 & -2 \\ 1 & 0 \end{bmatrix} \quad \text{en} \quad \mathbf{M}_2 = \begin{bmatrix} 0 & 2 \\ 1 & 0 \end{bmatrix}.$$

Vind de matrix die hoort bij de lineaire transformatie wanneer men eerst de transformatie horend bij  $\mathbf{M}_1$  toepast en daarna de transformatie die hoort bij  $\mathbf{M}_2$ .

We hebben zonet gezien dat de matrix die we zoeken gegeven wordt door het matrixproduct

$$\mathbf{M}_2\mathbf{M}_1 = \begin{bmatrix} 0 & 2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & -2 \\ 1 & 0 \end{bmatrix}.$$

De eerste kolom van dit matrixproduct wordt gegeven door

$$\begin{bmatrix} 0 & 2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} = 1 \begin{bmatrix} 0 \\ 1 \end{bmatrix} + 1 \begin{bmatrix} 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 2 \\ 1 \end{bmatrix}.$$

De tweede kolom wordt gegeven door

$$\begin{bmatrix} 0 & 2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} -2 \\ 0 \end{bmatrix} = -2 \begin{bmatrix} 0 \\ 1 \end{bmatrix} + 0 \begin{bmatrix} 2 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ -2 \end{bmatrix}.$$

Het resulterende matrixproduct is

$$\mathbf{M}_2 \mathbf{M}_1 = \begin{bmatrix} 0 & 2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & -2 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 1 & -2 \end{bmatrix}. \quad \blacksquare$$

### 2.1.3 Het matrixproduct

De meeste onder jullie hebben het matrixproduct wellicht gezien als een relatief ingewikkelde formule die je uit het hoofd moest leren. Door het inzicht dat het matrixproduct overeenkomt met het samenstellen van lineaire transformaties heb je nu wat achtergrond.

Laat ons nu in het algemeen bekijken hoe we twee lineaire transformaties, allebei van  $\mathbb{R}^2$  naar  $\mathbb{R}^2$  kunnen samenstellen om een formule te bekomen voor het product van willekeurige twee bij twee matrices. Stel

$$\mathbf{M}_2 = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \quad \text{en} \quad \mathbf{M}_1 = \begin{bmatrix} e & f \\ g & h \end{bmatrix}.$$

Vind het product

$$\mathbf{M}_2 \mathbf{M}_1 = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix}.$$

Om te volgen waar **i** terechtkomt onder de samengestelde transformatie nemen we in essentie het matrix-vectorproduct van  $\mathbf{M}_2$  met de eerste kolom van  $\mathbf{M}_1$ :

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e \\ g \end{bmatrix} = e \begin{bmatrix} a \\ c \end{bmatrix} + g \begin{bmatrix} b \\ d \end{bmatrix} = \begin{bmatrix} ae + bg \\ ce + dg \end{bmatrix}.$$

Voor de vector **j** volgen we hetzelfde proces, maar nu uiteraard met de tweede kolom van  $\mathbf{M}_1$ :

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} f \\ h \end{bmatrix} = f \begin{bmatrix} a \\ c \end{bmatrix} + h \begin{bmatrix} b \\ d \end{bmatrix} = \begin{bmatrix} af + bh \\ cf + dh \end{bmatrix}.$$



Dit betekent dat het resulterende matrixproduct gegeven wordt door

$$\mathbf{M}_2 \mathbf{M}_1 = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix}.$$

Deze resultaten m.b.t. het matrixproduct worden op de natuurlijke manier uitgebreid naar vectoren behorend tot  $\mathbb{R}^n$ , en zelfs tot vectoren met een verschillend aantal componenten, op voorwaarde dat *de uitvoer van de eerste transformatie evenveel componenten heeft als de invoer van de tweede transformatie*. M.a.w. als  $T_1$  een lineaire transformatie is van  $\mathbb{R}^n$  naar  $\mathbb{R}^m$  en als  $T_2$  een lineaire transformatie is van  $\mathbb{R}^m$  naar  $\mathbb{R}^p$ , dan kan men de samengestelde transformatie bepalen waarbij men eerst  $T_1$  uitvoert, gevolgd door  $T_2$ . De matrix die  $T_1$  bepaalt heeft  $n$  kolommen (want je moet beschrijven wat er met elk van de basisvectoren in de invoer gebeurt) en  $m$  rijen (want elke uitvoer heeft  $m$  componenten). De matrix die  $T_2$  bepaalt heeft  $m$  kolommen en  $p$  rijen. De matrix die hoort bij de samenstelling van de lineaire transformaties heeft  $n$  kolommen en  $p$  rijen. Symbolisch wordt dit:

$$(\mathbf{M}_2)_{p \times m} (\mathbf{M}_1)_{m \times n} \rightarrow (\mathbf{M}_2 \mathbf{M}_1)_{p \times n}.$$

Om dit product te berekenen zeggen we dat de  $i$ -de kolom van het resultaat gegeven wordt door het matrix-vectorproduct van  $\mathbf{M}_2$  en de  $i$ -de kolom van  $\mathbf{M}_1$ .

**Voorbeeld 2.3 (Product van twee 3 bij 3 matrices)** We berekenen het matrixproduct van twee 3 bij 3 matrices  $\mathbf{A}$  en  $\mathbf{B}$ :

$$\mathbf{A} = \begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{bmatrix} \quad \text{en} \quad \mathbf{B} = \begin{bmatrix} B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,1} & B_{3,2} & B_{3,3} \end{bmatrix}.$$

De eerste kolom van het matrixproduct  $\mathbf{AB}$  wordt gegeven door het volgende matrix-vectorproduct:

$$\begin{aligned} \begin{bmatrix} A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,1} & A_{3,2} & A_{3,3} \end{bmatrix} \begin{bmatrix} B_{1,1} \\ B_{2,1} \\ B_{3,1} \end{bmatrix} &= B_{1,1} \begin{bmatrix} A_{1,1} \\ A_{2,1} \\ A_{3,1} \end{bmatrix} + B_{2,1} \begin{bmatrix} A_{1,2} \\ A_{2,2} \\ A_{3,2} \end{bmatrix} + B_{3,1} \begin{bmatrix} A_{1,3} \\ A_{2,3} \\ A_{3,3} \end{bmatrix} \\ &= \begin{bmatrix} A_{1,1}B_{1,1} + A_{1,2}B_{2,1} + A_{1,3}B_{3,1} \\ A_{2,1}B_{1,1} + A_{2,2}B_{2,1} + A_{2,3}B_{3,1} \\ A_{3,1}B_{1,1} + A_{3,2}B_{2,1} + A_{3,3}B_{3,1} \end{bmatrix}. \end{aligned}$$

De formule voor de volgende twee kolommen is volledig analoog: men dient enkel de kolomindex voor  $\mathbf{B}$  te vervangen door de gepaste kolom.

Hieruit herkennen we dan de “klassieke” formule voor het matrixproduct. Als  $\mathbf{AB} = \mathbf{C}$  dan is

$$C_{i,j} = A_{i,1}B_{1,j} + A_{i,2}B_{2,j} + A_{i,3}B_{3,j}. \quad \blacksquare$$

**Opmerking 2.4** In de praktijk worden matrixproducten berekend m.b.v. software zoals Numpy. ■

**Eigenschap 2.5** Het matrixproduct is associatief. Dit betekent dat

$$\mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C}.$$

voor alle matrices waarvoor de uitdrukking is gedefinieerd. ■

*Bewijs* Dit volgt onmiddellijk uit het feit dat de samenstelling van lineaire transformaties associatief is. Men kan dit ook expliciet narekenen a.d.h.v. een eenvoudige maar relatief saaie berekening. ◇

**Opmerking 2.6 (Matrixproduct is niet-commutatief)** Het is uiterst belangrijk om te weten dat in het algemeen

$$\mathbf{AB} \neq \mathbf{BA},$$

zelfs al zijn beide leden gedefinieerd. ■

**Voorbeeld 2.7 (Voorbeeld niet-commutativiteit)** Daarnet zagen we dat

$$\mathbf{M}_2\mathbf{M}_1 = \begin{bmatrix} 0 & 2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & -2 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 2 & 0 \\ 1 & -2 \end{bmatrix}.$$

Als we het matrixproduct in de andere volgorde uitvoeren dan krijgen we

$$\mathbf{M}_1\mathbf{M}_2 = \begin{bmatrix} 1 & -2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 2 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} -2 & 2 \\ 0 & 2 \end{bmatrix}.$$

Dit geeft duidelijk een verschillend resultaat. ■

## 2.2 Transponeren van een matrix

Alhoewel we matrices hebben geïntroduceerd als een manier om het effect van lineaire transformaties te noteren wordt een matrix in de praktijk vaak eenvoudigweg beschouwd als een manier om een groot aantal datapunten (getallen) op een eenvoudige manier te verzamelen en te manipuleren.

Elke matrix kan getransponeerd worden, d.w.z. dat de rol van de rijen en kolommen wordt omgewisseld. De getransponeerde van de matrix  $\mathbf{A}$  wordt aangeduid met  $\mathbf{A}^T$ . Wanneer men een matrix twee keer transponeert, dan komt men uiteraard terug uit bij de originele matrix:

$$(\mathbf{A}^T)^T = \mathbf{A}.$$

**Voorbeeld 2.8 (Voorbeeld transponeren)** Veronderstel dat de matrix  $\mathbf{A}$  gelijk is aan

$$\mathbf{A} = \begin{bmatrix} 2 & 1 & 0 \\ 0 & 1 & -1 \end{bmatrix}$$

dan geldt

$$\mathbf{A}^T = \begin{bmatrix} 2 & 0 \\ 1 & 1 \\ 0 & -1 \end{bmatrix}.$$

Als we de matrix  $\mathbf{A}^T$  transponeren, dan vinden we (uiteraard) opnieuw de matrix  $\mathbf{A}$ :

$$(\mathbf{A}^T)^T = \begin{bmatrix} 2 & 1 & 0 \\ 0 & 1 & -1 \end{bmatrix} = \mathbf{A}. \quad \blacksquare$$

Sommige matrices wijzigen niet als we ze transponeren, deze worden SYMMETRISCHE MATRICES genoemd. Een symmetrische matrix heeft uiteraard steeds evenveel rijen als kolommen. Een matrix met evenveel rijen als kolommen noemen we een VIERKANTE MATRIX.

**Voorbeeld 2.9** De volgende matrix is een symmetrische matrix:

$$\mathbf{A} = \begin{bmatrix} 5 & 1 \\ 1 & 2 \end{bmatrix}.$$

Er geldt

$$\mathbf{A}^T = \begin{bmatrix} 5 & 1 \\ 1 & 2 \end{bmatrix} = \mathbf{A}. \quad \blacksquare$$

### 2.2.1 Inwendig product als matrixproduct

De manier waarop we vectoren hebben genoteerd suggereert dat we deze ook kunnen beschouwen als matrices met één kolom. De getransponeerde van zo'n matrix met één kolom is dan een matrix met één enkele rij. Het inwendig product van twee vectoren kan ook gezien worden als het matrixproduct van een rijmatrix en een kolommatrix, waarbij we dan het feit "negeren" dat het inwendig product een getal als resultaat heeft terwijl het matrixproduct eigenlijk in een matrix resulteert met één rij en één kolom.

**Voorbeeld 2.10** Beschouw twee vectoren

$$\mathbf{a} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \quad \text{en} \quad \mathbf{b} = \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix}.$$

Dan weten we dat

$$\mathbf{a} \cdot \mathbf{b} = 1 \times 2 + 0 \times (-1) + 1 \times 1 = 3.$$

Bekijken we

$$\mathbf{a}^T \mathbf{b} = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 2 \\ -1 \\ 1 \end{bmatrix} = \begin{bmatrix} 3 \end{bmatrix},$$

dan vinden we in essentie hetzelfde resultaat. ■

Op die manier kan een matrix-vectorproduct ook gezien worden als het berekenen van de inwendige producten tussen de rijen van de matrix en de vector.

**Voorbeeld 2.11** Beschouw het matrix-vectorproduct dat we reeds voordien zagen

$$\begin{bmatrix} 2 & 1 & 0 \\ 0 & 1 & -1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}.$$

Als we nu twee vectoren definiëren, één voor elke rij van de matrix

$$\mathbf{a}_1 = \begin{bmatrix} 2 \\ 1 \\ 0 \end{bmatrix} \quad \text{en} \quad \mathbf{a}_2 = \begin{bmatrix} 0 \\ 1 \\ -1 \end{bmatrix}$$

en als we ook de vector een naam geven:

$$\mathbf{v} = \begin{bmatrix} x \\ y \\ z \end{bmatrix},$$

dan wordt de eerste (respectievelijk tweede) component van het resultaat gegeven door

$$\mathbf{a}_1 \cdot \mathbf{v} = 2x + y \quad \text{resp.} \quad \mathbf{a}_2 \cdot \mathbf{v} = y - z,$$

zoals we ook reeds zagen in vergelijking (2.1). ■

## 2.3 Bewerkingen en eigenschappen van matrices

Het optellen van matrices is mogelijk als ze dezelfde vorm hebben: in dit geval neemt men eenvoudigweg de som van de elementen op dezelfde plaats.

We kunnen eveneens alle componenten van een matrix vermenigvuldigen met een getal. Dit noemen we, net zoals voor vectoren, een scalaire vermenigvuldiging.

**Voorbeeld 2.12** We berekenen de som van twee matrices:

$$\mathbf{M}_1 + \mathbf{M}_2 = \begin{bmatrix} 1 & -2 \\ 1 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 2 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 2 & 0 \end{bmatrix}.$$

De scalaire vermenigvuldiging van  $\mathbf{M}_1$  met  $\alpha$  wordt gegeven door

$$\alpha \begin{bmatrix} 1 & -2 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} \alpha & -2\alpha \\ \alpha & 0 \end{bmatrix}. \quad \blacksquare$$

**Eigenschap 2.13** Het matrixproduct voldoet aan de volgende eigenschappen.

1. Het matrixproduct is associatief<sup>1</sup>:  $\mathbf{A}(\mathbf{BC}) = (\mathbf{AB})\mathbf{C}$ .
2. Het matrixproduct is links- en rechtsdistributief t.o.v. de optelling van matrices

$$\begin{aligned} \mathbf{A}(\mathbf{B} + \mathbf{C}) &= \mathbf{AB} + \mathbf{AC} \quad \text{en} \\ (\mathbf{B} + \mathbf{C})\mathbf{A} &= \mathbf{BA} + \mathbf{CA} \end{aligned}$$

---

<sup>1</sup>Dit hadden we reeds vermeld.

3. Voor elk reëel getal  $\alpha$  geldt dat

$$\alpha(\mathbf{AB}) = (\alpha\mathbf{A})\mathbf{B} = \mathbf{A}(\alpha\mathbf{B}).$$

4. Het matrixproduct interageert als volgt met het transponeren van matrices:

$$(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T.$$

In woorden: het getransponeerde van een product is het product van de getransponeerden *in omgekeerde volgorde*. ■

**Voorbeeld 2.14 (Interactie transponeren en matrixproduct)** Stel dat de matrices  $\mathbf{A}$  en  $\mathbf{B}$  gegeven worden door

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \quad \text{en} \quad \mathbf{B} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & -1 \\ 1 & 1 & 0 \end{bmatrix}.$$

Dan vinden we dat

$$\mathbf{AB} = \begin{bmatrix} 5 & 4 & -1 \\ 11 & 10 & -1 \end{bmatrix} \quad \text{en} \quad (\mathbf{AB})^T = \begin{bmatrix} 5 & 11 \\ 4 & 10 \\ -1 & -1 \end{bmatrix}.$$

Aan de andere kant geldt dat

$$\mathbf{B}^T = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & -1 & 0 \end{bmatrix} \quad \text{en} \quad \mathbf{A}^T = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}.$$

Nu bereken je gemakkelijk dat

$$\mathbf{B}^T \mathbf{A}^T = \begin{bmatrix} 5 & 11 \\ 4 & 10 \\ -1 & -1 \end{bmatrix}.$$

In dit voorbeeld zie je dus duidelijk dat

$$(\mathbf{AB})^T = \mathbf{B}^T \mathbf{A}^T. \quad \blacksquare$$

Als  $\mathbf{A}$  een vierkante matrix is, i.e. een matrix met evenveel rijen als kolommen, dan kan men zich afvragen of er een matrix bestaat die geen effect

heeft wanneer men deze links of rechts vermenigvuldigt met  $\mathbf{A}$ . We zoeken m.a.w. een matrix analoog aan het getal “1” voor de vermenigvuldiging van de reële getallen.

Het antwoord op deze vraag is eenvoudig en wordt de EENHEIDSMATRIX genoemd. Dit is een vierkante matrix die volledig uit nullen bestaat, behalve op de diagonaal (i.e. op die elementen waar de rij- en kolomindex gelijk is), waar er enen staan. Hieronder de drie kleinste eenheidsmatrices:

$$\mathbf{I}_1 = [1], \quad \mathbf{I}_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{en} \quad \mathbf{I}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

## 2.4 Inverse van een matrix

De eenheidsmatrix  $\mathbf{I}_2$  correspondeert met de (unieke) lineaire transformatie van  $\mathbb{R}^2$  naar  $\mathbb{R}^2$  die alle vectoren ongewijzigd laat, i.e. het beeld van elke vector onder de lineaire transformatie is zichzelf. Gegeven een lineaire transformatie  $L$  van  $\mathbb{R}^2$  naar  $\mathbb{R}^2$  dan is het niet vergezocht om zich af te vragen of er een lineaire transformatie bestaat die ervoor zorgt dat het effect van  $L$  ongedaan wordt gemaakt.

Als  $L$  bv. een rotatie is over 90 graden in wijzerzin, dan zorgt een rotatie over 90 graden in tegenwijzerzin ervoor dat de samenstelling van deze twee lineaire transformaties zodanig is dat het lijkt alsof er niets is gebeurd. We bekijken dit nu een beetje meer in detail. Een rotatie in wijzerzin heeft het volgende effect op de basisvectoren.

$$\mathbf{i} \mapsto -\mathbf{j} \quad \text{en} \quad \mathbf{j} \mapsto \mathbf{i}.$$

Dit geeft een matrix

$$\mathbf{R}_w = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}.$$

Een rotatie in tegenwijzerzin werkt als volgt op de basisvectoren:

$$\mathbf{i} \mapsto \mathbf{j} \quad \text{en} \quad \mathbf{j} \mapsto -\mathbf{i},$$

met als matrixvoorstelling

$$\mathbf{R}_t = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}.$$

Als we nu het matrixproduct berekenen, wat overeenkomt met het samenstellen van deze twee lineaire transformaties, dan vinden we:

$$\mathbf{R}_t \mathbf{R}_w = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

en ook

$$\mathbf{R}_w \mathbf{R}_t = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

We zeggen dat  $\mathbf{R}_t$  de inverse matrix is van  $\mathbf{R}_w$  (en omgekeerd).

Als volgend voorbeeld bekijken we de scheeftrekking

$$\mathbf{S} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

die we reeds voordien zagen, dan is het duidelijk dat als we de  $\mathbf{j}$  vector naar de andere kant laten bewegen, we de inverse transformatie krijgen. We vermoeden dus dat

$$\mathbf{T} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix}$$

het effect van  $\mathbf{S}$  ongedaan maakt. Je kan inderdaad verifiëren dat

$$\mathbf{T}\mathbf{S} = \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

en ook dat

$$\mathbf{S}\mathbf{T} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & -1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

**Definitie 2.15** Als  $\mathbf{A} \in \mathbb{R}^{n \times n}$  een vierkante matrix is waarvoor er een matrix  $\mathbf{B}$  bestaat waarvoor geldt dat

$$\mathbf{AB} = \mathbf{I}_n = \mathbf{BA},$$

dan zeggen we dat de matrix  $\mathbf{A}$  INVERTEERBAAR is en dat  $\mathbf{B}$  de INVERSE MATRIX is van  $\mathbf{A}$ , wat we noteren als

$$\mathbf{B} = \mathbf{A}^{-1}. \quad \blacksquare$$

**Opmerking 2.16** Uit de definitie is duidelijk dat als  $\mathbf{B}$  de inverse is van  $\mathbf{A}$ , dat  $\mathbf{A}$  ook de inverse is van  $\mathbf{B}$ , i.e.

$$(\mathbf{A}^{-1})^{-1} = \mathbf{A}. \quad \blacksquare$$



**Opmerking 2.17 (Niet-inverteerbare matrix)** Niet elke vierkante matrix is inverteerbaar! Bekijken we bv. de matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 1 & 2 \end{bmatrix}.$$

We merken op dat de kolommen van  $\mathbf{A}$  lineair afhankelijk zijn, en dit wil zeggen dat  $\mathbf{A}$  de volledige ruimte  $\mathbb{R}^2$  afbeeldt op een kleinere deelruimte. In dit geval is deze kleinere deelruimte een rechte.

De basisvector  $\mathbf{i}$  wordt afgebeeld op  $\mathbf{i} + \mathbf{j}$  maar ook de vector  $\mathbf{j}/2$  wordt hierop afgebeeld (en nog oneindig veel andere vectoren). Het is m.a.w. onmogelijk om het effect van  $\mathbf{A}$  ongedaan te maken, omdat de vector  $\mathbf{i} + \mathbf{j}$  niet tegelijkertijd op  $\mathbf{i}$  en  $\mathbf{j}/2$  kan worden afgebeeld. ■

**Definitie 2.18** Het aantal lineaire onafhankelijke kolommen van een matrix  $\mathbf{A}$  (die niet noodzakelijk vierkant is) noemen we de RANG van de matrix  $\mathbf{A}$ . De rang is m.a.w. de dimensie van de lineaire deelruimte opgespannen door de kolommen van  $\mathbf{A}$ . ■

We kunnen het begrip rang gebruiken om te karakteriseren welke vierkante matrices inverteerbaar zijn.

**Eigenschap 2.19** Een vierkante matrix  $\mathbf{A} \in \mathbb{R}^{n \times n}$  is inverteerbaar als en slechts de rang van  $\mathbf{A}$  gelijk is aan  $n$ . ■

*Bewijs* Zonder bewijs. ◇

**Eigenschap 2.20** De inverse van een 2 bij 2 matrix

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

bestaat als en slechts als  $ad - bc \neq 0$  en wordt gegeven door

$$\mathbf{A}^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}. \quad \blacksquare$$

*Bewijs* Als  $ad - bc = 0$  dan is de rang van  $\mathbf{A}$  strikt kleiner dan 2 (dit wordt uitgewerkt in een oefening) en bijgevolg is de matrix niet inverteerbaar wegens Eigenschap 2.19.

Als  $ad - bc \neq 0$  dan verifieer je gemakkelijk dat

$$\frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix} \cdot \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \frac{1}{ad - bc} \begin{bmatrix} ad - bc & 0 \\ 0 & ad - bc \end{bmatrix} = \mathbf{I}_2.$$

De andere volgorde verifieer je op dezelfde manier.  $\diamond$

**Opmerking 2.21** Later zullen we technieken zien om de inverse matrix te berekenen. In Python kunnen we de inverse matrix berekenen m.b.v. de numpy-functie `np.linalg.inv`<sup>2</sup>. ■

We eindigen met een belangrijke eigenschap van inverse matrices die intuïtief duidelijk is als we denken aan het feit dat matrices eigenlijk representaties zijn voor lineaire transformaties. Stel dat we eerst een lineaire transformatie  $A$  uitvoeren, en daarna een lineaire transformatie  $B$ . Het resultaat noemen we de lineaire transformatie  $C$ . In matrixvorm wordt dit

$$\mathbf{C} = \mathbf{B}\mathbf{A}.$$

Stel nu dat de lineaire transformaties  $A$  en  $B$  allebei inverteerbaar zijn, dan is het duidelijk dat om  $C$  ongedaan te maken, je eerst  $B$  ongedaan maakt en vervolgens  $A$ . In matrixvorm:

$$\mathbf{C}^{-1} = \mathbf{A}^{-1}\mathbf{B}^{-1}.$$

We vinden m.a.w. de volgende eigenschap.

**Eigenschap 2.22** Als  $\mathbf{A}$  en  $\mathbf{B}$  inverteerbare matrices zijn in  $\mathbb{R}^{n \times n}$ , dan is hun product inverteerbaar en er geldt:

$$(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}. \quad \blacksquare$$

*Bewijs* We moeten aantonen dat

$$(\mathbf{AB})(\mathbf{B}^{-1}\mathbf{A}^{-1}) = \mathbf{I} = (\mathbf{B}^{-1}\mathbf{A}^{-1})(\mathbf{AB}).$$

We verifiëren enkel de eerste gelijkheid, de tweede is analoog.

<sup>2</sup>Hier gaan we ervan uit dat je numpy op de standaard manier hebt geïmporteerd namelijk als `import numpy as np`

Ten eerste merken we op dat het matrixproduct zeker gedefinieerd is, want alle matrices zijn vierkante matrices van dezelfde grootte. Verder maken we gebruik van het feit dat het matrixproduct associatief is.

$$\begin{aligned}
 (\mathbf{AB})(\mathbf{B}^{-1}\mathbf{A}^{-1}) &= \mathbf{A}(\mathbf{BB}^{-1})\mathbf{A}^{-1} && \text{associativiteit} \\
 &= \mathbf{AIA}^{-1} && \text{definitie inverse} \\
 &= \mathbf{AA}^{-1} && \text{neutraal element} \\
 &= \mathbf{I} && \text{definitie inverse}
 \end{aligned}
 \quad \diamond$$

## 2.5 Loodrechte projectie op een deelruimte

Veronderstel dat een aantal lineair onafhankelijke vectoren  $\mathbf{a}_1$  t.e.m.  $\mathbf{a}_m$  uit  $\mathbb{R}^n$  gegeven zijn. Daarnaast is er een vector  $\mathbf{b} \in \mathbb{R}^n$  gegeven.

We wensen de vector  $\mathbf{b}$  loodrecht te projecteren op het omhulsel van  $\mathbf{a}_1$  t.e.m.  $\mathbf{a}_m$ . Dit komt neer op het schrijven van de vector  $\mathbf{b}$  als de som van de volgende twee delen

$$\mathbf{b} = \text{lineaire combinatie van } \mathbf{a}_1 \text{ t.e.m. } \mathbf{a}_m + \text{foutterm.} \quad (2.3)$$

Een lineaire combinatie van de vectoren  $\mathbf{a}_1$  t.e.m.  $\mathbf{a}_m$  kunnen we schrijven als

$$\begin{bmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_m \\ | & | & & | \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix} = \mathbf{Ax}$$

waarbij we de (onbekende) getallen  $x_1$  t.e.m.  $x_m$  trachten te bepalen.

Als de projectie loodrecht is, dan staat de foutterm uit vergelijking (2.3) loodrecht op de vectoren  $\mathbf{a}_1$  t.e.m.  $\mathbf{a}_m$ . Dit feit kunnen we gebruiken om de onbekende coëfficiënten  $x_1$  t.e.m.  $x_m$  te bepalen. De foutterm kunnen we schrijven als

$$\mathbf{b} - \mathbf{Ax}.$$

Als we uitdrukken dat deze vector loodrecht staat op de vectoren  $\mathbf{a}_1$  t.e.m.  $\mathbf{a}_m$ , dan vinden we de volgende  $m$  vergelijkingen, die elk overeenkomen met het feit dat het inwendig product van een vector  $\mathbf{a}_i$  met de vector  $\mathbf{b} - \mathbf{Ax}$

gelijk is aan nul:

$$\begin{aligned} \mathbf{a}_1^T(\mathbf{b} - \mathbf{A}\mathbf{x}) &= 0, \\ \mathbf{a}_2^T(\mathbf{b} - \mathbf{A}\mathbf{x}) &= 0, \\ &\vdots \\ \mathbf{a}_m^T(\mathbf{b} - \mathbf{A}\mathbf{x}) &= 0. \end{aligned}$$

Dit kunnen we ook compacter schrijven als

$$\mathbf{A}^T(\mathbf{b} - \mathbf{A}\mathbf{x}) = \mathbf{0}.$$

Deze matrixvergelijking kunnen we nu manipuleren om  $\mathbf{x}$  te bepalen:

$$\begin{aligned} \mathbf{A}^T(\mathbf{b} - \mathbf{A}\mathbf{x}) &= \mathbf{0} \\ \iff \mathbf{A}^T\mathbf{b} - \mathbf{A}^T\mathbf{A}\mathbf{x} &= \mathbf{0} \\ \iff \mathbf{A}^T\mathbf{b} &= \mathbf{A}^T\mathbf{A}\mathbf{x}. \end{aligned}$$

Aangezien de kolommen van  $\mathbf{A}$  lineair onafhankelijk zijn is  $\mathbf{A}^T\mathbf{A} \in \mathbb{R}^{m \times m}$  zeker inverteerbaar<sup>3</sup> en kunnen we de onbekende coëfficiënten bepalen als

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} \in \mathbb{R}^m. \quad (2.4)$$

De finale projectie wordt echter gegeven door  $\mathbf{A}\mathbf{x}$  ofwel

$$\mathbf{A}\mathbf{x} = \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} = \mathbf{P}\mathbf{b} \in \mathbb{R}^n,$$

waarbij de PROJECTIEMATRIX  $\mathbf{P}$  gegeven wordt door

$$\mathbf{P} = \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T \in \mathbb{R}^{n \times n}. \quad (2.5)$$

**Opmerking 2.23 (Speciaal geval  $m = 1$ )** Wanneer  $m = 1$ , dan vinden we de formules terug die we reeds in Sectie 1.5 hebben gevonden. Inderdaad: de projectiematrix  $\mathbf{P}$  wordt gegeven door:

$$\mathbf{P} = \begin{bmatrix} | \\ \mathbf{a} \\ | \end{bmatrix} \left( \begin{bmatrix} \text{---} & \mathbf{a}^T & \text{---} \end{bmatrix} \begin{bmatrix} | \\ \mathbf{a} \\ | \end{bmatrix} \right)^{-1} \begin{bmatrix} \text{---} & \mathbf{a}^T & \text{---} \end{bmatrix}.$$

Het berekenen van de inverse matrix is in dit geval triviaal omdat het een 1 bij 1 matrix, m.a.w. een reëel getal, is. We kunnen dit reëel getal voorop plaatsen.

$$\mathbf{P} = \frac{1}{\mathbf{a} \cdot \mathbf{a}} \begin{bmatrix} | \\ \mathbf{a} \\ | \end{bmatrix} \begin{bmatrix} \text{---} & \mathbf{a}^T & \text{---} \end{bmatrix}.$$

<sup>3</sup>Dit mag je aannemen zonder bewijs.

De projectie van de vector  $\mathbf{b}$  wordt gegeven door  $\mathbf{Pb}$ :

$$\begin{aligned}\mathbf{Pb} &= \frac{1}{\mathbf{a} \cdot \mathbf{a}} \begin{bmatrix} | \\ \mathbf{a} \\ | \end{bmatrix} \left[ \text{---} \mathbf{a}^T \text{---} \right] \begin{bmatrix} | \\ \mathbf{b} \\ | \end{bmatrix} \\ &= \frac{1}{\mathbf{a} \cdot \mathbf{a}} \begin{bmatrix} | \\ \mathbf{a} \\ | \end{bmatrix} \left( \left[ \text{---} \mathbf{a}^T \text{---} \right] \begin{bmatrix} | \\ \mathbf{b} \\ | \end{bmatrix} \right) \\ &= \frac{1}{\mathbf{a} \cdot \mathbf{a}} \begin{bmatrix} | \\ \mathbf{a} \\ | \end{bmatrix} \mathbf{a} \cdot \mathbf{b} \\ &= \frac{\mathbf{a} \cdot \mathbf{b}}{\mathbf{a} \cdot \mathbf{a}} \mathbf{a}.\end{aligned}$$

De laatste manipulatie is geldig omdat  $\mathbf{a} \cdot \mathbf{b}$  een reëel getal is, en we dit ook voor de vector  $\mathbf{a}$  kunnen plaatsen.

We vinden m.a.w. dat in het geval  $m = 1$

$$\mathbf{Pb} = \frac{\mathbf{a} \cdot \mathbf{b}}{\mathbf{a} \cdot \mathbf{a}} \mathbf{a}.$$

Dit is precies gelijk aan hetgeen reeds in Eigenschap 1.21 werd gevonden, en toont aan dat de formules die we gevonden hebben in dit hoofdstuk een logische uitbreiding zijn van de reeds gekende formules voor het bijzondere geval  $m = 1$ . ■

**Voorbeeld 2.24** Veronderstel dat we werken in  $\mathbb{R}^3$  en dat de vectoren  $\mathbf{a}_1$  en  $\mathbf{a}_2$  gegeven worden door

$$\mathbf{a}_1 = \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} \quad \text{en} \quad \mathbf{a}_2 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}.$$

We wensen de loodrechte projectie te vinden van vectoren op het omhulsel van  $\mathbf{a}_1$  en  $\mathbf{a}_2$ . Hiertoe bepalen we de projectiematrix  $\mathbf{P}$ . We starten met de berekening van  $\mathbf{A}^T \mathbf{A}$  waarbij  $\mathbf{A}$  de vectoren  $\mathbf{a}_1$  en  $\mathbf{a}_2$  als kolommen heeft:

$$\mathbf{A}^T \mathbf{A} = \begin{bmatrix} 1 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 5 & 3 \\ 3 & 2 \end{bmatrix}.$$

De inverse hiervan wordt wegens Eigenschap 2.20 gegeven door

$$(\mathbf{A}^T \mathbf{A})^{-1} = \frac{1}{5 \times 2 - 3 \times 3} \begin{bmatrix} 2 & -3 \\ -3 & 5 \end{bmatrix} = \begin{bmatrix} 2 & -3 \\ -3 & 5 \end{bmatrix}.$$

De projectiematrix  $\mathbf{P}$  is dus

$$\begin{aligned} \mathbf{P} &= \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 2 & -3 \\ -3 & 5 \end{bmatrix} \begin{bmatrix} 1 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 \\ 2 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} -1 & 1 & 0 \\ 2 & -1 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \end{aligned}$$

Als we nu een willekeurige vector  $\mathbf{b}$  projecteren m.b.v. deze projectiematrix dan vinden we

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ 0 \end{bmatrix}.$$

Dit komt (in dit geval) neer op het op nul zetten van de  $z$ -component. Dit is logisch aangezien de vectoren  $\mathbf{a}_1$  en  $\mathbf{a}_2$  samen het  $XY$ -vlak opspannen.

Als we willen weten wat de coëfficiënten zijn van deze projectie in de basis gevormd door  $\mathbf{a}_1$  en  $\mathbf{a}_2$ , dan berekenen we

$$(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b} = \begin{bmatrix} -1 & 1 & 0 \\ 2 & -1 & 0 \end{bmatrix} \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} -b_1 + b_2 \\ 2b_1 - b_2 \end{bmatrix}.$$

Inderdaad, er geldt dat

$$\begin{bmatrix} b_1 \\ b_2 \\ 0 \end{bmatrix} = (-b_1 + b_2) \begin{bmatrix} 1 \\ 2 \\ 0 \end{bmatrix} + (2b_1 - b_2) \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}.$$

Wanneer je een vector  $\mathbf{b}$  projecteert die reeds behoort tot het omhulsel van de vectoren  $\mathbf{a}_1$  t.e.m.  $\mathbf{a}_m$  dan is de projectie uiteraard gelijk aan de vector zelf. Inderdaad, stel dat

$$\mathbf{b} = \mathbf{A} \mathbf{y},$$

waarbij  $\mathbf{y}$  de coördinaten zijn van  $\mathbf{b}$  (t.o.v. de basis  $\mathbf{a}_1$  t.e.m.  $\mathbf{a}_m$ ), dan geldt volgens vergelijking (2.4) dat de coördinaten van de projectie gegeven worden door:

$$\begin{aligned}
 \mathbf{x} &= (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b} && \text{definitie } \mathbf{x} \\
 &= (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{A} \mathbf{y} && \mathbf{b} = \mathbf{A} \mathbf{y} \\
 &= (\mathbf{A}^T \mathbf{A})^{-1} (\mathbf{A}^T \mathbf{A}) \mathbf{y} && \text{associativiteit matrixproduct} \\
 &= \mathbf{I} \mathbf{y} && \text{definitie inverse} \\
 &= \mathbf{y} && \mathbf{I} \text{ is neutraal element.}
 \end{aligned}$$

De coördinaten van de projectie zijn m.a.w. gelijk aan de originele coördinaten. Dit betekent ook dat twee keer projecteren hetzelfde effect heeft als één keer projecteren.

In het algemeen noemen we een matrix waarvoor  $\mathbf{A}^2 = \mathbf{A}$  een IDEMPO-TENTE matrix.

**Eigenschap 2.25** Een projectiematrix  $\mathbf{P}$  is idempotent:

$$\mathbf{P}^2 = \mathbf{P}.$$

De matrix  $\mathbf{P}$  is bovendien ook symmetrisch:

$$\mathbf{P}^T = \mathbf{P}. \quad \blacksquare$$

*Bewijs* Een oefening in matrixmanipulatie.  $\diamond$

**Opmerking 2.26** Net zoals in het ééndimensionale geval is de loodrechte projectie van  $\mathbf{b}$  op een lineaire deelruimte de vector van die lineaire deelruimte die *het dichtst bij*  $\mathbf{b}$  ligt.  $\blacksquare$

## 2.6 Oefeningen

1. Bereken het matrixproduct  $\mathbf{AB}$  van de volgende matrices. Bereken ook  $\mathbf{BA}$  indien dit mogelijk is en vergelijk met  $\mathbf{AB}$ .

a)  $\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$  en  $\mathbf{B} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ .

b)  $\mathbf{A} = \begin{bmatrix} \alpha & 0 \\ 0 & \alpha \end{bmatrix}$  en  $\mathbf{B} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ .

c)  $\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix}$  en  $\mathbf{B} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$ .

d)  $\mathbf{A} = \begin{bmatrix} 2 \\ 2 \\ 1 \end{bmatrix}$  en  $\mathbf{B} = \begin{bmatrix} 3 & 4 & 6 \end{bmatrix}$ .

2. Numpy. Implementeer het matrixproduct als een concatenatie van matrix-vectorproducten. Vergelijk je antwoord met het ingebouwde matrixproduct in Numpy.
3. Numpy. Verifieer de eigenschappen van het matrixproduct zoals vermeld in Eigenschap 2.13 door willekeurige matrices te genereren en de linker- en rechterleden te vergelijken.
4. Veronderstel dat  $\mathbf{A} \in \mathbb{R}^{n \times m}$  een willekeurige matrix is. Overtuig jezelf dat

$$\mathbf{A}^T \mathbf{A} \quad \text{en} \quad \mathbf{A} \mathbf{A}^T$$

allebei gedefinieerd zijn en dat ze allebei symmetrische matrices zijn.

5. Stel dat

$$\mathbf{A} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

en dat  $ad - bc = 0$ . Toon aan dat de kolommen van  $\mathbf{A}$  lineair afhankelijk zijn. (Dit heeft dan onmiddellijk ook als gevolg dat de rang van  $\mathbf{A}$  strikt kleiner dan twee is.)

6. Vind de projectiematrix voor de orthogonale projectie op het omhulsel van

$$\mathbf{a}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \text{en} \quad \mathbf{a}_2 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}.$$

Gebruik deze projectiematrix om de projectie te vinden van

$$\mathbf{b} = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}$$

op deze deelruimte. Wat zijn de coördinaten van de projectie van  $\mathbf{b}$  t.o.v. de basisvectoren  $\mathbf{a}_1$  en  $\mathbf{a}_2$ ?



7. We werken in  $\mathbb{R}^3$ . Vind de projectiematrix  $\mathbf{P}$  op het vlak met als vergelijking:

$$x + y - z = 0.$$

Doe dit op twee manieren:

- Projecteer loodrecht op het vlak door eerst een basis te vinden voor deze deelruimte. Deze basis zal bestaan uit twee vectoren.
- Projecteer loodrecht op de normaalvector op het vlak (i.e. de rechte loodrecht op het vlak), en realiseer je dat de originele projectiematrix  $\mathbf{P}$  gegeven wordt door

$$\mathbf{I} - \mathbf{P}_{\text{rechte}},$$

waarbij  $\mathbf{P}_{\text{rechte}}$  de projectiematrix op de normaalvector voorstelt.

8. Student G. Roentje “bewijst” dat elke idempotente matrix noodzakelijk gelijk is aan de identiteitsmatrix. Hij redeneert als volgt:

$$\begin{aligned} \mathbf{A}^2 &= \mathbf{A} \\ \implies \mathbf{A}\mathbf{A} &= \mathbf{A} \\ \implies \mathbf{A}^{-1}\mathbf{A}\mathbf{A} &= \mathbf{A}^{-1}\mathbf{A} \\ \implies (\mathbf{A}^{-1}\mathbf{A})\mathbf{A} &= \mathbf{A}^{-1}\mathbf{A} \\ \implies \mathbf{I}\mathbf{A} &= \mathbf{I} \\ \implies \mathbf{A} &= \mathbf{I} \end{aligned}$$

Helaas voor hem is deze redenering **fout!** Waar gaat hij de mist in?

9. Student G. Roentje “bewijst” op de volgende manier dat een projectiematrix steeds gelijk is aan de identiteitsmatrix, maar zijn redenering is **fout!** Waar gaat hij de mist in?

$$\begin{aligned} \mathbf{P} &= \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T \\ &= \mathbf{A}(\mathbf{A}^{-1}(\mathbf{A}^T)^{-1})\mathbf{A}^T \\ &= (\mathbf{A}\mathbf{A}^{-1})((\mathbf{A}^T)^{-1}\mathbf{A}^T) \\ &= (\mathbf{I})(\mathbf{I}) \\ &= \mathbf{I}. \end{aligned}$$

# Stelsels lineaire vergelijkingen

We starten met een raadsel:

Een moeder is 21 jaar ouder dan haar zoon. Over 6 jaar zal ze 5 keer zijn leeftijd hebben. Waar is de vader ?

De oplossing voor dit raadsel vergt een beetje wiskunde en iets wat men “stelsels lineaire vergelijkingen” noemt, en dat is (uiteraard) het onderwerp van dit hoofdstuk.

## 3.1 Stelsels lineaire vergelijkingen

In de wiskunde en bij toepassingen ervan in de natuurkunde, informatica, economie, etc. komt het vaak voor dat men vergelijkingen moet oplossen. Eén van de meest eenvoudige gevallen is wanneer alle onbekenden “op zichzelf” voorkomen, zonder dat er andere functies op worden toegepast of zonder dat de variabelen interageren met elkaar. Een voorbeeld zal het één en ander veel concreter maken. Hieronder vinden we een stelsel met drie lineaire vergelijkingen in drie onbekenden  $x$ ,  $y$  en  $z$ :

$$\begin{cases} x + 2y + z = 2 \\ 3x + 8y + z = 12 \\ 4y + z = 2. \end{cases} \quad (3.1)$$

Typisch is het dan de bedoeling om zo’n stelsel op te lossen, i.e. om alle combinaties van  $x$ ,  $y$  en  $z$  te vinden waarvoor alledrie de vergelijkingen tegelijkertijd voldaan zijn.

Om zo'n stelsel handig voor te stellen, zeker wanneer men het wil manipuleren m.b.v. computercode, schrijven we het linkerlid als een matrix-vectorproduct en het rechterlid als een kolomvector. Zo wordt het stelsel (3.1) geschreven als

$$\begin{bmatrix} 1 & 2 & 1 \\ 3 & 8 & 1 \\ 0 & 4 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2 \\ 12 \\ 2 \end{bmatrix}.$$

Meestal wordt de COËFFICIËNTENMATRIX afgekort als **A**, de vector van de onbekenden als **x** en het rechterlid als de vector **b**. In dit geval is dus

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 1 \\ 3 & 8 & 1 \\ 0 & 4 & 1 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad \text{en} \quad \mathbf{b} = \begin{bmatrix} 2 \\ 12 \\ 2 \end{bmatrix},$$

zodat het stelsel geschreven wordt als

$$\mathbf{Ax} = \mathbf{b}. \tag{3.2}$$

Deze notatie heeft niet enkel het voordeel dat ze veel compacter is, ze maakt ook duidelijk wat we eigenlijk betekent om een stelsel op te lossen én men kan (theoretisch) zien wanneer er een oplossing zal zijn.

Oplossen van het stelsel (3.2) betekent m.a.w. het vinden van een vector **x** die onder **A** wordt afgebeeld op **b**. Herinner je dat we het matrix-vectorproduct gedefinieerd hebben als een lineaire combinatie van kolommen. Het stelsel zal dus enkel een oplossing hebben wanneer **b** geschreven kan worden als een lineaire combinatie van de kolommen van **A**, i.e. als **b** behoort tot het omhulsel van de kolommen van **A**, i.e. als

$$\mathbf{b} \in \text{span}\{\mathbf{a}_1, \mathbf{a}_2, \dots, \mathbf{a}_n\}$$

met

$$\mathbf{A} = \begin{bmatrix} | & | & \cdots & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_n \\ | & | & \cdots & | \end{bmatrix}.$$

Het lineair omhulsel van de kolommen van een matrix **A** noemen we de KOLOMRUIMTE van de matrix **A**, en noteren we als  $\text{Col}(\mathbf{A})$ .

De volgende eigenschap formuleert nog eens duidelijk de voorwaarde waaraan een stelsel lineaire vergelijkingen moet voldoen opdat het een oplossing zou hebben.

**Eigenschap 3.1 (Oplosbaarheid stelsel lineaire vergelijkingen)** Een stelsel lineaire vergelijkingen

$$\mathbf{Ax} = \mathbf{b}$$

is oplosbaar als en slechts als  $\mathbf{b}$  behoort tot de kolomruimte van  $\mathbf{A}$ , i.e. als  $\mathbf{b} \in \text{Col}(\mathbf{A})$ . ■

*Bewijs* Dit volgt rechtstreeks uit het feit dat  $\mathbf{Ax}$  bij definitie een lineaire combinatie is van de kolommen van  $\mathbf{A}$ , en bijgevolg steeds behoort tot  $\text{Col}(\mathbf{A})$ . ◇

### 3.1.1 Oefeningen

1. Noteer volgend stelsel lineaire vergelijkingen in de vorm (3.2):

$$\begin{cases} x + 2u + y - z = 3 \\ x + 2y - 3z + 7 = 0 \\ y - u = 1. \end{cases}$$

2. “Vertaal” het raadsel aan de start van dit hoofdstuk en schrijf het als een stelsel lineaire vergelijkingen.
3. Geef voor elk van de volgende vergelijkingen aan of het een lineaire vergelijking is of niet.
  - a)  $x^2 + x + 1 = 0$ ,
  - b)  $x^2 + y^2 = 1$
  - c)  $x + y + xy = 0$
  - d)  $2^x + y = 7$
  - e)  $\sqrt{x} + \sqrt{y} + \sqrt{z} = 1$

## 3.2 Gaussische eliminatie

Sommige stelsels lineaire vergelijkingen lijken “gemakkelijker” op te lossen dan andere.

Veronderstel bv. dat een stelsel lineaire vergelijkingen gegeven wordt door

$$\begin{cases} x + 2y + z = 2 \\ 2y - 2z = 6 \\ 5z = -10. \end{cases} \quad (3.3)$$

Uit de laatste vergelijking vinden we onmiddellijk dat  $z = -2$ . Als we deze waarde substitueren in de middelste vergelijking dan vinden we

$$2y + 4 = 6 \iff y = 1.$$

Als we tenslotte de gevonden waarden voor  $z$  en  $y$  substitueren in de eerste vergelijking dan vinden we

$$x + 2 - 2 = 2 \iff x = 2.$$

De OPLOSSINGSVERZAMELING van het stelsel in vergelijking (3.3) wordt m.a.w. gegeven door het singleton

$$V = \left\{ \begin{bmatrix} 2 \\ 1 \\ -2 \end{bmatrix} \right\}.$$

De reden dat het stelsel (3.3) eenvoudig op te lossen was ligt in de vorm van de coëfficiëntenmatrix  $\mathbf{A}$ :

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 2 & -2 \\ 0 & 0 & 5 \end{bmatrix}.$$

Deze matrix is een BOVENDRIEHOEKSMATRIX, i.e. alle elementen onder de diagonaal zijn gelijk aan nul. Hierdoor kan het stelsel eenvoudig opgelost worden m.b.v. ACHTERWAARTSE SUBSTITUTIE zoals zonet werd gedemonstreerd.

Het doel van Gaussische eliminatie bestaat erin om een (willekeurig) stelsel lineaire vergelijkingen om te vormen tot een EQUIVALENT STELSEL, i.e. een stelsel met dezelfde oplossingsverzameling als het originele stelsel, maar met een coëfficiëntenmatrix in bovendriehoeksvorm. Dit laatste stelsel kan dan eenvoudig opgelost worden m.b.v. achterwaartse substitutie.

Het is niet moeilijk om in te zien dat wanneer men bij een vergelijking een willeurig veelvoud van een *andere* vergelijking optelt, dat men dan een equivalent stelsel bekomt. Beschouw als voorbeeld het stelsel dat we reeds hebben gezien:

$$\begin{cases} x + 2y + z = 2 \\ 3x + 8y + z = 12 \\ 4y + z = 2. \end{cases}$$

Als we van de tweede vergelijking drie keer de eerste vergelijking aftrekken<sup>1</sup>, dan blijven de eerste en derde vergelijking ongewijzigd, en de tweede vergelijking wordt

$$3x + 8y + z - 3(x + 2y + z) = 12 - 3(2) \iff 2y - 2z = 6$$

zodat het nieuwe en equivalente stelsel gelijk is aan

$$\begin{cases} x + 2y + z = 2 \\ 2y - 2z = 6 \\ 4y + z = 2. \end{cases}$$

Vervolgens verminderen we de derde vergelijking met twee keer de tweede vergelijking. De derde vergelijking wordt:

$$4y + z - 2(2y - 2z) = 2 - 2(6) \iff 5z = -10,$$

en het stelsel is finaal gelijk aan

$$\begin{cases} x + 2y + z = 2 \\ 2y - 2z = 6 \\ 5z = -10. \end{cases}$$

Dit stelsel kan eenvoudig opgelost worden (zoals we reeds hebben gedaan) m.b.v. achterwaartse substitutie. Omdat het laatste stelsel equivalent is met het originele stelsel (3.1) is de unieke oplossing van het eenvoudige laatste stelsel meteen ook de unieke oplossing van het stelsel (3.1).

**Voorbeeld 3.2 (Strijdig stelsel)** Een strijdig stelsel herkent men m.b.v. Gaussische eliminatie doordat er “onmogelijke” vergelijkingen tevoorschijn komen die zeggen dat nul gelijk moet zijn aan iets dat verschillend is van nul. Bekijk we bv. het stelsel

$$\begin{bmatrix} 1 & 2 \\ 2 & 4 \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}.$$

We zien onmiddellijk dat dit stelsel geen oplossing kan hebben omdat de kolomruimte van de coëfficiëntenmatrix  $\mathbf{A}$  bestaat uit de veelvouden van  $\begin{bmatrix} 1 \\ 2 \end{bmatrix}$ , en het rechterlid behoort hier niet toe.

<sup>1</sup>Merk op: we tellen m.a.w.  $-3$  keer de eerste vergelijking op bij de tweede.

Als we Gaussische eliminatie toepassen en de tweede rij verminderen met 2 keer de eerste rij dan wordt de tweede vergelijking:

$$2x + 4y - 2(x + 2y) = 3 - 2 \times 1 \iff 0x + 0y = 1,$$

of  $0 = 1$ , wat natuurlijk onmogelijk is. Dit wijst erop dat dit stelsel geen oplossingen heeft. Het is m.a.w. een strijdig stelsel (wat we eigenlijk al reeds wisten). ■

**Voorbeeld 3.3 (Stelsel met oneindig veel oplossingen)** Als een stelsel lineaire vergelijkingen meer dan 1 oplossing heeft, dan zijn er altijd oneindig veel oplossingen. We geven hier nu een voorbeeld van. Beschouw onderstaand stelsel:

$$\begin{bmatrix} 1 & 2 & 3 \\ 2 & 5 & 7 \\ 3 & 7 & 10 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 5 \\ 12 \\ 17 \end{bmatrix}.$$

Voluit geschreven wordt dit stelsel:

$$\begin{cases} x + 2y + 3z = 5 \\ 2x + 5y + 7z = 12 \\ 3x + 7y + 10z = 17. \end{cases}$$

We trekken van de tweede vergelijking twee keer de eerste vergelijking af. De nieuwe tweede vergelijking wordt dan

$$y + z = 2.$$

We trekken van de derde vergelijking drie keer de eerste vergelijking af. De nieuwe derde vergelijking wordt

$$y + z = 2.$$

Tenslotte nemen we het verschil van de nieuwe derde en nieuwe tweede vergelijking en het stelsel wordt

$$\begin{cases} x + 2y + 3z = 5 \\ y + z = 2 \\ 0 = 0. \end{cases}$$

We zien dat van de drie vergelijkingen er eigenlijk nog maar twee overblijven, want de derde vergelijking is uiteraard altijd voldaan. Er zijn dus niet

genoeg vergelijkingen om alle veranderlijken een unieke waarde te geven. Als we achterwaartse substitutie toepassen dan zien we dat we  $z$  een willekeurige waarde  $r$  kunnen geven. Als  $z$  deze waarde heeft gekregen dan volgt uit de middelste vergelijking dat  $y = 2 - r$ . Uit de eerste vergelijking volgt tenslotte dat

$$x = 5 - 2y - 3z = 5 - 2(2 - r) - 3r = 1 - r.$$

De oplossingenverzameling van dit stelsel is m.a.w. gelijk aan

$$V = \left\{ \begin{bmatrix} 1 - r \\ 2 - r \\ r \end{bmatrix} \mid r \in \mathbb{R} \right\}.$$

In dit geval heeft het stelsel één vrijheidsgraad: dit komt omdat de rang van de coëfficiëntenmatrix gelijk is aan twee (want de derde kolom is de som van de twee eerste kolommen), terwijl het aantal variabelen gelijk is aan drie. Het aantal vrijheidsgraden is dus  $3 - 2 = 1$ . ■

### 3.2.1 Oefeningen

1. Schrijf het volgende probleem als een stelsel lineaire vergelijkingen en los het op m.b.v. Gaussische eliminatie.

Bij de constructie van een groot industriegebouw zijn aanzienlijke fouten gemaakt. De herstellkosten worden geraamd op 300 000 EUR. Bij de bouw waren de bouwheer (opdrachtgever), de aannemer en de ir.-architect betrokken. Ze dragen elk een deel van de verantwoordelijkheid in wat misgelopen is. De ir.-architect heeft fouten gemaakt in de berekening van de constructie, de aannemer heeft niet overal de voorgeschreven materialen gebruikt en de bouwheer heeft achteraf zwaardere machines op de eerste verdieping geplaatst dan voorzien. Door een onafhankelijke expert wordt de verantwoordelijkheid van de aannemer als dubbel zo hoog ingeschat als die van de bouwheer. Het aandeel van de ir.-architect en de aannemer samen in de totale herstellkosten bedraagt 250 000 EUR. Hoeveel moet elk betalen?

2. Los het raadsel uit de inleiding op. Los hiertoe het stelsel lineaire vergelijkingen op dat je reeds in een vorige oefening hebt opgesteld.



### 3.3 De LU matrixdecompositie

In deze sectie beschouwen we enkel “vierkante” stelsels, i.e. stelsels waarvan het aantal vergelijkingen gelijk is aan het aantal onbekenden.

#### 3.3.1 LU matrixdecompositie zonder rijverwisselingen

Als men het proces van Gaussische eliminatie wenst te implementeren in computercode, dan is het handiger als men de bewerkingen rechtstreeks op de coëfficiëntenmatrix  $\mathbf{A}$  (en op het rechterlid  $\mathbf{b}$ ) kan uitvoeren. Bij wijze van voorbeeld:

$$\begin{bmatrix} 1 & 2 & 1 \\ 3 & 8 & 1 \\ 0 & 4 & 1 \end{bmatrix} \xrightarrow{R_2 \leftarrow R_2 - (3/1)R_1} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 2 & -2 \\ 0 & 4 & 1 \end{bmatrix} \xrightarrow{R_3 \leftarrow R_3 - (4/2)R_2} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 2 & -2 \\ 0 & 0 & 5 \end{bmatrix} \quad (3.4)$$

Dit proces is tamelijk mechanisch en vereist geen “inzichten”: het veelvoud dat men dient te gebruiken om een nul te maken op een bepaalde plaats wordt gemakkelijk berekend. Tijdens het proces is er steeds een bepaald element, het PIVOTELEMENT, waaronder we nullen wensen te maken. *Dit pivotelement moet steeds verschillend zijn van nul!* Als het pivotelement op rij  $i$  gelijk is aan  $a$  en we wensen een element op rij  $j$  (in dezelfde kolom als het pivotelement) dat gelijk is aan  $b$  de waarde nul te geven dan voeren we de rijoperatie

$$R_j \leftarrow R_j - \frac{b}{a}R_i$$

uit. Merk op dat

$$b - \frac{b}{a}a = 0,$$

zodat onder het pivotelement  $a$  (op rij  $i$ ) daar waar eerst een  $b$  (op rij  $j$ ) stond nu inderdaad een nul verschijnt.

In Hoofdstuk 2 hebben we gezien dat het een matrix-vectorproduct gelijk is aan een lineaire combinatie van de kolommen van de matrix. Zo’n matrix-vectorproduct is ook gelijk aan het matrixproduct van een matrix en een matrix met één kolom (i.e. een kolomvector). Als we zo’n product transponeren dan krijgen we een product van een matrix met één rij (i.e. een rijvector) en een matrix. Zo’n product is equivalent met een *lineaire combinatie van*

de rijen van de matrix. Zo is bv.

$$\begin{bmatrix} -3 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \\ 3 & 8 & 1 \\ 0 & 4 & 1 \end{bmatrix} = -3 \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} + 1 \begin{bmatrix} 3 & 8 & 1 \end{bmatrix} + 0 \begin{bmatrix} 0 & 4 & 1 \end{bmatrix} \\ = \begin{bmatrix} 0 & 2 & -2 \end{bmatrix}.$$

Bemerk de gelijkheid met de rijoperatie

$$R_2 \leftarrow R_2 - 3R_1.$$

Op deze manier is het ook niet moeilijk te zien dat de eerste rijoperatie van (3.4) equivalent is met

$$\begin{bmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \\ 3 & 8 & 1 \\ 0 & 4 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 2 & -2 \\ 0 & 4 & 1 \end{bmatrix}.$$

De tweede stap in (3.4) kunnen we dan schrijven als

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 2 & -2 \\ 0 & 4 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 2 & -2 \\ 0 & 0 & 5 \end{bmatrix}.$$

Het volledige proces om van de originele coëfficiëntenmatrix over te gaan op de bovendriehoeksmatrix is m.a.w.

$$\mathbf{E}_{3,2}\mathbf{E}_{2,1}\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \\ 3 & 8 & 1 \\ 0 & 4 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 2 & -2 \\ 0 & 0 & 5 \end{bmatrix} = \mathbf{U}.$$

Door de ELEMENTAIRE MATRICES  $\mathbf{E}_{i,j}$  één voor één te inverteren kunnen we dit ook schrijven als:

$$\mathbf{A} = \mathbf{E}_{2,1}^{-1}\mathbf{E}_{3,2}^{-1}\mathbf{U},$$

waarbij we de bovendriehoeksmatrix  $\mathbf{U}$  hebben genoemd. De inverse matrices zijn zeer eenvoudig te bepalen<sup>2</sup>:

$$\begin{bmatrix} 1 & 0 & 0 \\ -3 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{en} \quad \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -2 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix}.$$

<sup>2</sup>Je zal dit expliciet narekenen in een oefening maar het is duidelijk dat als je eerst drie keer de eerste rij aftrekt van de tweede rij en vervolgens drie keer de eerste rij optelt bij de tweede rij, dat je dan netto “niets” hebt gedaan.

Als we deze twee inverse matrices met elkaar vermenigvuldigen dan krijgen we

$$\begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix}.$$

Deze laatste matrix is een benedendriehoeksmatrix en noemen we  $\mathbf{L}$ . We hebben m.a.w. de matrix  $\mathbf{A}$  herschreven als het product van twee bijzondere matrices. In dit geval zijn de twee bijzondere matrices een beneden- en bovendriehoeksmatrix:

$$\mathbf{A} = \mathbf{L}\mathbf{U},$$

of

$$\begin{bmatrix} 1 & 2 & 1 \\ 3 & 8 & 1 \\ 0 & 4 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 2 & -2 \\ 0 & 0 & 5 \end{bmatrix}. \quad (3.5)$$

### 3.3.2 LU decompositie met rijverwisselingen

De LU-decompositie faalt wanneer één van de pivotelementen gelijk is aan nul (want er moet door de pivotelementen worden gedeeld.) Als bv. de matrix  $\mathbf{A}$  gegeven wordt door

$$\begin{bmatrix} 0 & 4 & 1 \\ 1 & 2 & 1 \\ 3 & 8 & 1 \end{bmatrix}$$

dan faalt het proces bij de eerste stap. Men kan in dit geval het probleem oplossen door op zoek te gaan naar een nieuw pivotelement *onder* de positie van het problematische element maar in dezelfde kolom.

Omwille van de *numerieke stabiliteit* wordt er dan gezocht naar het element dat het grootst is in absolute waarde.

Als we dit proces nu eens uitvoeren voor de gegeven matrix dan wisselen we eerste de eerste en de derde rij. In matrixvorm doen we dit aan de hand van een PERMUTATIEMATRIX  $\mathbf{P}_{13}$ :

$$\mathbf{P}_{13}\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 4 & 1 \\ 1 & 2 & 1 \\ 3 & 8 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 8 & 1 \\ 1 & 2 & 1 \\ 0 & 4 & 1 \end{bmatrix}.$$

**Opmerking 3.4** De matrix  $\mathbf{P}_{13}$  is gelijk aan de eenheidsmatrix waarvan de eerste en de derde rij van plaats werden gewisseld. ■

Vervolgens voeren we een elementaire rij-operatie uit

$$\mathbf{E}_{21} \begin{bmatrix} 3 & 8 & 1 \\ 1 & 2 & 1 \\ 0 & 4 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -\frac{1}{3} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 3 & 8 & 1 \\ 1 & 2 & 1 \\ 0 & 4 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 8 & 1 \\ 0 & -\frac{2}{3} & \frac{2}{3} \\ 0 & 4 & 1 \end{bmatrix}$$

Nu komt een verwisseling van rijen 2 en 3 omdat het element 4 (in absolute waarde) het grootst mogelijke pivotelement is:

$$\mathbf{P}_{23} \begin{bmatrix} 3 & 8 & 1 \\ 0 & -\frac{2}{3} & \frac{2}{3} \\ 0 & 4 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 3 & 8 & 1 \\ 0 & -\frac{2}{3} & \frac{2}{3} \\ 0 & 4 & 1 \end{bmatrix} = \begin{bmatrix} 3 & 8 & 1 \\ 0 & 4 & 1 \\ 0 & -\frac{2}{3} & \frac{2}{3} \end{bmatrix}.$$

Tot slot volgt nog een elementaire rij-operatie:

$$\mathbf{E}_{32} \begin{bmatrix} 3 & 8 & 1 \\ 0 & 4 & 1 \\ 0 & -\frac{2}{3} & \frac{2}{3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & \frac{1}{6} & 1 \end{bmatrix} \begin{bmatrix} 3 & 8 & 1 \\ 0 & 4 & 1 \\ 0 & -\frac{2}{3} & \frac{2}{3} \end{bmatrix} = \begin{bmatrix} 3 & 8 & 1 \\ 0 & 4 & 1 \\ 0 & 0 & \frac{5}{6} \end{bmatrix} = \mathbf{U}. \quad (3.6)$$

Als we alles samenleggen dan vinden we de volgende gelijkheid:

$$\mathbf{E}_{32}\mathbf{P}_{23}\mathbf{E}_{21}\mathbf{P}_{13}\mathbf{A} = \mathbf{U}.$$

Omdat de permutatiematrices in dit geval steeds twee rijen van plaats wisselen zijn ze hun eigen inverse en kunnen we dit ook schrijven als:

$$(\mathbf{E}_{32}\mathbf{P}_{23}\mathbf{E}_{21}\mathbf{P}_{23})\mathbf{P}_{23}\mathbf{P}_{13}\mathbf{A} = \mathbf{U},$$

of ook als

$$(\mathbf{P}_{23}\mathbf{P}_{13})\mathbf{A} = (\mathbf{E}_{32}\mathbf{P}_{23}\mathbf{E}_{21}\mathbf{P}_{23})^{-1}\mathbf{U}.$$

We weten reeds wat de matrix  $\mathbf{U}$  is want die wordt gegeven door vergelijking (3.6). Verder berekenen we

$$\mathbf{P}_{23}\mathbf{P}_{13} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \mathbf{P},$$

en

$$\begin{aligned}
 (\mathbf{E}_{32}\mathbf{P}_{23}\mathbf{E}_{21}\mathbf{P}_{23})^{-1} &= \mathbf{P}_{23}^{-1}\mathbf{E}_{21}^{-1}\mathbf{P}_{23}^{-1}\mathbf{E}_{32}^{-1} \\
 &= \left( \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{3} & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \right) \left( \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -\frac{1}{6} & 1 \end{bmatrix} \right) \\
 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ \frac{1}{3} & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & -\frac{1}{6} & 1 \\ 0 & 1 & 0 \end{bmatrix} \\
 &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \frac{1}{3} & -\frac{1}{6} & 1 \end{bmatrix} \\
 &= \mathbf{L}.
 \end{aligned}$$

Finaal vinden we dus

$$\mathbf{PA} = \mathbf{LU},$$

waarbij  $\mathbf{P}$  een permutatiematrix is en  $\mathbf{L}$  en  $\mathbf{U}$  een beneden- en een bovendriehoeksmatrix zijn.

Het feit dat dit proces altijd werkt wanneer de matrix  $\mathbf{A}$  inverteerbaar is wordt samengevat in de volgende eigenschap.

**Eigenschap 3.5** Voor elke inverteerbare vierkante matrix  $\mathbf{A}$  bestaat er een permutatiematrix  $\mathbf{P}$ , een benedendriehoeksmatrix  $\mathbf{L}$  en een bovendriehoeksmatrix  $\mathbf{U}$  zodanig dat

$$\mathbf{PA} = \mathbf{LU}. \quad \blacksquare$$

**Opmerking 3.6** De procedure zoals hier voorgesteld levert steeds een  $\mathbf{L}$  waarvan alle elementen op de diagonaal gelijk zijn aan 1.  $\blacksquare$

### 3.3.3 Oefeningen

1. Vind (handmatig) de LU-decompositie van de volgende matrix.

$$\mathbf{A} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}.$$

2. In deze oefening manipuleren we elementaire matrices.

a) Verifieer dat

$$\begin{bmatrix} 1 & 0 & 0 \\ e_{21} & 1 & 0 \\ e_{31} & 0 & 1 \end{bmatrix}^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ -e_{21} & 1 & 0 \\ -e_{31} & 0 & 1 \end{bmatrix}$$

b) Verifieer dat

$$\begin{bmatrix} 1 & 0 & 0 \\ e_{21} & 1 & 0 \\ e_{31} & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & e_{32} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ e_{21} & 1 & 0 \\ e_{31} & e_{32} & 1 \end{bmatrix}.$$

3. Toon aan dat de volgende matrices geen LU-decompositie hebben. Toon in het bijzonder aan dat de voorgestelde decomposities geen oplossing hebben.

a)

$$\begin{bmatrix} 0 & 1 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ l & 1 \end{bmatrix} \begin{bmatrix} a & 0 \\ b & c \end{bmatrix}$$

b)

$$\begin{bmatrix} 1 & 1 & 0 \\ 1 & 1 & 2 \\ 2 & 3 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ l & 1 & 0 \\ m & n & 1 \end{bmatrix} \begin{bmatrix} a & b & c \\ 0 & d & e \\ 0 & 0 & f \end{bmatrix}$$

## 3.4 Toepassingen van de LU-decompositie

### 3.4.1 Oplossen van stelsels lineaire vergelijkingen

Eens je de LU-decompositie van een matrix  $\mathbf{A}$  berekend hebt dan kan je deze LU-decompositie gebruiken om stelsels lineaire vergelijkingen

$$\mathbf{Ax} = \mathbf{b} \tag{3.7}$$

op te lossen. Als we weten dat

$$\mathbf{PA} = \mathbf{LU}. \tag{3.8}$$

dan kunnen we in formule (3.7) linker- en rechterlid met  $\mathbf{P}$  vermenigvuldigen (omdat  $\mathbf{P}$  inverteerbaar is), en vervolgens gebruikmaken van formule (3.8) om het volgende te bekomen:

$$\mathbf{P}\mathbf{A}\mathbf{x} = \mathbf{P}\mathbf{b} \iff \mathbf{L}\mathbf{U}\mathbf{x} = \mathbf{P}\mathbf{b}.$$

Dan kunnen we eerst

$$\mathbf{L}\mathbf{y} = \mathbf{P}\mathbf{b}$$

oplossen naar  $\mathbf{y}$  door middel van VOORWAARTSE SUBSTITUTIE. Vervolgens los je

$$\mathbf{U}\mathbf{x} = \mathbf{y}$$

op door achterwaartse substitutie.

**Voorbeeld 3.7** We lossen het stelsel

$$\begin{bmatrix} 1 & 2 & 1 \\ 3 & 8 & 1 \\ 0 & 4 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 2 \\ 12 \\ 2 \end{bmatrix}$$

nogmaals op. We weten reeds uit vergelijking (3.5) dat de LU-decompositie van  $\mathbf{A}$  gegeven wordt door

$$\begin{bmatrix} 1 & 2 & 1 \\ 3 & 8 & 1 \\ 0 & 4 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \\ 0 & 2 & -2 \\ 0 & 0 & 5 \end{bmatrix}$$

In eerste instantie lossen we

$$\mathbf{L}\mathbf{y} = \mathbf{b} \iff \begin{bmatrix} 1 & 0 & 0 \\ 3 & 1 & 0 \\ 0 & 2 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 12 \\ 2 \end{bmatrix}$$

op m.b.v. voorwaartse substitutie. Uit de eerste vergelijking volgt dat  $y_1 = 2$ . Uit de tweede vergelijking volgt dat

$$y_2 = 12 - 3y_1 = 12 - 3 \times 2 = 6,$$

en de derde vergelijking zegt dat

$$y_3 = 2 - 2y_2 = 2 - 2 \times 6 = -10.$$

In tweede instantie moeten we

$$\mathbf{U}\mathbf{x} = \mathbf{y} \iff \begin{bmatrix} 1 & 2 & 1 \\ 0 & 2 & -2 \\ 0 & 0 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 2 \\ 6 \\ -10 \end{bmatrix}$$

oplossen m.b.v. achterwaartse substitutie. Dit stelsel hebben we reeds opgelost, het is namelijk gelijk aan het stelsel (3.3). ■

### Tijdscomplexiteit oplossen stelsels m.b.v. de LU-decompositie

We veronderstellen dat de coëfficiëntenmatrix een vierkante  $n \times n$  matrix is, en we wensen een inschatting te maken van het aantal floating-point bewerkingen dat men moet uitvoeren om een stelsel lineaire vergelijkingen op te lossen m.b.v. de LU-decompositie.

Wanneer er geen rijverwisselingen nodig zijn, dan zijn er  $n - 1$  stappen nodig.

- Bij de eerste stap zijn er  $n - 1$  rijen waarvoor er ongeveer  $2n$  floating-point operaties per rij nodig zijn. Een rij heeft immers lengte  $n$  en er moet een product en een verschil worden berekend per element op de rij.
- Bij de tweede stap zijn er  $n - 2$  rijen waarvoor er ongeveer  $2(n - 1)$  floating-point operaties nodig zijn. Merk op dat het eerste element van deze  $n - 2$  rijen reeds gelijk is aan nul, en dat deze elementen ook niet meer wijzigen. Dit is de reden waarom er nu minder floating-point operaties nodig zijn dan in de eerste stap.
- enzovoort.
- Bij stap  $n - 1$  is er nog 1 rij waarvoor er dan ongeveer 2 floating-point operaties nodig zijn.

Samen zijn er dus ongeveer

$$\sum_{k=1}^{n-1} \overbrace{(n-k)}^{\text{aantal rijen}} \times \underbrace{2(n-k+1)}_{\text{aantal bewerkingen per rij}} = \frac{2}{3}(n^3 - n)$$



floating point operaties nodig<sup>3</sup>. De tijdscomplexiteit voor de LU-decompositie is m.a.w. van de orde  $\mathcal{O}(n^3)$ .

Het oplossen van de stelsels m.b.v. voor- en achterwaartse substitutie hebben elk een tijdscomplexiteit van de orde  $\mathcal{O}(n^2)$ .

De tijdscomplexiteit voor het oplossen van een enkel stelsel wordt gedomineerd door de tijd nodig voor het bepalen van de LU-decompositie en is dus gelijk aan  $\mathcal{O}(n^3)$ .

### 3.4.2 Berekenen van de inverse matrix

De inverse matrix van een niet-singuliere vierkante matrix  $\mathbf{A} \in \mathbb{R}^{n \times n}$  is de matrix  $\mathbf{X}$  waarvoor geldt dat

$$\mathbf{A}\mathbf{X} = \mathbf{I}_n \iff \mathbf{A} \begin{bmatrix} | & | & & | \\ \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_n \\ | & | & & | \end{bmatrix} = \mathbf{I}_n,$$

waarbij we de matrix  $\mathbf{X}$  hebben uitgeschreven in termen van zijn kolommen. Dit betekent dat

$$\mathbf{A}\mathbf{x}_i = i\text{-de kolom van } \mathbf{I}_n.$$

Het vinden van de inverse matrix van  $\mathbf{A}$  komt m.a.w. neer op het oplossen van  $n$  stelsels lineaire vergelijkingen waarvan de coëfficiëntenmatrix telkens gelijk is aan  $\mathbf{A}$ , en waarbij de rechterleden bestaan uit de kolommen van de eenheidsmatrix.

Gebruikmakend van de LU-decompositie van  $\mathbf{A}$  kan men deze  $n$  stelsels oplossen om de inverse matrix van  $\mathbf{A}$  te bekomen.

De tijdscomplexiteit van de LU-decompositie is  $\mathcal{O}(n^3)$ . Oplossen van één stelsel lineaire vergelijkingen met gekende LU-decompositie heeft een tijdscomplexiteit  $\mathcal{O}(n^2)$ , waardoor het oplossen van  $n$  stelsels van de orde  $\mathcal{O}(n^3)$  wordt. De totale tijdscomplexiteit voor het bepalen van de inverse matrix is op deze manier  $\mathcal{O}(n^3)$ .

<sup>3</sup>Deze som bereken je gemakkelijk m.b.v. een programma als Wolfram Alpha.

### 3.4.3 Berekenen van de determinant

#### Determinant m.b.v. elementaire rijoperaties

De DETERMINANT van een matrix is een getal dat met elke vierkante matrix wordt geassocieerd en dat heel wat informatie omtrent de matrix als geheel bevat. De determinant van een matrix kan op twee verschillende manieren worden genoteerd. De eerste manier is als

$$\det \mathbf{A},$$

de tweede manier is met twee verticale lijnen rond de elementen van de matrix, m.a.w.

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} \text{ is de determinant van de matrix } \begin{bmatrix} a & b \\ c & d \end{bmatrix}.$$

De determinant kan worden gedefinieerd als een afbeelding van vierkante matrices naar getallen die voldoet aan de volgende drie regels:

- De determinant van de eenheidsmatrix  $\mathbf{I}_n$  is gelijk aan 1.
- Wanneer men twee rijen van een matrix wisselt, dan keert het teken van de determinant om.
- De derde eigenschap bestaat uit twee delen. Wanneer men de eerste rij van een determinant vermenigvuldigt met een constante factor, dan wordt de determinant met die factor vermenigvuldigd. Bv.

$$\begin{vmatrix} t a & t b \\ c & d \end{vmatrix} = t \begin{vmatrix} a & b \\ c & d \end{vmatrix}.$$

Verder is de determinant een lineaire functie in de eerste rij.

$$\begin{vmatrix} a + a' & b + b' \\ c & d \end{vmatrix} = \begin{vmatrix} a & b \\ c & d \end{vmatrix} + \begin{vmatrix} a' & b' \\ c & d \end{vmatrix}.$$

Aan de hand van deze drie definiërende eigenschappen bewijst men alle andere eigenschappen die je misschien ooit over determinanten hebt gezien, zoals het feit dat een determinant gelijk is aan nul als die twee gelijke rijen bevat. Inderdaad: wegens de tweede eigenschap van determinanten moet het teken omwisselen als men de twee gelijke rijen van plaats wisselt, maar

aangezien de determinant niet echt verandert (omdat de twee rijen gelijk zijn) moet de determinant gelijk blijven. De enige mogelijkheid is bijgevolg dat de determinant gelijk is aan nul.

Een andere eigenschap die men kan aantonen is dat de determinant niet wijzigt als men bij een rij een veelvoud van een andere rij optelt. Bv.

$$\begin{aligned}
 \begin{vmatrix} a+tc & b+td \\ c & d \end{vmatrix} &= \begin{vmatrix} a & b \\ c & d \end{vmatrix} + \begin{vmatrix} tc & td \\ c & d \end{vmatrix} && \text{derde eigenschap determinanten} \\
 &= \begin{vmatrix} a & b \\ c & d \end{vmatrix} + t \begin{vmatrix} c & d \\ c & d \end{vmatrix} && \text{derde eigenschap determinanten} \\
 &= \begin{vmatrix} a & b \\ c & d \end{vmatrix} + 0 && \text{determinant met gelijke rijen is nul} \\
 &= \begin{vmatrix} a & b \\ c & d \end{vmatrix} && \text{neutraal element optelling}
 \end{aligned}$$

Dit betekent dat de elementaire rijoperaties die men uitvoert om de LU-decompositie te bekomen de determinant niet wijzigen, met uitzondering van het wisselen van twee rijen dat een tekenwisseling in de determinant veroorzaakt. Deze procedure kan men gebruiken om de determinant van een vierkante matrix te bepalen, als men ook gebruikmaakt van het feit dat de determinant van een boven- of benedendriehoeksmatrix gelijk is aan het product van de elementen op de diagonaal.

### Determinant m.b.v. LU-decompositie

Als men de LU-decompositie van een matrix reeds heeft dan kan men gebruikmaken van de belangrijke eigenschap (die weliswaar iets moeilijker te bewijzen is) dat de determinant van het product van vierkante matrices gelijk is aan het product van de determinanten:

$$\det \mathbf{AB} = \det \mathbf{A} \det \mathbf{B}.$$

Als we dit nu toepassen op de LU-decompositie van een vierkante matrix  $\mathbf{A}$ , dan vinden we dat de determinant van  $\mathbf{A}$  gelijk is aan het product van de determinanten van  $\mathbf{L}$  en  $\mathbf{U}$ . De determinant van  $\mathbf{L}$  is gelijk aan 1 (omdat op de diagonaal enkel enen staan), en de determinant van  $\mathbf{U}$  is gelijk aan het product van de diagonaalelementen.

In het geval men ook nog een permutatiematrix  $\mathbf{P}$  in rekening brengt, moet er enkel nog gelet worden op het aantal rijverwisselingen dat er uitgevoerd werd.

### 3.4.4 Oefeningen

1. Schrijf een Python-methode `los_vierkant_stelsel_op(A, b)` om een stelsel lineaire vergelijkingen met evenveel vergelijkingen als onbekenden op te lossen. Maak gebruik van de `scipy`-methode `scipy.linalg.lu` om de LU-decompositie te bekomen, maar schrijf zelf de methodes om voor- en achterwaartse substitutie uit te voeren. Controleer je methode door een aantal testen uit te voeren.
2. Schrijf een Python-methode `bereken_inverse_matrix(A)` om de inverse matrix van een vierkante matrix te bepalen. Maak opnieuw gebruik van de `scipy`-methode `scipy.linalg.lu` en van je zelfgeschreven methodes voor voor- en achterwaartse substitutie.
3. Schrijf een Python-methode `bereken_determinant(A)` om de determinant van een vierkante matrix te bepalen. Implementeer deze methode zonder gebruik te maken van ingebouwde methodes. Doe rijverwisselingen om je methode robuust te maken wanneer nullen (of kleine getallen) worden ontmoet op de diagonaal.

## 3.5 Oplossen van strijdige stelsels

Wanneer het aantal vergelijkingen in een lineair stelsel groter is dan het aantal onbekenden, dan bestaat er *meestal* geen oplossing voor dit stelsel.

**Voorbeeld 3.8** Het volgende stelsel heeft geen oplossing:

$$\begin{cases} x_1 + x_2 = 1 \\ x_1 + 2x_2 = 2 \\ x_1 + 3x_2 = 2. \end{cases}$$

De coëfficiëntenmatrix en het rechterlid van dit stelsel zijn respectievelijk

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \quad \text{en} \quad \mathbf{b} = \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix}.$$

Het is duidelijk dat geen enkele lineaire combinatie van de kolommen van  $\mathbf{A}$  het rechterlid kan opleveren. Inderdaad, uit de laatste twee vergelijkingen volgt onmiddellijk dat  $x_1 = 2$  en  $x_2 = 0$  nul zou moeten zijn, maar dit klopt dan niet voor de eerste vergelijking. ■

In plaats van “op te geven” en te zeggen dat zo’n strijdig stelsel geen oplossing heeft, kunnen we ook proberen om een benadering  $\mathbf{x}^*$  te vinden die er voor zorgt dat

$$\mathbf{A}\mathbf{x}^*$$

zo dicht als mogelijk bij  $\mathbf{b}$  ligt. Aangezien  $\mathbf{A}\mathbf{x}^*$  behoort tot de kolomruimte van  $\mathbf{A}$  zoeken we de vector in deze kolomruimte die zo dicht mogelijk bij  $\mathbf{b}$  ligt. *Dit is uiteraard niets anders dan de orthogonale projectie van  $\mathbf{b}$  op de vectorruimte opgespannen door de kolommen van  $\mathbf{A}$ .*

We weten dat  $\mathbf{A}^T\mathbf{A}$  een belangrijke rol speelt bij orthogonale projectie, dus kunnen we linker- en rechterlid van

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

langs links vermenigvuldigen met  $\mathbf{A}^T$ :

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b}.$$

Als we aannemen dat de kolommen van  $\mathbf{A}$  lineair onafhankelijk zijn dan is  $\mathbf{A}^T\mathbf{A}$  inverteerbaar<sup>4</sup> en vinden we

$$\mathbf{x}^* = (\mathbf{A}^T\mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}.$$

Merk op dat

$$\mathbf{A}\mathbf{x}^* = \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}$$

inderdaad de orthogonale projectie is van  $\mathbf{b}$  op de kolomruimte van  $\mathbf{A}$ , want we herkennen hierin de toepassing van de projectiematrix uit vergelijking (2.5).

**Voorbeeld 3.9** We werken verder met het stelsel gegeven in Voorbeeld 3.8. Als we  $\mathbf{A}^T\mathbf{A}$  berekenen, dan vinden we

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 6 \\ 6 & 14 \end{bmatrix}$$

<sup>4</sup>Dit werd reeds in Sectie 2.5 vermeld zonder bewijs.

De inverse van deze matrix is

$$\begin{bmatrix} 3 & 6 \\ 6 & 14 \end{bmatrix}^{-1} = \frac{1}{6} \begin{bmatrix} 14 & -6 \\ -6 & 3 \end{bmatrix},$$

zodat de benaderende oplossing gegeven wordt door:

$$\frac{1}{6} \begin{bmatrix} 14 & -6 \\ -6 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix} = \frac{1}{6} \begin{bmatrix} 14 & -6 \\ -6 & 3 \end{bmatrix} \begin{bmatrix} 5 \\ 11 \end{bmatrix} = \begin{bmatrix} 2/3 \\ 1/2 \end{bmatrix}.$$

Bij deze benaderende oplossing hoort het rechterlid

$$\begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 2/3 \\ 1/2 \end{bmatrix} = \begin{bmatrix} 7/6 \\ 10/6 \\ 13/6 \end{bmatrix}.$$

Dit is (uiteraard) niet gelijk aan het originele rechterlid, maar het is, van alle rechterleden die men wel kan bereiken met de kolommen uit **A**, diegene die het dichtst bij het originele rechterlid ligt. ■

### 3.5.1 Toepassing: kleinste kwadraten methode

#### Best passende rechte in het vlak

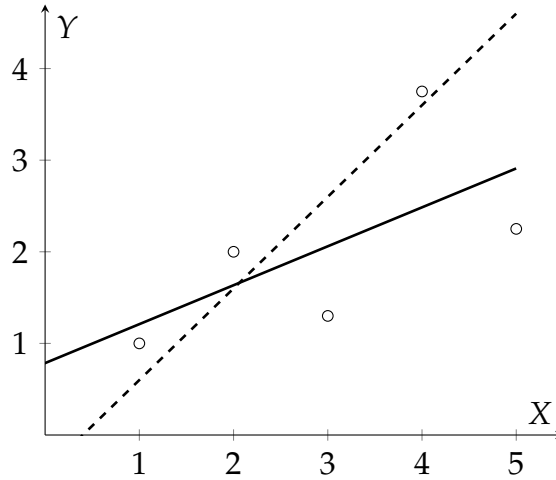
In veel data-analyses tracht men een “best passende” rechte te vinden door een aantal datapunten

$$(x^{(i)}, y^{(i)}) \quad \text{met} \quad i \in \{1, 2, \dots, m\},$$

waarbij we nu voor de eenvoud aannemen dat zowel de  $x^{(i)}$  als  $y^{(i)}$  reële getallen zijn.

**Voorbeeld 3.10** Hieronder wordt een dataset bestaande uit vijf datapunten getoond.

$i$	$x^{(i)}$	$y^{(i)}$
1	1.0	1.0
2	2.0	2.0
3	3.0	1.3
4	4.0	3.75
5	5.0	2.25



**Figuur 3.1:** Een dataset bestaande uit vijf voorbeelden. De rechte in stippellijn heeft vergelijking  $y = -0.4 + x$ , de rechte in volle lijn heeft als vergelijking  $y = 0.785 + 0.425x$ .

Deze dataset wordt afgebeeld in Figuur 3.1. ■

Een rechte in het vlak heeft de volgende vergelijking:

$$y = \theta_0 + \theta_1 x,$$

waarbij  $\theta_0$  en  $\theta_1$  vrij te kiezen zijn. Voor de datapunten vinden we op die manier de voorspelde waarden:

$$\hat{y}^{(i)} = \theta_0 + \theta_1 x^{(i)}, \quad \text{voor } i \in \{1, 2, \dots, m\},$$

waarbij we een “hoedje” plaatsen boven de  $y$  om aan te geven dat het gaat over een voorspelde waarde.

Om te meten hoe goed een rechte (bepaald door  $\theta_0$  en  $\theta_1$ ) past bij de gegeven datapunten wordt traditioneel de MEAN SQUARED ERROR gebruikt, wat de gemiddelde kwadratische afwijking is tussen de waarden  $y^{(i)}$  en de voorspelde waarde  $\hat{y}^{(i)}$ :

$$\text{MSE} = \frac{1}{m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2. \quad (3.9)$$

De KLEINSTE KWADRATEN OPLOSSING is die rechte waarvoor de MSE minimaal is.

**Voorbeeld 3.11 (Berekening MSE)** We geven een voorbeeld van de berekening van de MSE voor gegeven vaste waarden  $\theta_0 = -0.4$  en  $\theta_1 = 1$ . We voegen extra kolommen toe aan de tabel om de berekening te illustreren:

$i$	$x^{(i)}$	$y^{(i)}$	$\hat{y}^{(i)} = -0.4 + x^{(i)}$	$(\hat{y}^{(i)} - y^{(i)})^2$
1	1.0	1.0	$-0.4 + 1.0 = 0.6$	$(0.6 - 1.0)^2$
2	2.0	2.0	$-0.4 + 2.0 = 1.6$	$(1.6 - 2.0)^2$
3	3.0	1.3	$-0.4 + 3.0 = 2.6$	$(2.6 - 1.3)^2$
4	4.0	3.75	$-0.4 + 4.0 = 3.6$	$(3.6 - 3.75)^2$
5	5.0	2.25	$-0.4 + 5.0 = 4.6$	$(4.6 - 2.25)^2$

De MSE is het gemiddelde van de getallen in de laatste kolom:

$$\text{MSE} = \frac{1}{5} ((-0.4)^2 + (-0.4)^2 + 1.3^2 + (-0.15)^2 + 2.35^2) = 1.511.$$

De rechte horend bij deze waarden van  $\theta_0$  en  $\theta_1$  is afgebeeld in Figuur 3.1. Het is duidelijk dat als men de waarden voor  $\theta_0$  en  $\theta_1$  wijzigt dat dan de waarde van de MSE in het algemeen ook wijzigt. De bedoeling is om dié waarden van  $\theta_0$  en  $\theta_1$  te vinden waarvoor de MSE zo klein mogelijk is. ■

We formuleren het probleem van het minimaliseren van de MSE als een probleem in lineaire algebra. Als we de volgende vectoren definiëren:

$$\mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \quad \text{en} \quad \hat{\mathbf{y}} = \begin{bmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ \vdots \\ \hat{y}^{(m)} \end{bmatrix},$$

dan is

$$\text{MSE} = \frac{1}{m} \|\hat{\mathbf{y}} - \mathbf{y}\|^2.$$

We tonen nu aan dat de vector  $\hat{\mathbf{y}}$  een lineaire combinatie is van de kolommen van de DATAMATRIX  $\mathbf{X}$ , waarbij de rijen van  $\mathbf{X}$  de  $x$ -waarden van de datapunten bevatten (aangevuld met een kolom bestaande uit uitsluitend de waarde 1). De datamatrix  $\mathbf{X}$  wordt dus gegeven door:

$$\mathbf{X} = \begin{bmatrix} 1 & x^{(1)} \\ 1 & x^{(2)} \\ \vdots & \vdots \\ 1 & x^{(m)} \end{bmatrix}.$$



Inderdaad, als we de vector

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix}$$

beschouwen, dan is

$$\mathbf{X}\boldsymbol{\theta} = \hat{\mathbf{y}}.$$

De beste vector  $\hat{\mathbf{y}}$  is m.a.w. niets anders dan de orthogonale projectie van  $\mathbf{y}$  op  $\text{Col}(\mathbf{X})$ . De coëfficiënten  $\theta_0$  en  $\theta_1$  die deze beste vector  $\hat{\mathbf{y}}$  opleveren worden m.a.w. gegeven door

$$\boldsymbol{\theta} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}.$$

Dit is een toepassing van formule (2.4). De bijhorende voorspellingen  $\hat{\mathbf{y}}$  worden dan gegeven door

$$\hat{\mathbf{y}} = \mathbf{X}\boldsymbol{\theta} = \mathbf{X}(\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{y}.$$

**Voorbeeld 3.12** We berekenen de best passende rechte door de vijf datapunten gegeven in Voorbeeld 3.10. We stellen de datamatrix  $\mathbf{X}$  en vector  $\mathbf{y}$  op:

$$\mathbf{X} = \begin{bmatrix} 1 & 1.0 \\ 1 & 2.0 \\ 1 & 3.0 \\ 1 & 4.0 \\ 1 & 5.0 \end{bmatrix} \quad \text{en} \quad \mathbf{y} = \begin{bmatrix} 1.0 \\ 2.0 \\ 1.3 \\ 3.75 \\ 2.25 \end{bmatrix}.$$

Vervolgens berekenen we het matrixproduct  $\mathbf{X}^T \mathbf{X}$ :

$$\mathbf{X}^T \mathbf{X} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1.0 & 2.0 & 3.0 & 4.0 & 5.0 \end{bmatrix} \begin{bmatrix} 1 & 1.0 \\ 1 & 2.0 \\ 1 & 3.0 \\ 1 & 4.0 \\ 1 & 5.0 \end{bmatrix} = \begin{bmatrix} 5.0 & 15.0 \\ 15.0 & 55.0 \end{bmatrix}.$$

De inverse matrix wordt gegeven door

$$(\mathbf{X}^T \mathbf{X})^{-1} = \frac{1}{50} \begin{bmatrix} 55.0 & -15.0 \\ -15.0 & 5.0 \end{bmatrix}.$$

Je rekent ook gemakkelijk na dat

$$\mathbf{X}^T \mathbf{y} = \begin{bmatrix} 10.3 \\ 35.15 \end{bmatrix},$$

zodat finaal de beste waarden voor  $\theta_0$  en  $\theta_1$  gevonden worden door

$$\begin{bmatrix} \theta_0 \\ \theta_1 \end{bmatrix} = \frac{1}{50} \begin{bmatrix} 55.0 & -15.0 \\ -15.0 & 5.0 \end{bmatrix} \begin{bmatrix} 10.3 \\ 35.15 \end{bmatrix} = \begin{bmatrix} 0.785 \\ 0.425 \end{bmatrix}.$$

De bijhorende vector  $\hat{\mathbf{y}}$  van voorspelde waarden is:

$$\hat{\mathbf{y}} = \begin{bmatrix} 1 & 1.0 \\ 1 & 2.0 \\ 1 & 3.0 \\ 1 & 4.0 \\ 1 & 5.0 \end{bmatrix} \begin{bmatrix} 0.785 \\ 0.425 \end{bmatrix} = \begin{bmatrix} 1.21 \\ 1.635 \\ 2.06 \\ 2.485 \\ 2.91 \end{bmatrix}.$$

Deze 5 waarden liggen allemaal op de best passende rechte met vergelijking

$$y = 0.785 + 0.425x.$$

■

### Best passend hypervlak in hogere dimensies

Een volledig analoge redenering is mogelijk wanneer een datapunt niet uit één enkele reële waarde bestaat, maar gegeven wordt door  $n$  numerieke waarden  $x_1$  t.e.m.  $x_n$ . Een voorspelling voor het  $i$ -de datapunt wordt dan als volgt bepaald

$$\hat{y}^{(i)} = \theta_0 + \theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \cdots + \theta_n x_n^{(i)}.$$

We kunnen de datamatrix  $\mathbf{X}$  op dezelfde manier opbouwen

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & x_n^{(1)} \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & x_n^{(2)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & x_1^{(m)} & x_2^{(m)} & \cdots & x_n^{(m)} \end{bmatrix} \in \mathbb{R}^{m \times (n+1)}.$$

De matrix  $\mathbf{X}^T \mathbf{X}$  is dan een vierkante matrix met  $n + 1$  rijen en kolommen. Deze matrix moet geïnverteerd worden om de projectie van de vector  $\mathbf{y}$  te vinden op kolomruimte van  $\mathbf{X}$ . Dit inverteren vereist dat de kolommen van  $\mathbf{X}$  lineair onafhankelijk zijn. In het geval de kolommen van  $\mathbf{X}$  lineair afhankelijk zijn, dan zal dit proces falen en moet een andere methode worden gebruikt, die we in een later hoofdstuk zullen behandelen.

### 3.5.2 Oefeningen

1. Beschouw het stelsel lineaire vergelijkingen:

$$\begin{cases} 2x + y = 0 \\ x - y - 1 = 0 \\ x + y + 1 = 0. \end{cases}$$

- a) Geef de coëfficiëntenmatrix en het rechterlid.
- b) Gebruik Gaussische eliminatie om vast te stellen dat het een strijdig stelsel is.
- c) Geef de beste benaderende oplossing voor dit stelsel door de methode uit deze sectie toe te passen. Wat is deze oplossing? Wat is het bijhorende rechterlid?

## Orthogonale matrices

In dit hoofdstuk bespreken we een belangrijke klasse van matrices die in de praktijk zeer bruikbaar zijn en straks ook een grote rol spelen bij de belangrijkste van alle matrixdecomposities.

### 4.1 Orthonormale vectoren

We noemen een verzameling vectoren  $\mathbf{q}_1$  t.e.m.  $\mathbf{q}_m$  ORTHONORMAAL als elk paar vectoren orthogonaal is én als de lengte van elke vector gelijk is aan 1. In symbolen:

$$\mathbf{q}_i \cdot \mathbf{q}_j = 0 \quad \text{als } i \neq j$$

en

$$\mathbf{q}_i \cdot \mathbf{q}_i = \|\mathbf{q}_i\|^2 = 1, \quad \text{voor } i \in \{1, 2, \dots, m\}.$$

**Voorbeeld 4.1** De volgende vectoren:

$$\mathbf{q}_1 = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{q}_2 = \begin{bmatrix} -1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \\ 0 \end{bmatrix} \quad \text{en} \quad \mathbf{q}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

is een verzameling orthonormale vectoren. ■

Aangezien orthonormale vectoren loodrecht op elkaar staan kan je verwachten dat ze lineair onafhankelijk zullen zijn. Dit blijkt ook uit de volgende eigenschap.

**Eigenschap 4.2** Een verzameling orthonormale vectoren is steeds lineair onafhankelijk. ■

*Bewijs* Veronderstel dat  $\mathbf{q}_1$  t.e.m.  $\mathbf{q}_m$  orthonormale vectoren zijn. Beschouw een lineaire combinatie van deze vectoren die de nulvector oplevert:

$$\alpha_1 \mathbf{q}_1 + \alpha_2 \mathbf{q}_2 + \cdots + \alpha_m \mathbf{q}_m = \mathbf{0}.$$

We moeten volgens Definitie 1.5 aantonen dat alle coëfficiënten  $\alpha_i$  gelijk zijn aan nul.

Om aan te tonen dat  $\alpha_i$  gelijk is aan nul nemen we het inwendig product van beide leden met  $\mathbf{q}_i$ :

$$\begin{aligned} \mathbf{q}_i \cdot (\alpha_1 \mathbf{q}_1 + \alpha_2 \mathbf{q}_2 + \cdots + \alpha_m \mathbf{q}_m) &= \mathbf{q}_i \cdot \mathbf{0} \quad \text{inwendig product is lineair} \\ \iff \alpha_1 \mathbf{q}_i \cdot \mathbf{q}_1 + \alpha_2 \mathbf{q}_i \cdot \mathbf{q}_2 + \cdots & \\ + \alpha_i \mathbf{q}_i \cdot \mathbf{q}_i + \cdots + \alpha_m \mathbf{q}_i \cdot \mathbf{q}_m &= 0 \quad \text{orthonormaliteit} \\ \iff 0 + 0 + \cdots + \alpha_i + \cdots + 0 &= 0 \\ \iff \alpha_i &= 0. \quad \diamond \end{aligned}$$

Orthonormale vectoren zijn in de praktijk zeer handig om mee te rekenen omdat allerhande bewerkingen heel eenvoudig worden. Stel bv. dat  $\mathbf{q}_1$  t.e.m.  $\mathbf{q}_m$  een orthonormale basis is voor één of andere vectorruimte. Stel dat de vector  $\mathbf{a}$  tot die vectorruimte behoort. De coördinaten van  $\mathbf{a}$  t.o.v. de basis  $\mathbf{q}_1$  t.e.m.  $\mathbf{q}_m$  zijn de getallen  $a_1$  t.e.m.  $a_m$  zodat

$$\mathbf{a} = a_1 \mathbf{q}_1 + a_2 \mathbf{q}_2 + \cdots + a_m \mathbf{q}_m.$$

In het algemeen zou men een stelsel lineaire vergelijkingen moeten oplossen om de getallen  $a_1$  t.e.m.  $a_m$  te bepalen, maar hier kunnen we om  $a_i$  te bepalen eenvoudigweg het inwendig product nemen met  $\mathbf{q}_i$ . Op één na alle termen zijn gelijk aan nul en we vinden:

$$\mathbf{q}_i \cdot \mathbf{a} = a_i \mathbf{q}_i \cdot \mathbf{q}_i = a_i.$$

Dit toont de volgende eigenschap aan.

**Eigenschap 4.3** Als  $\mathbf{q}_1$  t.e.m.  $\mathbf{q}_m$  een orthonormale basis is voor een vectorruimte, dan worden de coördinaten t.o.v. die basis van een vector  $\mathbf{a}$  behorend tot die vectorruimte gegeven door de inwendige producten

$$\mathbf{q}_i \cdot \mathbf{a} \quad \text{voor } i \in \{1, 2, \dots, m\}.$$

Anders gezegd:

$$\mathbf{a} = (\mathbf{q}_1 \cdot \mathbf{a})\mathbf{q}_1 + (\mathbf{q}_2 \cdot \mathbf{a})\mathbf{q}_2 + \cdots + (\mathbf{q}_m \cdot \mathbf{a})\mathbf{q}_m. \quad \blacksquare$$

### 4.1.1 Oefeningen

1. Bepaal de coëfficiënten  $a$  en  $b$  zodat

$$\mathbf{q}_1 = a \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \text{en} \quad \mathbf{q}_2 = b \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}$$

eenheidsvectoren zijn. Verifieer bovendien dat deze vectoren orthogonaal zijn. Bepaal tenslotte de coördinaten van

$$\mathbf{v} = \begin{bmatrix} 2 \\ 3 \\ 2 \end{bmatrix}$$

t.o.v. deze orthonormale basis. (Je kan verifiëren dat  $\mathbf{v}$  inderdaad in de deelruimte ligt opgespannen door de twee gegeven orthonormale vectoren.)

## 4.2 Orthogonale matrices

Als men een aantal orthonormale vectoren als kolommen toevoegt aan een matrix  $\mathbf{Q}$  dan heeft deze matrix de interessante eigenschap dat

$$\mathbf{Q}^T \mathbf{Q} = \mathbf{I}.$$

De reden is dat op positie  $(i, j)$  van  $\mathbf{Q}^T \mathbf{Q}$  niets anders staat dan het inwendig product van  $\mathbf{q}_i$  en  $\mathbf{q}_j$ :

$$\mathbf{Q}^T \mathbf{Q} = \begin{bmatrix} \text{---} & \mathbf{q}_1^T & \text{---} \\ \text{---} & \mathbf{q}_2^T & \text{---} \\ \text{---} & \vdots & \text{---} \\ \text{---} & \mathbf{q}_m^T & \text{---} \end{bmatrix} \begin{bmatrix} | & | & \cdots & | \\ \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_m \\ | & | & \cdots & | \end{bmatrix} = \mathbf{I}.$$

Deze redenering geldt ook in de andere richting: als  $\mathbf{Q}^T \mathbf{Q} = \mathbf{I}$  dan vormen de kolommen van  $\mathbf{Q}$  een orthonormale verzameling vectoren.

**Voorbeeld 4.4** Beschouw de matrix  $\mathbf{Q}$

$$\mathbf{Q} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \\ 0 & 0 \end{bmatrix}.$$

Je kan als volgt verifiëren dat de kolommen inderdaad twee orthonormale vectoren zijn. Bereken

$$\mathbf{Q}^T \mathbf{Q} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1/\sqrt{2} & -1/\sqrt{2} & 0 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}.$$

Het feit dat  $\mathbf{Q}^T \mathbf{Q} = \mathbf{I}$  toont aan dat de kolommen van  $\mathbf{Q}$  orthonormaal zijn.■

Het is heel belangrijk om op te merken dat in het algemeen

$$\mathbf{Q}\mathbf{Q}^T \neq \mathbf{I}.$$

Enkel wanneer de matrix  $\mathbf{Q}$  een vierkante matrix is geldt dat

$$\mathbf{Q}\mathbf{Q}^T = \mathbf{I}.$$

**Voorbeeld 4.5** ( $\mathbf{Q}\mathbf{Q}^T \neq \mathbf{I}$ ) Bekijk de matrix  $\mathbf{Q}$  uit het vorige voorbeeld:

$$\mathbf{Q} = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ 1/\sqrt{2} & -1/\sqrt{2} \\ 0 & 0 \end{bmatrix}.$$

Dan geldt

$$\mathbf{Q}\mathbf{Q}^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \neq \mathbf{I}.$$

De rijen van  $\mathbf{Q}$  vormen m.a.w. *geen* orthonormale verzameling. Dit is logisch want 3 vectoren met slechts 2 componenten elk kunnen niet lineair onafhankelijk zijn, en kunnen bijgevolg ook niet orthonormaal zijn.

Wanneer we een tweede matrix  $\mathbf{Q}_1$  beschouwen met een extra orthonormale kolom

$$\mathbf{Q}_1 = \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} & 0 \\ 1/\sqrt{2} & -1/\sqrt{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Dan geldt wel

$$\mathbf{Q}_1^T \mathbf{Q}_1 = \mathbf{I} = \mathbf{Q}_1 \mathbf{Q}_1^T \quad \blacksquare$$

**Definitie 4.6** Een ORTHOGONALE MATRIX<sup>1</sup>  $\mathbf{Q}$  is een vierkante matrix waarvan de kolommen een orthonormale verzameling vormen. Voor zo'n matrix geldt dat

$$\mathbf{Q}^T \mathbf{Q} = \mathbf{I} = \mathbf{Q} \mathbf{Q}^T. \quad \blacksquare$$

Hieruit volgt onmiddellijk de volgende eigenschap.

**Eigenschap 4.7** Een orthogonale matrix is m.a.w. inverteerbaar en de *inverse matrix* is gelijk aan de *getransponeerde matrix*:

$$\mathbf{Q}^{-1} = \mathbf{Q}^T. \quad \blacksquare$$

Dit is een bijzonder groot voordeel, omdat het berekenen van de getransponeerde matrix een zeer eenvoudige en efficiënte berekening is, in tegenstelling tot de algemene berekening die nodig is voor het vinden van een inverse matrix.

De lineaire transformatie geassocieerd met een orthogonale matrix behoudt lengtes en hoeken (zie Eigenschap (4.8) hierna), wat ervoor zorgt dat berekeningen met orthogonale matrices typisch problemen van numerieke instabiliteit vermijden.

**Eigenschap 4.8** Als  $\mathbf{Q} \in \mathbb{R}^{n \times n}$  een orthogonale matrix is dan geldt voor alle vectoren  $\mathbf{x}$  en  $\mathbf{y}$  in  $\mathbb{R}^n$  dat

$$\|\mathbf{x}\| = \|\mathbf{Q}\mathbf{x}\|$$

en

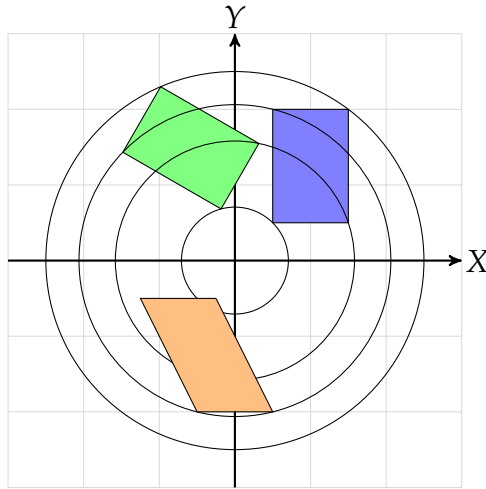
$$\mathbf{x} \cdot \mathbf{y} = \mathbf{Q}\mathbf{x} \cdot \mathbf{Q}\mathbf{y}. \quad \blacksquare$$

*Bewijs* We bewijzen de tweede eigenschap omdat de eerste hieruit volgt door  $\mathbf{x} = \mathbf{y}$  te kiezen. Het inwendig product kunnen we schrijven als een matrixproduct van een rijmatrix en een kolommatrix:

$$\begin{aligned} \mathbf{Q}\mathbf{x} \cdot \mathbf{Q}\mathbf{y} &= (\mathbf{Q}\mathbf{x})^T \mathbf{Q}\mathbf{y} && \text{inwendig product als product van matrices} \\ &= (\mathbf{x}^T \mathbf{Q}^T) \mathbf{Q}\mathbf{y} && \text{getransponeerde product} \\ &= \mathbf{x}^T (\mathbf{Q}^T \mathbf{Q}) \mathbf{y} && \text{matrixproduct is associatief} \\ &= \mathbf{x}^T \mathbf{I} \mathbf{y} && \mathbf{Q} \text{ is orthogonaal} \\ &= \mathbf{x}^T \mathbf{y} && \text{eenheidsmatrix is neutraal element} \\ &= \mathbf{x} \cdot \mathbf{y}. && \text{inwendig product als matrixproduct} \quad \diamond \end{aligned}$$

<sup>1</sup>Een meer correcte naam was wellicht "orthonormale" matrix geweest, maar dat is niet de algemeen gebruikte term.





**Figuur 4.1:** Voorbeeld van een orthogonale lineaire transformatie, nl. een rotatie over 60 graden. De blauwe rechthoek wordt getransformeerd in de groene rechthoek. Afstanden en hoeken blijven bewaard. Bemerk het verschil met een niet-orthogonale transformatie met matrix

$$\mathbf{A} = \begin{bmatrix} -1 & 1/2 \\ 0 & -1 \end{bmatrix}, \text{ waarbij afstanden noch hoeken worden bewaard.}$$

**Opmerking 4.9** Het feit dat hoeken worden behouden volgt uit het feit dat

$$\mathbf{x} \cdot \mathbf{y} = \|\mathbf{x}\| \|\mathbf{y}\| \cos(\theta),$$

waarbij  $\theta$  de hoek voorstelt ingesloten door  $\mathbf{x}$  en  $\mathbf{y}$ . Bovendien geldt

$$\mathbf{Qx} \cdot \mathbf{Qy} = \|\mathbf{Qx}\| \|\mathbf{Qy}\| \cos(\phi) = \|\mathbf{x}\| \|\mathbf{y}\| \cos(\phi),$$

waarbij  $\phi$  de hoek voorstelt ingesloten door  $\mathbf{Qx}$  en  $\mathbf{Qy}$ . Uit de voorgaande twee vergelijkingen volgt dus dat

$$\cos(\theta) = \cos(\phi). \quad \blacksquare$$

### 4.2.1 Oefeningen

1. Toon aan dat de determinant van een orthogonale matrix ofwel  $+1$  ofwel  $-1$  is.
2. Toon aan dat het product van twee orthogonale matrices in  $\mathbb{R}^{n \times n}$  opnieuw een orthogonale matrix is.

### 4.3 Projectie op orthonormale basis

We zagen reeds in Sectie 2.5 dat projectie op een deelruimte bepaald wordt door een projectiematrix  $\mathbf{P}$ . Als de vectoren  $\mathbf{a}_1$  t.e.m.  $\mathbf{a}_m$  tot  $\mathbb{R}^n$  behoren en lineair onafhankelijk zijn, dan behoort de projectiematrix  $\mathbf{P}$  tot  $\mathbb{R}^{n \times n}$  en wordt gegeven door

$$\mathbf{P} = \mathbf{A}(\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T,$$

waarbij de kolommen van  $\mathbf{A}$  de vectoren  $\mathbf{a}_1$  t.e.m.  $\mathbf{a}_m$  zijn. Het bepalen van de projectiematrix  $\mathbf{P}$  vereist het berekenen van een inverse matrix, en we weten reeds dat dit een “dure” bewerking is.

Als  $\mathbf{q}_1$  t.e.m.  $\mathbf{q}_m$  een orthonormale basis is van de deelruimte waarop we willen projecteren, dan vereenvoudigt de projectiematrix:

$$\mathbf{P} = \mathbf{Q}(\mathbf{Q}^T \mathbf{Q})^{-1} \mathbf{Q}^T = \mathbf{Q}(\mathbf{I})^{-1} \mathbf{Q}^T = \mathbf{Q} \mathbf{Q}^T,$$

waarbij  $\mathbf{Q}$  de  $n$  bij  $m$  matrix waarvan de kolommen bestaan uit de vectoren  $\mathbf{q}_1$  t.e.m.  $\mathbf{q}_m$ . In dit geval is het berekenen van de inverse matrix triviaal!

Om de coördinaten t.o.v. de vectoren  $\mathbf{q}_1$  t.e.m.  $\mathbf{q}_m$  te vinden van de loodrechte projectie van  $\mathbf{b} \in \mathbb{R}^n$  op de vectorruimte opgespannen door de vectoren  $\mathbf{q}_1$  t.e.m.  $\mathbf{q}_m$  gebruiken we formule (2.4) die vereenvoudigt tot

$$\mathbf{x} = \mathbf{Q}^T \mathbf{b},$$

wat uiteraard een vector in  $\mathbb{R}^m$  is.

#### 4.3.1 Projectie op bestaande basisvectoren

Stel dat  $\mathbf{q}_1$  t.e.m.  $\mathbf{q}_n$  een orthonormale basis is van  $\mathbb{R}^n$ , die niet noodzakelijk gelijk is aan de “standaard” basis. Dit betekent dat elke vector  $\mathbf{b} \in \mathbb{R}^n$  kan geschreven worden als lineaire combinatie van de vectoren  $\mathbf{q}_1$  t.e.m.  $\mathbf{q}_n$ :

$$\mathbf{b} = b_1 \mathbf{q}_1 + b_2 \mathbf{q}_2 + \cdots + b_m \mathbf{q}_m + \cdots + b_n \mathbf{q}_n.$$

Als we nu  $\mathbf{b}$  loodrecht projecteren op de deelruimte opgespannen door de eerste  $m$  basisvectoren, dan wordt de projectie intuïtief gegeven door

$$\text{proj } \mathbf{b} = b_1 \mathbf{q}_1 + b_2 \mathbf{q}_2 + \cdots + b_m \mathbf{q}_m.$$

Hierbij hebben we m.a.w. de eerste  $m$  coördinaten behouden en we hebben eenvoudigweg de coëfficiënten van de vectoren  $\mathbf{q}_{m+1}$  t.e.m.  $\mathbf{q}_n$  gelijkgesteld aan nul.

We kunnen dit ook formeel narekenen. De coördinaten van de projectie worden gegeven door

$$\mathbf{x} = \mathbf{Q}^T \mathbf{b} = \begin{bmatrix} \text{---} & \mathbf{q}_1^T & \text{---} \\ \text{---} & \mathbf{q}_2^T & \text{---} \\ & \vdots & \\ \text{---} & \mathbf{q}_m^T & \text{---} \end{bmatrix} (b_1 \mathbf{q}_1 + b_2 \mathbf{q}_2 + \cdots + b_n \mathbf{q}_n)$$

Omdat we echter te maken hebben met een orthonormale basis geldt inderdaad dat

$$\mathbf{x} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}.$$

### 4.3.2 Oefeningen

1. We wensen te projecteren op de deelruimte opgespannen door de twee orthonormale vectoren:

$$\mathbf{q}_1 = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \text{en} \quad \mathbf{q}_2 = \frac{1}{\sqrt{6}} \begin{bmatrix} -1 \\ 2 \\ -1 \end{bmatrix}.$$

Beantwoord de volgende vragen:

- a) Vind de projectiematrix  $\mathbf{P}$ .
- b) Vind de projectie van een willekeurige vector  $\mathbf{b}$ .

## 4.4 Creëren van een orthonormale basis

Wanneer een willekeurige basis  $\mathbf{a}_1$  t.e.m.  $\mathbf{a}_m$  voor een (deel)vectorruimte gegeven is, dan is het een logische vraag om op basis hiervan een orthonormale basis  $\mathbf{q}_1$  t.e.m.  $\mathbf{q}_m$  te construeren.

De methode van GRAM-SCHMIDT is zo'n methode. In het bijzonder zal gelden dat

$$\text{span}\{\mathbf{a}_1, \dots, \mathbf{a}_i\} = \text{span}\{\mathbf{q}_1, \dots, \mathbf{q}_i\} \quad \text{voor } i \in \{1, 2, \dots, m\}.$$

Het moeilijkere deel van de procedure is uiteraard om een stel *orthogonale* vectoren te vinden dat de juiste eigenschappen heeft. Daarna kan men gemakkelijk elke vector individueel normaliseren om te komen tot een ortho-normale basis.

We noemen de ongenormaliseerde maar orthogonale vectoren  $\mathbf{e}_1$  t.e.m.  $\mathbf{e}_m$ . We geven eerst een concreet voorbeeld met drie vectoren om de procedure te illustreren.

**Voorbeeld 4.10** Gegeven de volgende drie vectoren in  $\mathbb{R}^4$ :

$$\mathbf{a}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{a}_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} \quad \text{en} \quad \mathbf{a}_3 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix},$$

vind drie orthogonale vectoren  $\mathbf{e}_1$ ,  $\mathbf{e}_2$  en  $\mathbf{e}_3$  zodat

$$\begin{aligned} \text{span}\{\mathbf{a}_1\} &= \text{span}\{\mathbf{e}_1\}, \\ \text{span}\{\mathbf{a}_1, \mathbf{a}_2\} &= \text{span}\{\mathbf{e}_1, \mathbf{e}_2\} \quad \text{en} \\ \text{span}\{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3\} &= \text{span}\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}. \end{aligned}$$

Het lijkt logisch om voor de eerste vector  $\mathbf{e}_1$  de vector  $\mathbf{a}_1$  te kiezen:

$$\mathbf{e}_1 = \mathbf{a}_1.$$

De tweede vector  $\mathbf{e}_2$  moet orthogonaal zijn met  $\mathbf{e}_1$  en moet een lineaire combinatie zijn van  $\mathbf{e}_1$  en  $\mathbf{a}_2$  en een niet-nul component hebben in de richting van  $\mathbf{a}_2$ . Als we deze niet-nul component gelijk kiezen aan 1, dan is

$$\mathbf{e}_2 = \mathbf{a}_2 + \alpha \mathbf{e}_1.$$

Als we nu uitdrukken dat  $\mathbf{e}_1 \cdot \mathbf{e}_2$  gelijk is aan nul, dan vinden we dat

$$\mathbf{e}_1 \cdot (\mathbf{a}_2 + \alpha \mathbf{e}_1) = 0 \iff \alpha = -\frac{\mathbf{e}_1 \cdot \mathbf{a}_2}{\mathbf{e}_1 \cdot \mathbf{e}_1}.$$

Dit is logisch! We herkennen hierin immers de projectie van  $\mathbf{a}_2$  op  $\mathbf{e}_1$  en dat stuk nemen we precies weg van  $\mathbf{a}_2$ . We doen de berekening om  $\mathbf{e}_2$  te bepalen:

$$\mathbf{e}_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} - \frac{1}{1} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}.$$

Voor  $\mathbf{e}_3$  doen we hetzelfde: we starten met  $\mathbf{a}_3$  en verwijderen zijn projectie in de richting van  $\mathbf{e}_1$  en  $\mathbf{e}_2$ :

$$\begin{aligned}\mathbf{e}_3 &= \mathbf{a}_3 - \frac{\mathbf{e}_1 \cdot \mathbf{a}_3}{\mathbf{e}_1 \cdot \mathbf{e}_1} \mathbf{e}_1 - \frac{\mathbf{e}_2 \cdot \mathbf{a}_3}{\mathbf{e}_2 \cdot \mathbf{e}_2} \mathbf{e}_2 \\ &= \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} - \frac{1}{1} \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} - \frac{1}{2} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 \\ -\frac{1}{2} \\ \frac{1}{2} \\ 0 \end{bmatrix}.\end{aligned}$$

Als we tenslotte alledrie deze vectoren normaliseren dan vinden we:

$$\mathbf{q}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{q}_2 = \begin{bmatrix} 0 \\ 1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{bmatrix} \quad \text{en} \quad \mathbf{q}_3 = \begin{bmatrix} 0 \\ -1/\sqrt{2} \\ 1/\sqrt{2} \\ 0 \end{bmatrix}.$$

Het algemene proces voor het vinden van een orthonormale basis volgens de procedure van Gram-Schmidt gaat m.a.w. als volgt: in de eerste stap construeren we orthogonale vectoren  $\mathbf{e}_1$  t.e.m.  $\mathbf{e}_m$  op de volgende manier:

- Start met  $\mathbf{e}_1 = \mathbf{a}_1$
- Vervolgens construeren we  $\mathbf{e}_2$  t.e.m.  $\mathbf{e}_m$  in deze volgorde a.d.h.v. de volgende formule:

$$\mathbf{e}_i = \mathbf{a}_i - \frac{\mathbf{e}_1 \cdot \mathbf{a}_i}{\mathbf{e}_1 \cdot \mathbf{e}_1} \mathbf{e}_1 - \frac{\mathbf{e}_2 \cdot \mathbf{a}_i}{\mathbf{e}_2 \cdot \mathbf{e}_2} \mathbf{e}_2 - \dots - \frac{\mathbf{e}_{i-1} \cdot \mathbf{a}_i}{\mathbf{e}_{i-1} \cdot \mathbf{e}_{i-1}} \mathbf{e}_{i-1}. \quad (4.1)$$

In de tweede stap wordt elke  $\mathbf{e}_i$  eenvoudigweg genormaliseerd:

$$\mathbf{q}_i = \frac{1}{\|\mathbf{e}_i\|} \mathbf{e}_i \quad \text{voor } i \in \{1, 2, \dots, m\}.$$

## 4.5 De QR-decompositie

De Gram-Schmidt procedure kan eveneens gebruikt worden om een interessante matrixdecompositie te vinden: de QR-decompositie.

Omdat de vectoren  $\mathbf{q}_i$  voor  $i \in \{1, 2, \dots, m\}$  een orthonormale basis vormen voor de ruimte opgespannen door de vectoren  $\mathbf{a}_j$  kan elke vector  $\mathbf{a}_j$  met  $j \in \{1, 2, \dots, m\}$  geschreven worden als een lineaire combinatie van de vectoren  $\mathbf{q}_i$ , en de coördinaten zijn, zoals hierboven geargumenteed, niets anders dan de inwendige producten van  $\mathbf{a}_j$  en de vectoren  $\mathbf{q}_i$ . Bovendien, door de manier waarop de vectoren  $\mathbf{e}_i$  werden geconstrueerd weten we dat elke vector  $\mathbf{a}_j$  een lineaire combinatie is van vectoren  $\mathbf{q}_i$  waarvoor  $i \leq j$ . In het bijzonder geldt:

$$\mathbf{a}_j = (\mathbf{q}_1 \cdot \mathbf{a}_j)\mathbf{q}_1 + (\mathbf{q}_2 \cdot \mathbf{a}_j)\mathbf{q}_2 + \dots + (\mathbf{q}_j \cdot \mathbf{a}_j)\mathbf{q}_j.$$

We kunnen dit ook uitdrukken in matrixvorm: het rechterlid van deze formule is een lineaire combinatie van vectoren  $\mathbf{q}_i$ . Als we een matrix construeren met de  $\mathbf{q}_i$  als kolommen dan is dit niets anders dan een lineaire combinatie van die kolommen, m.a.w. het is een matrix-vectorproduct! Als we dit doen voor alle vectoren  $\mathbf{a}_j$  dan vinden we de volgende decompositie:

$$\begin{bmatrix} | & | & & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_m \\ | & | & & | \end{bmatrix} = \begin{bmatrix} | & | & & | \\ \mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_m \\ | & | & & | \end{bmatrix} \begin{bmatrix} \mathbf{q}_1 \cdot \mathbf{a}_1 & \mathbf{q}_1 \cdot \mathbf{a}_2 & \mathbf{q}_1 \cdot \mathbf{a}_3 & \cdots & \mathbf{q}_1 \cdot \mathbf{a}_m \\ 0 & \mathbf{q}_2 \cdot \mathbf{a}_2 & \mathbf{q}_2 \cdot \mathbf{a}_3 & \cdots & \mathbf{q}_2 \cdot \mathbf{a}_m \\ 0 & 0 & \mathbf{q}_3 \cdot \mathbf{a}_3 & \cdots & \mathbf{q}_3 \cdot \mathbf{a}_m \\ 0 & 0 & 0 & & \\ \vdots & & & & \\ 0 & 0 & 0 & \cdots & \mathbf{q}_m \cdot \mathbf{a}_m \end{bmatrix}.$$

De laatste matrix is steeds een bovendriehoeksmatrix en wordt typisch met  $\mathbf{R}$  genoteerd<sup>2</sup>.

We hebben m.a.w. de volgende eigenschap aangetoond.

**Eigenschap 4.11** Als  $\mathbf{A}$  een matrix is met lineair onafhankelijke kolommen, dan kan  $\mathbf{A}$  steeds geschreven worden als het product van een matrix  $\mathbf{Q}$  met

<sup>2</sup>Het had ook een  $\mathbf{U}$  kunnen zijn maar dat is niet wat er typisch wordt gebruikt.

orthonormale kolommen en een bovendriehoeksmatrix  $\mathbf{R}$ :

$$\mathbf{A}_{n \times m} = \mathbf{Q}_{n \times m} \mathbf{R}_{m \times m}.$$

■

### 4.5.1 Oefeningen

1. Vind de QR-decompositie van de volgende matrix  $\mathbf{A}$

$$\mathbf{A} = \begin{bmatrix} 2 & 1 \\ 2 & 1 \\ 1 & 5 \end{bmatrix}.$$

2. `numpy` Schrijf een `numpy`-methode om de QR-decompositie te bepalen van de kolommen in een matrix  $\mathbf{A}$ . Pas hiertoe de methode van Gram-Schmidt toe. De returnwaarde is een tuple bestaande uit (in die volgorde)  $\mathbf{Q}$  en  $\mathbf{R}$ .

## 4.6 Lineaire stelsels en de QR-decompositie

Als de coëfficiëntenmatrix  $\mathbf{A}$  van het stelsel

$$\mathbf{Ax} = \mathbf{b}$$

onafhankelijke kolommen heeft, dan kunnen we  $\mathbf{A}$  schrijven als  $\mathbf{QR}$ , zodat het stelsel geschreven wordt als

$$\mathbf{QRx} = \mathbf{b}.$$

Als we nu beide leden vermenigvuldigen met  $\mathbf{Q}^T$  en gebruikmaken van het feit dat

$$\mathbf{Q}^T \mathbf{Q} = \mathbf{I},$$

dan vinden we

$$\mathbf{Rx} = \mathbf{Q}^T \mathbf{b}.$$

Omdat  $\mathbf{R}$  een bovendriehoeksmatrix is kunnen we dit stelsel eenvoudig oplossen m.b.v. achterwaartse substitutie.

**Opmerking 4.12** Wanneer het stelsel strijdig is, dan vinden we opnieuw dezelfde benaderende oplossing als voorheen. De oplossing is dezelfde als die bekomen door formule (2.4), nl.

$$\mathbf{x} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}.$$

te bepalen. Dit is gemakkelijk na te rekenen door de QR-decompositie voor de matrix  $\mathbf{A}$  te gebruiken en er rekening mee te houden dat  $\mathbf{R}$  inverteerbaar is als de rang van  $\mathbf{A}$  gelijk is aan  $m$ . ■

**Voorbeeld 4.13** Veronderstel dat we het (strijdige) stelsel uit Voorbeeld 3.8 wensen “op te lossen” m.b.v. de QR-decompositie dan moeten we in eerste instantie de QR-decompositie van  $\mathbf{A}$  bepalen. We doen dit aan de hand van de Gram-Schmidt procedure. We stellen

$$\mathbf{a}_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}, \quad \text{en} \quad \mathbf{a}_2 = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}.$$

Volgens de Gram-Schmidt procedure construeren we eerst orthogonale vectoren  $\mathbf{e}_1$  en  $\mathbf{e}_2$  die we dan vervolgens normaliseren. Er geldt:

$$\mathbf{e}_1 = \mathbf{a}_1$$

en

$$\begin{aligned} \mathbf{e}_2 &= \mathbf{a}_2 - \frac{\mathbf{e}_1 \cdot \mathbf{a}_2}{\mathbf{e}_1 \cdot \mathbf{e}_1} \mathbf{e}_1 \\ &= \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} - \frac{6}{3} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix}. \end{aligned}$$

Normaliseren levert:

$$\mathbf{q}_1 = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \text{en} \quad \mathbf{q}_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix},$$

en bijgevolg:

$$\mathbf{Q} = \begin{bmatrix} 1/\sqrt{3} & -1/\sqrt{2} \\ 1/\sqrt{3} & 0 \\ 1/\sqrt{3} & 1/\sqrt{2} \end{bmatrix}.$$



De matrix  $\mathbf{R}$  wordt dan, na het berekenen van de inwendige producten<sup>3</sup>,

$$\mathbf{R} = \begin{bmatrix} \sqrt{3} & 2\sqrt{3} \\ 0 & \sqrt{2} \end{bmatrix}.$$

Het stelsel dat nog moet opgelost worden is

$$\mathbf{R}\mathbf{x} = \mathbf{Q}^T\mathbf{b}.$$

Er geldt

$$\mathbf{Q}^T\mathbf{b} = \begin{bmatrix} 5/\sqrt{3} \\ 1/\sqrt{2} \end{bmatrix}.$$

Hieruit volgt dan dat

$$x_2 = 1/2,$$

en

$$\sqrt{3}x_1 + \frac{2\sqrt{3}}{2} = \frac{5}{\sqrt{3}} \iff x_1 = \frac{2}{3}.$$

Dit is (zoals verwacht) dezelfde oplossing als diegene die bekomen was in Voorbeeld 3.9. ■

---

<sup>3</sup>Of na het berekenen van  $\mathbf{Q}^T\mathbf{A}$ .

# De singuliere waarden ontbinding

## 5.1 De singuliere waarden ontbinding

### 5.1.1 De volledige singuliere waarden ontbinding

De SINGULIERE WAARDEN ONTBINDING is een uiterst belangrijke en nuttige matrixfactorisatie, die toelaat om *elke* matrix  $\mathbf{A}$  te schrijven als een product van drie matrices:

$$\mathbf{A}_{n \times m} = \mathbf{U}_{n \times n} \mathbf{\Sigma}_{n \times m} \mathbf{V}_{m \times m}^T. \quad (5.1)$$

In deze ontbinding zijn  $\mathbf{U}$  en  $\mathbf{V}$  orthogonale matrices, i.e.

$$\mathbf{U} \mathbf{U}^T = \mathbf{I}_{n \times n} = \mathbf{U}^T \mathbf{U} \quad \text{en} \quad \mathbf{V} \mathbf{V}^T = \mathbf{I}_{m \times m} = \mathbf{V}^T \mathbf{V}.$$

Bovendien is  $\mathbf{\Sigma}$  een diagonaalmatrix met op de diagonaal elementen

$$\Sigma_{i,i} = \sigma_i$$

die gesorteerd zijn van groot naar klein:

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_k \geq 0, \quad \text{met } k = \min(n, m).$$

Alle elementen van  $\mathbf{\Sigma}$  die niet op de diagonaal staan zijn gelijk aan nul. De getallen  $\sigma_i$  worden de SINGULIERE WAARDEN van de matrix  $\mathbf{A}$  genoemd. De kolommen van  $\mathbf{U}$  worden de LINKSE SINGULIERE VECTOREN van  $\mathbf{A}$  genoemd, terwijl de kolommen van  $\mathbf{V}$  de RECHTSE SINGULIERE VECTOREN zijn.

**Opmerking 5.1** De Engelstalige term voor de singuliere waarden ontbinding is *singular value decomposition*, vandaar de afkorting SVD die we vanaf nu ook zullen gebruiken. ■

**Opmerking 5.2** De diagonaalmatrix in de SVD wordt typisch genoteerd met de Griekse hoofdletter Sigma, ofte  $\Sigma$ . Het sommatieteken lijkt hier zeer sterk op. Vanuit de context zou steeds duidelijk moeten zijn wat er wordt bedoeld. ■

**Voorbeeld 5.3** De SVD van

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

wordt gegeven door

$$\mathbf{A} = \overbrace{\begin{bmatrix} 2/\sqrt{6} & 0 & -1/\sqrt{3} \\ 1/\sqrt{6} & -1/\sqrt{2} & 1/\sqrt{3} \\ 1/\sqrt{6} & 1/\sqrt{2} & 1/\sqrt{3} \end{bmatrix}}^{\mathbf{U}} \overbrace{\begin{bmatrix} \sqrt{3} & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}}^{\Sigma} \overbrace{\begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}}^{\mathbf{V}^T}$$

Je kan gemakkelijk verifiëren dat de matrices  $\mathbf{U}$  en  $\mathbf{V}$  zoals hierboven aangeduid orthogonaal zijn. Bovendien is  $\Sigma$  een diagonaalmatrix waarvan de waarden op de diagonaal gesorteerd zijn van groot naar klein en bovendien niet-negatief zijn. Dit is m.a.w. de SVD van de matrix  $\mathbf{A}$ . ■

### 5.1.2 De zuinige SVD

Wanneer de matrix  $\mathbf{A} \in \mathbb{R}^{n \times m}$  een “lange smalle” matrix is, i.e.  $n > m$ , dan heeft  $\mathbf{A}$  hoogstens  $m$  singuliere waarden die verschillend zijn van nul. In dit geval zijn enkel de eerste  $m$  kolommen van  $\mathbf{U}$  van belang, de kolommen erna komen immers *niet* voor in het matrixproduct  $\mathbf{U}\Sigma$ . Om dit concreter te maken bekijken we een voorbeeld waarbij  $n = 4$  en  $m = 2$

$$\mathbf{U}\Sigma = \begin{bmatrix} | & | & | & | \\ \mathbf{u}_1 & \mathbf{u}_2 & \mathbf{u}_3 & \mathbf{u}_4 \\ | & | & | & | \end{bmatrix}_{4 \times 4} \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \\ 0 & 0 \\ 0 & 0 \end{bmatrix}_{4 \times 2} = \begin{bmatrix} | & | \\ \sigma_1 \mathbf{u}_1 & \sigma_2 \mathbf{u}_2 \\ | & | \end{bmatrix}_{4 \times 2}.$$

We bekomen dezelfde uitkomst wanneer we enkel de eerste twee kolommen van  $\mathbf{U}$  beschouwen en de eerste twee rijen van  $\Sigma$ . Deze kleinere matrices noteren we respectievelijk als  $\hat{\mathbf{U}}$  en  $\hat{\Sigma}$  en er geldt

$$\hat{\mathbf{U}}\hat{\Sigma} = \begin{bmatrix} | & | \\ \mathbf{u}_1 & \mathbf{u}_2 \\ | & | \end{bmatrix}_{4 \times 2} \begin{bmatrix} \sigma_1 & 0 \\ 0 & \sigma_2 \end{bmatrix}_{2 \times 2} = \begin{bmatrix} | & | \\ \sigma_1 \mathbf{u}_1 & \sigma_2 \mathbf{u}_2 \\ | & | \end{bmatrix}_{4 \times 2}.$$

M.a.w. wanneer  $\mathbf{A} \in \mathbb{R}^{n \times m}$  een “lange smalle” matrix is, i.e. wanneer  $n > m$ , dan kunnen we de SVD als volgt schrijven:

$$\begin{aligned}\mathbf{A}_{n \times m} &= \mathbf{U}_{n \times n} \mathbf{\Sigma}_{n \times m} \mathbf{V}_{m \times m}^T \\ &= \left[ \hat{\mathbf{U}}_{n \times m} \mid \hat{\mathbf{U}}^\perp \right] \begin{bmatrix} \hat{\mathbf{\Sigma}}_{m \times m} \\ \mathcal{O} \end{bmatrix} \mathbf{V}^T \\ &= \hat{\mathbf{U}}_{n \times m} \hat{\mathbf{\Sigma}}_{m \times m} \mathbf{V}_{m \times m}^T.\end{aligned}\tag{5.2}$$

Deze vorm van de SVD noemen we de ZUINIGE SINGULIERE WAARDEN ONTBINDING.

**Voorbeeld 5.4** De zuinige SVD van

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

kan eenvoudig afgelezen worden uit Voorbeeld 5.3 en wordt gegeven door

$$\mathbf{A} = \overbrace{\begin{bmatrix} 2/\sqrt{6} & 0 \\ 1/\sqrt{6} & -1/\sqrt{2} \\ 1/\sqrt{6} & 1/\sqrt{2} \end{bmatrix}}^{\hat{\mathbf{U}}} \overbrace{\begin{bmatrix} \sqrt{3} & 0 \\ 0 & 1 \end{bmatrix}}^{\hat{\mathbf{\Sigma}}} \overbrace{\begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}}^{\mathbf{V}^T}.$$

**Opmerking 5.5** Voor de matrix  $\hat{\mathbf{U}}$  is het belangrijk op te merken dat enkel de *kolommen* orthonormaal zijn. Dit betekent dat er nog steeds geldt dat

$$\hat{\mathbf{U}}^T \hat{\mathbf{U}} = \mathbf{I}_{m \times m}$$

maar dat, zoals we ook reeds vroeger hebben gezien,

$$\hat{\mathbf{U}} \hat{\mathbf{U}}^T \neq \mathbf{I}_{n \times n}.$$

Zowel de volledige SVD als de zuinige SVD kunnen eenvoudig berekend worden m.b.v. de `numpy` module in Python, zoals geïllustreerd in Figuur 5.1.

```
import numpy as np

# Volledige SVD
U, S, VT = np.linalg.svd(A, full_matrices = True)
# Zuinige SVD
U, S, VT = np.linalg.svd(A, full_matrices = False)
```

**Figuur 5.1:** Berekenen van de SVD m.b.v. numpy in Python.

### 5.1.3 Lage rang benadering m.b.v. de SVD

#### Uitwendig product

We starten deze sectie met nog een alternatieve manier om het product van twee matrices te berekenen. We zagen reeds dat je van twee vectoren het inwendig product kan berekenen, wat in essentie een getal is. Het is ook het matrixproduct van een rijvector en een kolomvector:

$$\left[ \text{---} \quad \mathbf{a}^T \quad \text{---} \right] \begin{bmatrix} | \\ \mathbf{b} \\ | \end{bmatrix} \in \mathbb{R}^{1 \times 1}.$$

Dit werkt enkel als  $\mathbf{a}$  en  $\mathbf{b}$  evenveel componenten hebben.

We kunnen willekeurige vectoren ook in de “andere volgorde” vermenigvuldigen, nl. eerst een kolomvector en dan een rijvector. Stel dat  $\mathbf{a} \in \mathbb{R}^n$  en  $\mathbf{b} \in \mathbb{R}^m$ , dan geldt:

$$\underbrace{\begin{bmatrix} | \\ \mathbf{a} \\ | \end{bmatrix}}_{\in \mathbb{R}^{n \times 1}} \underbrace{\left[ \text{---} \quad \mathbf{b}^T \quad \text{---} \right]}_{\in \mathbb{R}^{1 \times m}} \in \mathbb{R}^{n \times m}.$$

Op de plaats  $(i, j)$  van het resultaat staat het product van de  $i$ -de component van  $\mathbf{a}$  met de  $j$ -de component van  $\mathbf{b}$ . Deze manier om vectoren te vermenigvuldigen noemen we het **UITWENDIG PRODUCT**. De matrix die het resultaat is van een uitwendig product heeft steeds rang 1: alle kolommen zijn immers veelvoud van elkaar (en van de kolomvector die werd gebruikt bij het berekenen van het uitwendig product).

**Voorbeeld 5.6 (Uitwendig product)** Stel

$$\mathbf{a} = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad \text{en} \quad \mathbf{b} = \begin{bmatrix} 3 \\ 4 \\ 5 \end{bmatrix}$$

dan is het uitwendig product van  $\mathbf{a}$  en  $\mathbf{b}$  gelijk aan

$$\begin{bmatrix} 1 \\ 2 \end{bmatrix} \begin{bmatrix} 3 & 4 & 5 \end{bmatrix} = \begin{bmatrix} 3 & 4 & 5 \\ 6 & 8 & 10 \end{bmatrix}.$$

Merk op alle kolommen veelvoudig zijn van  $\mathbf{a}$ . Dit toont aan dat de rang van deze matrix gelijk is aan 1. ■

Een algemeen matrixproduct  $\mathbf{AB}$  kan steeds uitgedrukt worden als de som van de uitwendige producten van de kolommen van  $\mathbf{A}$  en de rijen van  $\mathbf{B}$ . Stel

$$\mathbf{AB} = \begin{bmatrix} | & | & \cdots & | \\ \mathbf{a}_1 & \mathbf{a}_2 & \cdots & \mathbf{a}_p \\ | & | & \cdots & | \end{bmatrix} \begin{bmatrix} \text{---} & \mathbf{b}_1^T & \text{---} \\ \text{---} & \mathbf{b}_2^T & \text{---} \\ \text{---} & \vdots & \text{---} \\ \text{---} & \mathbf{b}_p^T & \text{---} \end{bmatrix},$$

dan is

$$\mathbf{AB} = \mathbf{a}_1 \mathbf{b}_1^T + \mathbf{a}_2 \mathbf{b}_2^T + \cdots + \mathbf{a}_p \mathbf{b}_p^T.$$

Inderdaad, het element op de plaats  $(i, j)$  van  $\mathbf{a}_k \mathbf{b}_k^T$  (met  $k \in \{1, 2, \dots, p\}$ ) is gelijk aan het  $i$ -de element van  $\mathbf{a}_k$  en het  $j$ -de element van  $\mathbf{b}_k^T$ , nl.  $A_{i,k}$  en  $B_{k,j}$  respectievelijk. Dit betekent dat

$$(\mathbf{AB})_{i,j} = \sum_{k=1}^p A_{i,k} B_{k,j}.$$

*Dit is precies gelijk aan de formule voor het berekenen van het matrixproduct op de "klassieke" manier!*

**Voorbeeld 5.7 (Matrixproduct als som van uitwendige producten)** Stel

$$\mathbf{A} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad \text{en} \quad \mathbf{B} = \begin{bmatrix} 5 & 6 & 7 \\ 8 & 9 & 0 \end{bmatrix}.$$

Dan kunnen we  $\mathbf{AB}$  berekenen als

$$\begin{aligned}\mathbf{AB} &= \begin{bmatrix} 1 \\ 3 \end{bmatrix} \begin{bmatrix} 5 & 6 & 7 \end{bmatrix} + \begin{bmatrix} 2 \\ 4 \end{bmatrix} \begin{bmatrix} 8 & 9 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 5 & 6 & 7 \\ 15 & 18 & 21 \end{bmatrix} + \begin{bmatrix} 16 & 18 & 0 \\ 32 & 36 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 21 & 24 & 7 \\ 47 & 54 & 21 \end{bmatrix}.\end{aligned}$$

Je verifieert eenvoudig dat dit inderdaad het correcte antwoord is voor het matrixproduct  $\mathbf{AB}$ . ■

### Lage rang benadering

Door een matrixproduct te bekijken als een som van uitwendige producten kunnen we de zuinige SVD ook als volgt neerschrijven:

$$\mathbf{A} = \sum_{j=1}^m \sigma_j \mathbf{u}_j \mathbf{v}_j^T = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \cdots + \sigma_m \mathbf{u}_m \mathbf{v}_m^T. \quad (5.3)$$

Door de som in (5.3) af te breken na  $r$  termen krijgen we een *benadering* voor  $\mathbf{A}$  die de som is van  $r$  matrices die elk van rang 1 zijn. De resulterende matrix is dan typisch van rang  $r$ , en kan nooit een rang hebben hoger dan  $r$ :

$$\mathbf{A} \approx \sum_{j=1}^r \sigma_j \mathbf{u}_j \mathbf{v}_j^T = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \cdots + \sigma_r \mathbf{u}_r \mathbf{v}_r^T. \quad (5.4)$$

Deze benadering is de “beste” benadering van rang  $r$  van de originele matrix  $\mathbf{A}$  in die zin dat er geen andere rang  $r$  matrix bestaat die minder afwijkt van  $\mathbf{A}$  wanneer de afwijking wordt gemeten m.b.v. de Frobenius-norm (zie Opmerking 5.9 voor uitleg i.v.m. Frobenius-norm). We kunnen  $\mathbf{A}$  m.a.w. benaderen door een product van drie matrices

$$\tilde{\mathbf{A}}_{n \times m} = \tilde{\mathbf{U}}_{n \times r} \tilde{\Sigma}_{r \times r} \tilde{\mathbf{V}}_{m \times r}^T. \quad (5.5)$$

waarbij  $\tilde{\mathbf{U}}$  uit de eerste  $r$  kolommen van  $\mathbf{U}$  bestaat,  $\tilde{\Sigma}$  gelijk is aan het  $r \times r$  blok links bovenaan  $\Sigma$  en  $\tilde{\mathbf{V}}$  bestaat uit de eerste  $r$  kolommen van  $\mathbf{V}$ .

**Eigenschap 5.8 (Eckart-Young)** De beste benadering van rang  $r$  van een matrix  $\mathbf{A} \in \mathbb{R}^{n \times m}$  met  $n \geq m$  wordt gegeven door

$$\mathbf{A}_r = \sum_{j=1}^r \sigma_j \mathbf{u}_j \mathbf{v}_j^T = \sigma_1 \mathbf{u}_1 \mathbf{v}_1^T + \sigma_2 \mathbf{u}_2 \mathbf{v}_2^T + \cdots + \sigma_r \mathbf{u}_r \mathbf{v}_r^T,$$

waarbij de vectoren  $\mathbf{u}_j$  en  $\mathbf{v}_j$  met  $j \in \{1, 2, \dots, r\}$  de eerste  $r$  linkse en rechtse singuliere vectoren zijn van  $\mathbf{A}$  en waarbij  $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_r$  de eerste  $r$  singuliere waarden zijn van  $\mathbf{A}$ .

Meer concreet: als  $\mathbf{B} \in \mathbb{R}^{n \times m}$  een andere matrix is van hoogstens rang  $r$  dan geldt:

$$\|\mathbf{A} - \mathbf{A}_r\|_F \leq \|\mathbf{A} - \mathbf{B}\|_F. \quad \blacksquare$$

**Opmerking 5.9** Het symbool  $\|\mathbf{A}\|_F$  staat voor de FROBENIUS-NORM van de matrix  $\mathbf{A}$ . In essentie komt het er op dat neer dat men alle elementen van  $\mathbf{A}$  beschouwt als een lange vector met  $n \times m$  elementen en dat men de “gewone” twee-norm berekent van deze vector.  $\blacksquare$

**Voorbeeld 5.10** We zoeken de beste benadering van rang 1 van de matrix

$$\mathbf{A} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

die we reeds voordien hebben gezien. We kunnen de SVD afbreken na één uitwendig product en we krijgen

$$\begin{aligned} \tilde{\mathbf{A}} &= \overbrace{\begin{bmatrix} 2/\sqrt{6} \\ 1/\sqrt{6} \\ 1/\sqrt{6} \end{bmatrix}}^{\tilde{\mathbf{U}}} \overbrace{\begin{bmatrix} \sqrt{3} \end{bmatrix}}^{\tilde{\Sigma}} \overbrace{\begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}}^{\tilde{\mathbf{V}}^T} \\ &= \begin{bmatrix} 1 & 1 \\ 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix}. \end{aligned}$$

Zoals je kan zien heeft deze matrix  $\tilde{\mathbf{A}}$  rang 1, want de twee kolommen zijn lineair afhankelijk.

We berekenen de Frobenius-norm van het verschil tussen  $\mathbf{A}$  en  $\tilde{\mathbf{A}}$ . We vinden:

$$\mathbf{A} - \tilde{\mathbf{A}} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} - \begin{bmatrix} 1 & 1 \\ 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 1/2 & -1/2 \\ -1/2 & 1/2 \end{bmatrix}$$

Bijgevolg is

$$\|\mathbf{A} - \tilde{\mathbf{A}}\|_F = \sqrt{0^2 + (1/2)^2 + (-1/2)^2 + 0^2 + (-1/2)^2 + (1/2)^2} = 1.$$



Als we een willekeurige andere matrix nemen van rang 1, bv.

$$\mathbf{B} = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 0 \end{bmatrix}$$

en we berekenen

$$\|\mathbf{A} - \mathbf{B}\|_F = \left\| \begin{bmatrix} 0 & 0 \\ 0 & -1 \\ 0 & 1 \end{bmatrix} \right\|_F = \sqrt{2},$$

dan zien we dat de Frobenius-norm van het verschil groter is. ■

## 5.2 Principale Componenten Analyse

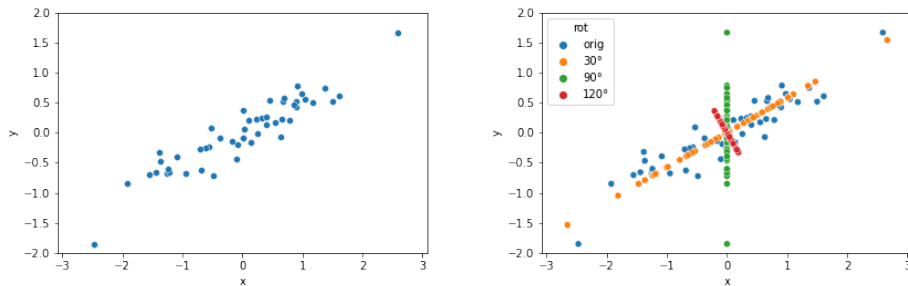
In machinaal leren komt het vaak voor dat datapunten bestaan uit zeer veel verschillende attributen, i.e. we zeggen dat de dataset “hoog-dimensionaal” is, terwijl de data toch vaak dicht liggen bij een deelruimte van een veel lagere dimensie. Zo’n hoge dimensie zorgt voor verschillende problemen: het is zeer moeilijk om de data te visualiseren en verschillende algoritmen voor machinaal leren krijgen het moeilijk wanneer er te veel attributen zijn, zeker als tegelijkertijd het aantal voorbeelden relatief beperkt is.

PRINCIPALE COMPONENTEN ANALYSE, dikwijls afgekort tot PCA, is een techniek die al meer dan 100 jaar wordt gebruikt om de dimensie van data te reduceren en tegelijkertijd zoveel mogelijk informatie in de data behoudt. We bespreken nu de belangrijkste ideeën van PCA.

Het basisidee van principale componenten analyse bestaat erin in om een orthonormale basis  $\mathbf{c}_1$  t.e.m.  $\mathbf{c}_n$  te vinden zodat elk datapunt  $\mathbf{x} \in \mathbb{R}^n$  kan geschreven worden als lineaire combinatie van deze basisvectoren waarbij de coëfficiënten worden gegeven door de inwendige producten van  $\mathbf{c}_i$  en  $\mathbf{x}$ .

Om de dimensie te reduceren projecteren we elk datapunt  $\mathbf{x}$  op de deelruimte opgespannen door de eerste  $r$  basisvectoren  $\mathbf{c}_1$  t.e.m.  $\mathbf{c}_r$ , waarbij de waarde van  $r$  wordt vastgelegd door diegene die PCA uitvoert.

Beschouw de dataset die wordt voorgesteld in Figuur 5.2. Deze dataset bestaat uit 50 tweedimensionale punten, maar je ziet dat er eigenlijk wel een bepaalde “dominante” richting is in deze datapunten. Het doel van PCA



**Figuur 5.2:** Tweedimensionale dataset bestaande uit 50 punten die we willen voorstellen als een ééndimensionale dataset. De tweedimensionale dataset is gecentreerd: het gemiddelde van alle  $x$ -waarden is gelijk aan nul, en dit geldt ook voor het gemiddelde van alle  $y$ -waarden.

is om deze dominante richting te vinden. We kunnen deze datapunten, zoals steeds, voorstellen door een datamatrix  $\mathbf{X} \in \mathbb{R}^{50 \times 2}$ , waarbij elke rij een datapunt voorstelt.

**Opmerking 5.11 (Gecentreerde data)** De data in Figuur 5.2 is zodanig dat het gemiddelde in de  $x$ -richting gelijk is aan nul, en het gemiddelde in de  $y$ -richting is eveneens nul. Dit betekent dat de twee kolommen van  $\mathbf{X}$  een gemiddelde waarde hebben die gelijk is aan nul. *Principale componenten analyse werkt steeds op gecentreerde data.* ■

In Figuur 5.2 zie je verder een drietal richtingen waarop deze 50 datapunten worden geprojecteerd: 30 graden, 90 graden en 120 graden. Het is duidelijk dat de (loodrechte) verschillen tussen de originele blauwe punten en de geprojecteerde oranje punten (op de rechte met hellingsgraad 30 graden) veel kleiner zijn dan de verschillen tussen de originele blauwe punten en hun groene projectie op de  $y$ -as en dat die op hun beurt weer kleiner zijn dan de verschillen tussen de blauwe punten en de rode projectie op de rechte met hellingsgraad 120 graden.

Het is duidelijk dat verschillende richtingen een grotere of kleinere *reconstructiefout* hebben. Bij PCA zoekt men die richting die de kleinste reconstructiefout heeft. De reconstructiefout is in essentie het gemiddelde van de (kwadratische) afstanden tussen de originele punten en hun projecties. De wiskundige definitie vind je in formule (5.6).

Het is relatief ingewikkeld om wiskundig correct vast te stellen en te bewijzen wat de optimale richting is, maar we trachten hier toch wat intuïtie te geven door de richtingen te tonen die gegeven worden door de rechtse singuliere vectoren. Concreet: we berekenen de SVD van  $\mathbf{X}$  en bekomen

$$\mathbf{X}_{50 \times 2} = \hat{\mathbf{U}}_{50 \times 2} \hat{\mathbf{\Sigma}}_{2 \times 2} \mathbf{V}_{2 \times 2}^T$$

We plotten nu de richtingen gegeven door de kolommen van  $\mathbf{V}$ . Het resultaat zie je in Figuur 5.3. Je ziet dat de richting van de eerste rechtse singuliere vector, i.e. van de eerste kolom van  $\mathbf{V}$ , een goed idee geeft van de algemene richting van de data.

**Opmerking 5.12 (Plotten van een richting)** Veronderstel dat de eerste rechtse singuliere vector gegeven wordt door

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix}$$

dan plotten we de rechte met richtingscoëfficiënt  $v_2/v_1$ , i.e. de rechte

$$y = \frac{v_2}{v_1} x.$$

Deze rechte gaat door de oorsprong en bevat het punt  $(v_1, v_2)$ . ■

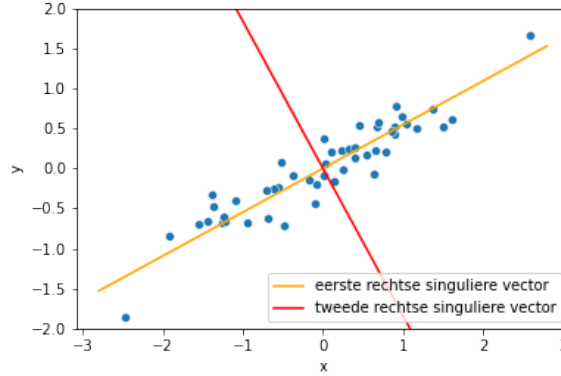
Wat is nu het precieze criterium om te zeggen of een bepaalde richting goed is? Voor elk datapunt  $\mathbf{x}^{(i)} \in \mathbb{R}^2$  bekijken we de loodrechte projectie  $\tilde{\mathbf{x}}^{(i)} \in \mathbb{R}^2$  op een bepaalde richting. Vervolgens kijken we naar de kwadratische afstanden tussen elk datapunt en zijn projectie en nemen hiervan het gemiddelde:

$$J = \frac{1}{m} \sum_{i=1}^m \|\mathbf{x}^{(i)} - \tilde{\mathbf{x}}^{(i)}\|^2. \quad (5.6)$$

*De richting bepaald door de eerste rechtse singuliere vector is diegene waarvoor  $J$  minimaal is. Er is m.a.w. geen enkele andere richting die  $J$  strikt kleiner maakt!*

Deze eigenschap is ook geldig in hogere dimensies.

**Eigenschap 5.13** Voor een gecentreerde datamatrix  $\mathbf{X} \in \mathbb{R}^{m \times n}$  bestaande uit  $m$  datapunten waarbij elk datapunt bestaat uit  $n$  numerieke attributen worden de principale richtingen  $\mathbf{c}_1$  t.e.m.  $\mathbf{c}_n$  gegeven door de rechtse singuliere



**Figuur 5.3:** De twee richtingen gegeven door de twee rechtse singuliere vectoren. Je ziet intuïtief dat de richting gevormd door de eerste singuliere vector optimaal is. De originele punten zullen zeer dicht bij de geprojecteerde punten liggen, of equivalent, de geprojecteerde punten liggen nog steeds ver uit elkaar.

vectoren van  $\mathbf{X}$ . Deze principale richtingen zijn zodanig dat voor elke  $r$  tussen 1 en  $n$  er geen  $r$ -dimensionale deelruimte bestaat die een strikt kleinere reconstructiefout geeft dan de deelruimte opgespannen door  $\mathbf{c}_1$  t.e.m.  $\mathbf{c}_r$ .

M.a.w. als  $\tilde{\mathbf{x}}^{(i)}$  de loodrechte projectie is van  $\mathbf{x}^{(i)}$  op de deelruimte opgespannen door  $\mathbf{c}_1$  t.e.m.  $\mathbf{c}_r$  en  $\mathbf{a}^{(i)}$  de loodrechte projectie is van  $\mathbf{x}^{(i)}$  op een andere deelruimte met dimensie  $r$  dan geldt

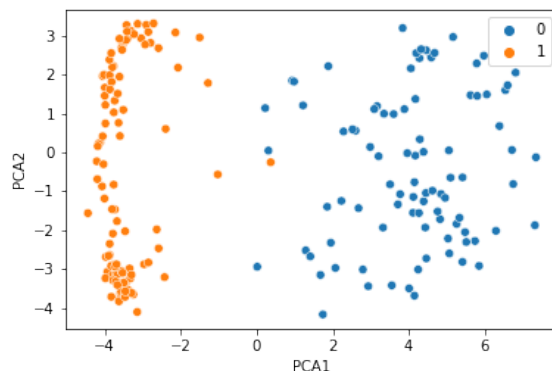
$$\frac{1}{m} \sum_{i=1}^m \|\mathbf{x}^{(i)} - \tilde{\mathbf{x}}^{(i)}\|^2 \leq \frac{1}{m} \sum_{i=1}^m \|\mathbf{x}^{(i)} - \mathbf{a}^{(i)}\|^2.$$

■

### 5.2.1 Voorbeeld: de MNIST dataset

We passen PCA toe op (een deel van) de MNIST dataset. De MNIST dataset bestaat uit een 60 000-tal (trainings)afbeeldingen, waarbij elke afbeelding bestaat uit een vierkant grid van 28 bij 28 pixels, waarbij elke pixel een grijs-waarde voorstelt (tussen 0 en 255).

Voor deze toepassing selecteren we enkel die afbeeldingen die een 0 of een 1 voorstellen, zo zijn er 12 665. We voeren de volgende stappen uit alvorens we PCA toepassen:



**Figuur 5.4:** Eerste twee PCA componenten voor 200 willekeurige afbeeldingen van 0 of 1.

```
import numpy as np

# laad data
X01 = ... # alle afbeeldingen van 0 en 1
        # vorm: (12665, 784) met floating point getallen

# centreer de data: elke kolom krijgt gemiddelde nul
X_centered = X01 - np.mean(X01, axis=0)

# Bereken SVD
U, s, Vt = np.linalg.svd(X_centered, full_matrices = False)
V = Vt.T # Bereken V voor de eenvoud

# Bereken de eerste twee PCA componenten.
# vorm: (12665, 2) met floating point getallen
X_pca2 = X_centered @ V[:, 0:2]

# Plot 200 van deze getallen op een grid
```

**Figuur 5.5:** Belangrijkste delen van de code om Figuur 5.4 te construeren.

- We delen elke pixelwaarde door 255 zodat alle waarden floating-point getallen zijn tussen 0 en 1.
- We beschouwen elk vierkant grid als een (rij)vector met  $28 \times 28 = 784$  componenten door alle pixelwaarden eenvoudigweg achter elkaar te zetten.

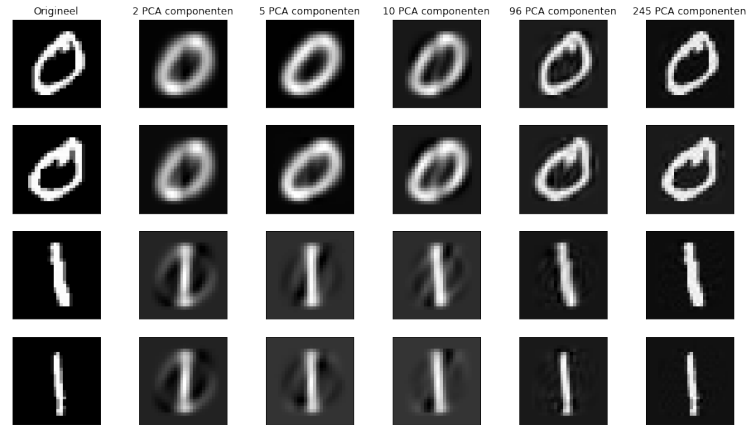
De datamatrix  $\mathbf{X}$  is m.a.w. een matrix bestaande uit 12 665 rijen en 784 kolommen.

In Figuur 5.4 tonen we de waarde van de eerste twee PCA componenten voor 200 willekeurige afbeeldingen. Op deze figuur zie je al duidelijk dat de afbeeldingen die een 1 voorstellen relatief dicht bij elkaar liggen (ze hebben allemaal een gelijkaardige waarde voor de eerste PCA-component), terwijl die afbeeldingen die een 0 voorstellen nog iets meer gespreid liggen maar toch allemaal een duidelijk grotere waarde hebben voor de eerste PCA-component dan de afbeeldingen die een 1 voorstellen. De code om dit te realiseren staat in Figuur 5.5.

In Figuur 5.6 zie je een aantal MNIST-afbeeldingen. De linkerkolom bevat de originele afbeeldingen. De tweede kolom bevat reconstructies wanneer we slechts de twee eerste principale componenten gebruiken. Om dit te verduidelijken, bekijk de afbeelding linksboven, deze heeft bv. coördinaten  $(4.39, -1.26)$ . Daarna berekenen we

$$\begin{aligned} &\text{gecentreerde reconstructie} \\ &= 4.39 \times \text{eerste kolom } \mathbf{V} - 1.26 \times \text{tweede kolom } \mathbf{V} \end{aligned}$$

Dit is een vector met lengte 784. Om de afbeelding te tonen tellen we bij deze vector nu nog de vector op met de gemiddelde pixelwaarde voor elk van de 784 pixels. Deze vector wordt dan geïnterpreteerd als een 28 bij 28 afbeelding en getoond. De andere kolommen worden op een gelijkaardige manier berekend, maar er wordt rekening gehouden met meer principale componenten. Je ziet dat naarmate men meer principale componenten gebruikt men alsmaar dichterbij de originele afbeelding komt. Bij 245 principale componenten ziet men nog slechts een zeer klein verschil met de originele afbeelding.



**Figuur 5.6:** Reconstructies van een aantal MNIST-afbeeldingen, gebruikmakend van een verschillend aantal PCA-componenten.

### 5.2.2 Verklaarde variabiliteit

Bij het MNIST voorbeeld zagen we dat als we meer principale componenten gebruiken voor de reconstructie we dichterbij de originele afbeelding komen. We proberen dit nu wat kwantitatiever uit te drukken.

Wanneer  $\mathbf{X}$  een gecentreerde datamatrix is (i.e. het gemiddelde van elke kolom is gelijk aan nul), dan is  $\mathbf{X}^T \mathbf{X}$  de COVARIANTIEMATRIX. Op de diagonaal staan de inwendige producten van de kolommen met zichzelf:

$$(\mathbf{X}^T \mathbf{X})_{i,i} = \mathbf{x}_i \cdot \mathbf{x}_i = \mathbf{x}_i^T \mathbf{x}_i.$$

M.a.w. op de  $i$ -de diagonaalplaats van  $\mathbf{X}^T \mathbf{X}$  staat de som van de kwadraten van de getallen in de  $i$ -de kolom van  $\mathbf{X}$ . Als deze getallen ver van nul afliggen dan zal dit een grote som opleveren, als de meeste getallen dicht bij nul liggen dan zal deze som klein zijn. Dit is m.a.w. een maat voor de variabiliteit van de data in kolom  $i$ .<sup>1</sup>

**Voorbeeld 5.14** We berekenen de covariantiematrix voor de data in Figuur 5.2.

<sup>1</sup>In het opleidingsonderdeel Data Science & AI zal je zien dat dit gerelateerd is aan de variantie.

In dit geval is  $\mathbf{X} \in \mathbb{R}^{50 \times 2}$  en

$$\mathbf{X}^T \mathbf{X} = \begin{bmatrix} 50.604 & 26.615 \\ 26.615 & 16.291 \end{bmatrix}.$$

Je ziet dat de variabiliteit in de  $x$ -richting (nl. 50.604) groter is dan die in de  $y$ -richting (nl. 16.291), wat overeenkomt met het feit dat in de  $x$ -richting de meeste punten tussen  $-2$  en  $2$  liggen terwijl dat in de  $y$ -richting tussen de  $-1$  en  $1$  is. ■

De covariantiematrix kunnen we ook berekenen aan de hand van SVD:

$$\begin{aligned} \mathbf{X}^T \mathbf{X} &= (\hat{\mathbf{U}} \hat{\mathbf{\Sigma}} \mathbf{V}^T)^T \hat{\mathbf{U}} \hat{\mathbf{\Sigma}} \mathbf{V}^T && \text{definitie SVD} \\ &= (\mathbf{V} \hat{\mathbf{\Sigma}}^T \hat{\mathbf{U}}^T) \hat{\mathbf{U}} \hat{\mathbf{\Sigma}} \mathbf{V}^T && \text{eigenschap transponeren product} \\ &= \mathbf{V} \hat{\mathbf{\Sigma}}^T (\hat{\mathbf{U}}^T \hat{\mathbf{U}}) \hat{\mathbf{\Sigma}} \mathbf{V}^T && \text{associativiteit matrixproduct} \\ &= \mathbf{V} \hat{\mathbf{\Sigma}}^T (\mathbf{I}) \hat{\mathbf{\Sigma}} \mathbf{V}^T && \hat{\mathbf{U}} \text{ bevat orthonormale kolommen} \\ &= \mathbf{V} \hat{\mathbf{\Sigma}}^2 \mathbf{V}^T && \hat{\mathbf{\Sigma}} \text{ is symmetrisch en eenheidselement} \end{aligned}$$

Dit toont aan dat  $\mathbf{X}^T \mathbf{X}$  op een eenvoudige manier gerelateerd is aan  $\hat{\mathbf{\Sigma}}^2$ . Als men een diagonaalmatrix kwadrateert (i.e. met zichzelf vermenigvuldigt) dan moet men eenvoudigweg alle diagonaalelementen kwadrateren.

Men kan aantonen dat de som van de elementen op de diagonaal van  $\mathbf{X}^T \mathbf{X}$  gelijk is aan de som van de elementen op de diagonaal van  $\hat{\mathbf{\Sigma}}^2$ . Dit wordt verder uitgewerkt in een oefening.

**Voorbeeld 5.15** De matrix met de singuliere waarden  $\hat{\mathbf{\Sigma}}$  van de datamatrix voor de data in Figuur 5.2 wordt gegeven door

$$\hat{\mathbf{\Sigma}} = \begin{bmatrix} 8.069 & 0 \\ 0 & 1.335 \end{bmatrix} \quad \text{en} \quad \hat{\mathbf{\Sigma}}^2 = \begin{bmatrix} 65.113 & 0 \\ 0 & 1.781 \end{bmatrix}.$$

Als we de som van de kwadraten van de singuliere waarden berekenen, dan is dit inderdaad precies gelijk<sup>2</sup> aan de som van de elementen op de diagonaal van  $\mathbf{X}^T \mathbf{X}$ :

$$\sigma_1^2 + \sigma_2^2 = 65.113 + 1.781 = 66.894$$

en

$$\mathbf{x}_1 \cdot \mathbf{x}_1 + \mathbf{x}_2 \cdot \mathbf{x}_2 = 50.604 + 16.291 = 66.895. \quad \blacksquare$$

<sup>2</sup>In dit geval is het verschil  $1/1000$  maar dit komt omdat we de getallen steeds hebben afgerond op één duizendste.



Wat er nu interessant is, is dat de (kwadraten van) de singuliere waarden kunnen gebruikt worden om te bepalen welke fractie van de totale variabiliteit men capteert door het gebruik van de eerste  $r$  principale componenten. Dit is namelijk precies gelijk aan:

$$\frac{\sigma_1^2 + \sigma_2^2 + \dots + \sigma_r^2}{\sigma_1^2 + \sigma_2^2 + \dots + \sigma_r^2 + \sigma_{r+1}^2 + \dots + \sigma_n^2}. \quad (5.7)$$

Stel bij wijze van voorbeeld dat we voor een grotere dataset (zoals de MNIST dataset) de eerste twee principale componenten wensen te gebruiken. Dan vinden we de nieuwe datamatrix (in twee dimensies) als volgt:

$$\underbrace{\mathbf{X}_{2D}}_{m \times 2} = \underbrace{\mathbf{X}}_{m \times n} \underbrace{\begin{bmatrix} | & | \\ \mathbf{v}_1 & \mathbf{v}_2 \\ | & | \end{bmatrix}}_{n \times 2}.$$

We zullen aantonen dat de covariantiematrix van  $\mathbf{X}_{2D}$  een diagonaalmatrix is met  $\sigma_1^2$  en  $\sigma_2^2$  op de diagonaal.

Bereken hiertoe de covariantiematrix van  $\mathbf{X}_{2D}$ , en maak gebruik van

$$\mathbf{X}^T \mathbf{X} = \mathbf{V} \hat{\Sigma}^2 \mathbf{V}^T.$$

We vinden

$$\begin{aligned} \mathbf{X}_{2D}^T \mathbf{X}_{2D} &= \begin{bmatrix} \text{---} & \mathbf{v}_1^T & \text{---} \\ \text{---} & \mathbf{v}_2^T & \text{---} \end{bmatrix} \mathbf{X}^T \mathbf{X} \begin{bmatrix} | & | \\ \mathbf{v}_1 & \mathbf{v}_2 \\ | & | \end{bmatrix} \\ &= \begin{bmatrix} \text{---} & \mathbf{v}_1^T & \text{---} \\ \text{---} & \mathbf{v}_2^T & \text{---} \end{bmatrix} \mathbf{V} \hat{\Sigma}^2 \mathbf{V}^T \begin{bmatrix} | & | \\ \mathbf{v}_1 & \mathbf{v}_2 \\ | & | \end{bmatrix}. \end{aligned}$$

Bekijk nu het laatste matrixproduct in deze expressie.

$$\mathbf{V}^T \begin{bmatrix} | & | \\ \mathbf{v}_1 & \mathbf{v}_2 \\ | & | \end{bmatrix} = \begin{bmatrix} \text{---} & \mathbf{v}_1^T & \text{---} \\ \text{---} & \mathbf{v}_2^T & \text{---} \\ & \vdots & \\ \text{---} & \mathbf{v}_n^T & \text{---} \end{bmatrix} \begin{bmatrix} | & | \\ \mathbf{v}_1 & \mathbf{v}_2 \\ | & | \end{bmatrix}$$

Door het feit dat de vectoren van  $\mathbf{V}$  orthonormaal zijn vinden we

$$\mathbf{V}^T \begin{bmatrix} | & | \\ \mathbf{v}_1 & \mathbf{v}_2 \\ | & | \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix}.$$

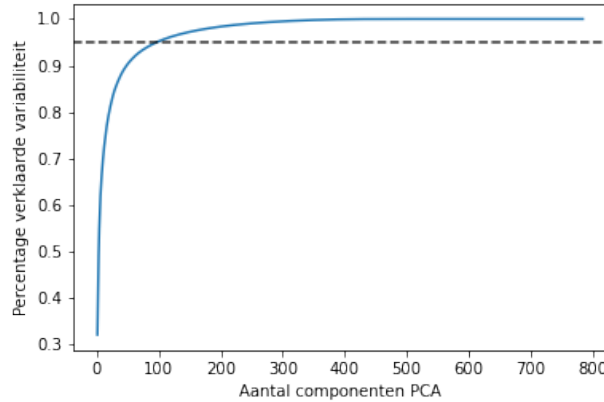
De eerste factoren zijn eenvoudig het getransponeerde hiervan en dus volgt:

$$\begin{aligned} \mathbf{X}_{2D}^T \mathbf{X}_{2D} &= \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \end{bmatrix} \hat{\Sigma}^2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \end{bmatrix} \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \\ 0 & 0 \\ \vdots & \vdots \\ 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}. \end{aligned}$$

Dit is precies wat we wilden aantonen.

Voor visualisatiedoeleinden kiest men het aantal principale componenten typisch gelijk aan twee (of drie). Wanneer PCA een eerste stap is in een groter proces dan kiest men het aantal principale componenten vaak op een zodanige manier dat men een groot percentage (typisch 90, 95 of 99 procent) van de variabiliteit in de data behoudt. Dit doet men door de  $r$  in formule (5.7) zodanig te kiezen dat deze verhouding net groter is dan het vereiste percentage.

**Voorbeeld 5.16 (Verklaarde variantie MNIST)** Voor de MNIST dataset (enkel bestaande uit de afbeeldingen van een 0 of een 1) kunnen we een grafiek opstellen waarbij op de horizontale as het aantal principale componenten staat en op de verticale as het percentage aan verklaarde variabiliteit. Deze grafiek zie je in Figuur 5.7. ■



**Figuur 5.7:** Percentage verklaarde variabiliteit in functie van het aantal gebruikte componenten in PCA. Deze grafiek toont een typisch verloop voor een succesvolle toepassing van PCA. Het verklaarde percentage groeit zeer snel voor de eerste principale componenten en neemt daarna zeer geleidelijk af. De horizontale stippellijn staat op een hoogte van 0.95.

### 5.3 Moore-Penrose pseudoinverse

We hebben in Sectie 2.4 reeds gezien dat een matrix  $\mathbf{A}$  enkel inverteerbaar is wanneer aan twee voorwaarden is voldaan:

1. Het is een vierkante matrix.
2. De kolommen van de matrix zijn lineair onafhankelijk, of anders gezegd, de matrix heeft de grootst mogelijke rang.

We willen deze twee voorwaarden nu enigszins loslaten en een (linkse) pseudoinverse definiëren die de matrix “zo goed mogelijk” invertteert. Een inverse matrix is zodanig dat

$$\mathbf{A}^{-1}\mathbf{A}\mathbf{x} = \mathbf{x}$$

voor alle vectoren  $\mathbf{x}$ . We willen nu kijken of we een matrix  $\mathbf{A}^+$  kunnen vinden waarvoor geldt:

$$\mathbf{A}^+\mathbf{A}\mathbf{x} = \mathbf{x} \quad \text{voor zoveel mogelijk vectoren } \mathbf{x}. \quad (5.8)$$

Hierbij zal de deelvectorruimte bestaande uit alle vectoren  $\mathbf{x}$  waarvoor  $\mathbf{Ax} = \mathbf{0}$  een belangrijke rol spelen.

We beginnen met een eenvoudig voorbeeld: een 2 bij 2 diagonaalmatrix waarbij het tweede element op de diagonaal gelijk is aan nul:

$$\mathbf{A} = \begin{bmatrix} a & 0 \\ 0 & 0 \end{bmatrix}, \quad \text{met } a \neq 0.$$

Voor zo'n matrix geldt:

$$\mathbf{Ax} = \begin{bmatrix} a & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} ax_1 \\ 0 \end{bmatrix}.$$

De vectoren waarvoor  $\mathbf{Ax} = \mathbf{0}$  zijn diegene waarvoor  $x_1 = 0$  en waarvoor  $x_2$  om het even wat kan zijn:

$$\begin{bmatrix} a & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$

Het gemakkelijk om in te zien dat als een vector  $\mathbf{x}$  een niet-nul component  $x_2$  heeft, dat vergelijking (5.8) nooit kan opgaan voor deze vector. Dus, voor deze matrix  $\mathbf{A}$  beperken we ons tot vectoren van de vorm

$$\begin{bmatrix} x_1 \\ 0 \end{bmatrix}.$$

Voor zo'n vector is nu duidelijk dat

$$\mathbf{A}^+ = \begin{bmatrix} 1/a & 0 \\ 0 & 0 \end{bmatrix}$$

deze vector terugbrengt naar de originele vector.

$$\mathbf{A}^+ \mathbf{A} \begin{bmatrix} x_1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1/a & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} a & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1/a & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} ax_1 \\ 0 \end{bmatrix} = \begin{bmatrix} x_1 \\ 0 \end{bmatrix}.$$

Beschouw als tweede voorbeeld een niet-vierkante matrix

$$\mathbf{A} = \begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{met } a \neq 0 \text{ en } b \neq 0.$$

De vectoren die door  $\mathbf{A}$  op nul worden afgebeeld zijn van de vorm

$$\begin{bmatrix} 0 \\ 0 \\ x_3 \end{bmatrix}.$$

Een vector waarvoor vergelijking (5.8) geldig is moet m.a.w. loodrecht staan op deze richting. Dit zijn vectoren van de vorm

$$\begin{bmatrix} x_1 \\ x_2 \\ 0 \end{bmatrix}.$$

De matrix  $\mathbf{A}^+$  moet je aan de rechterkant kunnen vermenigvuldigen met  $\mathbf{A}$  en het resultaat moet een vector zijn met drie elementen. In dit geval betekent dit dat  $\mathbf{A}^+$  een 3 bij 4 matrix moet zijn. Het is niet moeilijk om te verifiëren dat voor

$$\mathbf{A}^+ = \begin{bmatrix} 1/a & 0 & 0 & 0 \\ 0 & 1/b & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

inderdaad geldt dat

$$\mathbf{A}^+ \mathbf{A} \begin{bmatrix} x_1 \\ x_2 \\ 0 \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ 0 \end{bmatrix}.$$

In het bijzonder geldt dat

$$\mathbf{A}^+ \mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \text{en} \quad \mathbf{A} \mathbf{A}^+ = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

Nu we weten hoe we de pseudoinverse kunnen berekenen voor een diagonaalmatrix is het niet moeilijk om dit uit te breiden naar willekeurige matrices aan de hand van de SVD. De SVD laat immers toe om *elke* matrix te schrijven als het product van een orthogonale matrix, een diagonaalmatrix en een tweede orthogonale matrix. De orthogonale matrices zijn steeds inverteerbaar, dus het volgende is niet zo vergezocht.

**Definitie 5.17** Als

$$\mathbf{A} = \mathbf{U} \mathbf{\Sigma} \mathbf{V}^T,$$

dan definiëren we de MOORE-PENROSE PSEUDOINVERSE als

$$\mathbf{A}^+ = \mathbf{V}\mathbf{\Sigma}^+\mathbf{U}^T,$$

waarbij de pseudoinverse  $\mathbf{\Sigma}^+$  van  $\mathbf{\Sigma}$  wordt gegeven door:

1.  $\mathbf{\Sigma}$  te transponeren.
2. Elke singuliere waarde  $\sigma_i$  *verschillend van nul* te vervangen door zijn omgekeerde  $1/\sigma_i$ . ■

**Voorbeeld 5.18** We bekijken de matrix  $\mathbf{A}$  en bijhorende SVD uit Voorbeeld 5.3.

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 2/\sqrt{6} & 0 & -1/\sqrt{3} \\ 1/\sqrt{6} & -1/\sqrt{2} & 1/\sqrt{3} \\ 1/\sqrt{6} & 1/\sqrt{2} & 1/\sqrt{3} \end{bmatrix} \begin{bmatrix} \sqrt{3} & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix}.$$

We gebruiken de definitie van de pseudoinverse  $\mathbf{A}^+$  en we vinden

$$\begin{aligned} \mathbf{A}^+ &= \begin{bmatrix} 1/\sqrt{2} & -1/\sqrt{2} \\ 1/\sqrt{2} & 1/\sqrt{2} \end{bmatrix} \begin{bmatrix} 1/\sqrt{3} & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 2/\sqrt{6} & 1/\sqrt{6} & 1/\sqrt{6} \\ 0 & -1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{3} & 1/\sqrt{3} & 1/\sqrt{3} \end{bmatrix} \\ &= \begin{bmatrix} 1/\sqrt{6} & -1/\sqrt{2} & 0 \\ 1/\sqrt{6} & 1/\sqrt{2} & 0 \end{bmatrix} \begin{bmatrix} 2/\sqrt{6} & 1/\sqrt{6} & 1/\sqrt{6} \\ 0 & -1/\sqrt{2} & 1/\sqrt{2} \\ -1/\sqrt{3} & 1/\sqrt{3} & 1/\sqrt{3} \end{bmatrix} \\ &= \begin{bmatrix} 1/3 & 2/3 & -1/3 \\ 1/3 & -1/3 & 2/3 \end{bmatrix}. \end{aligned}$$

We berekenen vervolgens

$$\mathbf{A}^+\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{en} \quad \mathbf{A}\mathbf{A}^+ = \begin{bmatrix} 2/3 & 1/3 & 1/3 \\ 1/3 & 2/3 & -1/3 \\ 1/3 & -1/3 & 2/3 \end{bmatrix}.$$

Het is in dit geval duidelijk dat  $\mathbf{A}^+\mathbf{A}$  zeer hard gelijk op een identiteitsmatrix (want het is precies een identiteitsmatrix), maar het is minder duidelijk hoe  $\mathbf{A}\mathbf{A}^+$  zich gedraagt als een identiteitsmatrix. Je kan gemakkelijk verifiëren dat de twee kolommen van  $\mathbf{A}$  onder  $\mathbf{A}\mathbf{A}^+$  op zichzelf worden afgebeeld:

$$\mathbf{A}\mathbf{A}^+ \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \quad \text{en} \quad \mathbf{A}\mathbf{A}^+ \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}.$$

Dit betekent dat elke vector die een lineaire combinatie is van de kolommen onder  $\mathbf{A}\mathbf{A}^+$  op zichzelf wordt afgebeeld. Bekijken we nu een vector die loodrecht staat op beide kolommen van  $\mathbf{A}$ . Dit zijn vectoren van de vorm:

$$\begin{bmatrix} x \\ -x \\ -x \end{bmatrix}$$

en

$$\begin{bmatrix} 2/3 & 1/3 & 1/3 \\ 1/3 & 2/3 & -1/3 \\ 1/3 & -1/3 & 2/3 \end{bmatrix} \begin{bmatrix} x \\ -x \\ -x \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

## 5.4 Stelsels oplossen met de pseudoinverse

Een belangrijk thema in lineaire algebra en in machinaal leren is het oplossen van stelsels lineaire vergelijkingen met  $m$  vergelijkingen en  $n$  onbekenden:

$$\mathbf{A}\mathbf{x} = \mathbf{b},$$

met  $\mathbf{A} \in \mathbb{R}^{m \times n}$ .

We hebben al de volgende gevallen gezien:

1.  $m = n$  en  $\text{rang}(\mathbf{A}) = n$ . In dit geval wordt de unieke oplossing van het stelsel gegeven door

$$\mathbf{x} = \mathbf{A}^{-1}\mathbf{b}.$$

2.  $m > n$  en  $\text{rang}(\mathbf{A}) = n$ . In dit geval is het stelsel ofwel strijdig ofwel heeft het een unieke oplossing. De matrix  $\mathbf{A}^T\mathbf{A}$  is in dit geval inverteerbaar en de kleinste kwadraten oplossing wordt gegeven door

$$\mathbf{x}^* = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}.$$

De vector  $\mathbf{x}^*$  is diegene die  $\|\mathbf{A}\mathbf{x} - \mathbf{b}\|^2$  minimaliseert. Dit hebben we reeds gezien in Sectie 3.5.

Gevallen die we nog niet hebben gezien zijn

3.  $m < n$  en  $\text{rang}(\mathbf{A}) = m$ . In dit geval heeft het stelsel oneindig veel oplossingen voor elk rechterlid  $\mathbf{b}$ . We gaan hier nu niet dieper op in omdat de oplossing wordt omvat door die van het meest algemene (en volgende) geval.
4.  $\text{rang}(\mathbf{A}) < \min(n, m)$ . In dit geval is het stelsel ofwel strijdig, ofwel heeft het oneindig veel oplossingen. In dit geval berekenen we de “oplossing”  $\mathbf{x}^+$  als

$$\mathbf{x}^+ = \mathbf{A}^+ \mathbf{b}.$$

Deze vector  $\mathbf{x}^+$  voldoet aan twee belangrijke eigenschappen (die we niet bewijzen):

- a) Het is een *kleinste kwadraten oplossing* die  $\|\mathbf{Ax} - \mathbf{b}\|^2$  minimaliseert.
- b) Als er meer dan één vector is die  $\|\mathbf{Ax} - \mathbf{b}\|^2$  minimaliseert, dan heeft  $\mathbf{x}^+$  de *kleinste norm* van al deze vectoren.

**Voorbeeld 5.19 (Kleinste norm)** Beschouw de coëfficiëntenmatrix

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 0 \\ 2 & 0 & 4 \end{bmatrix}$$

met rang gelijk aan 2. Beschouw als rechterlid

$$\mathbf{b} = \begin{bmatrix} 3 \\ 1 \\ 6 \end{bmatrix}.$$

In dit geval heeft het stelsel zeker oplossingen, want het rechterlid is gelijk aan de som van de kolommen van  $\mathbf{A}$ . M.b.v. rijoperaties bepalen we de meest algemene oplossing van het stelsel. Na uitvoeren van de rijoperatie  $R_3 \leftarrow R_3 - 2R_1$  wordt het stelsel:

$$\begin{cases} x + 2z = 3 \\ y = 1 \\ 0 = 0. \end{cases}$$

De oplossingenverzameling wordt gegeven door

$$V = \left\{ \begin{bmatrix} 3 - 2z \\ 1 \\ z \end{bmatrix} \mid z \in \mathbb{R} \right\}$$



De 2-norm in het kwadraat van zo'n oplossing wordt gegeven door

$$(3 - 2z)^2 + 1^2 + z^2 = 5z^2 - 12z + 10.$$

De  $z$ -waarde waarvoor deze kwadratische functie minimaal<sup>3</sup> wordt is  $z = 12/10$ , zodat de oplossing met de minimale norm wordt gegeven door

$$\begin{bmatrix} 3/5 \\ 1 \\ 6/5 \end{bmatrix}.$$

We gebruiken nu de pseudoinverse van  $\mathbf{A}$  om een oplossing van het stelsel te bepalen. M.b.v. `numpy` vinden we dat

$$\mathbf{A}^+ = \begin{bmatrix} 4/100 & 0 & 8/100 \\ 0 & 1 & 0 \\ 8/100 & 0 & 16/100 \end{bmatrix}.$$

We berekenen nu  $\mathbf{A}^+\mathbf{b}$  en we vinden

$$\begin{bmatrix} 4/100 & 0 & 8/100 \\ 0 & 1 & 0 \\ 8/100 & 0 & 16/100 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \\ 6 \end{bmatrix} = \begin{bmatrix} 3/5 \\ 1 \\ 6/5 \end{bmatrix}.$$

Dit is precies dezelfde oplossing als diegene die we daarnet berekend hebben. Dit illustreert het feit dat de pseudoinverse de oplossing geeft met de minimale norm wanneer er meerdere oplossingen zijn. ■

**Voorbeeld 5.20 (Kleinste kwadraten oplossing)** We werken nog steeds met de coëfficiëntmatrix uit Voorbeeld 5.19.

Wanneer we een rechterlid nemen dat geen lineaire combinatie is van de kolommen van  $\mathbf{A}$ , dan is er uiteraard geen oplossing, maar krijgen we een kleinste kwadraten oplossing, bv. voor het rechterlid

$$\mathbf{b} = \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix}$$

vinden we

$$\mathbf{x}^+ = \mathbf{A}^+\mathbf{b} = \begin{bmatrix} 4/100 & 0 & 8/100 \\ 0 & 1 & 0 \\ 8/100 & 0 & 16/100 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 28/100 \\ 1 \\ 56/100 \end{bmatrix}, \quad (5.9)$$

---

<sup>3</sup>Een kwadratische functie  $az^2 + bz + c$  heeft haar top op  $z = -\frac{b}{2a}$ .

en

$$\mathbf{Ax}^+ = \begin{bmatrix} 14/10 \\ 1 \\ 28/10 \end{bmatrix}.$$

Om nu te verifiëren dat dit een kleinste kwadraten oplossing is kunnen we zien dat de laatste kolom van  $\mathbf{A}$  overbodig is, en we trachten het stelsel

$$\mathbf{A}'\mathbf{x} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 2 & 0 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix}$$

op te lossen. De matrix  $\mathbf{A}'$  heeft onafhankelijke kolommen, en dus is  $(\mathbf{A}')^T \mathbf{A}'$  inverteerbaar:

$$(\mathbf{A}')^T \mathbf{A}' = \begin{bmatrix} 5 & 0 \\ 0 & 1 \end{bmatrix} \quad \text{en} \quad ((\mathbf{A}')^T \mathbf{A}')^{-1} = \begin{bmatrix} 1/5 & 0 \\ 0 & 1 \end{bmatrix}$$

De oplossing  $\mathbf{x}^*$  wordt dan gegeven door

$$\begin{aligned} \mathbf{x}^* &= ((\mathbf{A}')^T \mathbf{A}')^{-1} \mathbf{A}'^T \mathbf{b} \\ &= \begin{bmatrix} 1/5 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 3 \\ 1 \\ 2 \end{bmatrix} \\ &= \begin{bmatrix} 1/5 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 7 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} 7/5 \\ 1 \end{bmatrix}. \end{aligned}$$

Dit is nu (uiteraard) verschillend van de oplossing in (5.9) maar we zien dat al het gewicht van de derde kolom 56/100 dubbel is overgedragen naar de eerste kolom:

$$7/5 = 28/100 + 2 \times 56/100.$$

Bovendien is:

$$\mathbf{A}'\mathbf{x}^* = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 2 & 0 \end{bmatrix} \begin{bmatrix} 7/5 \\ 1 \end{bmatrix} = \begin{bmatrix} 14/10 \\ 1 \\ 28/10 \end{bmatrix}.$$

In dit geval is het trouwens zo dat er oneindig veel kleinste kwadraten oplossingen zijn: het is immers alsof we het stelsel

$$\begin{bmatrix} 1 & 0 & 2 \\ 0 & 1 & 0 \\ 2 & 0 & 4 \end{bmatrix} \mathbf{x} = \begin{bmatrix} 14/10 \\ 1 \\ 28/10 \end{bmatrix}$$

oplossen. De oplossingen van de vergelijking zijn van de vorm:

$$\begin{bmatrix} 14/10 - 2z \\ 1 \\ z \end{bmatrix}.$$

Als we de waarde van  $z$  zoeken waarvoor de norm van de vector wordt geminimaliseerd (op dezelfde manier als voordien) dan vinden we  $z = 56/100$ . We krijgen m.a.w. de oplossing (5.9) gevonden m.b.v. de Moore-Penrose pseudoinverse! ■

## 5.5 Oefeningen

1. Het SPOOR (Eng. *trace*)  $\text{Tr}(\mathbf{A})$  van een vierkante matrix is de som van de elementen op de diagonaal. Gebruik `numpy` om te verifiëren welke van deze eigenschappen geldig zijn voor alle vierkante matrices.

- $\text{Tr}(\mathbf{AB}) \stackrel{?}{=} \text{Tr}(\mathbf{BA})$
- $\text{Tr}(\mathbf{ABC}) \stackrel{?}{=} \text{Tr}(\mathbf{BCA})$
- $\text{Tr}(\mathbf{ABC}) \stackrel{?}{=} \text{Tr}(\mathbf{BAC})$

Welke eigenschap kan je gebruiken om te bewijzen dat

$$\text{Tr}(\mathbf{X}^T \mathbf{X}) = \text{Tr}(\hat{\Sigma}^2)$$

lettend op het feit dat we hebben aangetoond dat

$$\mathbf{X}^T \mathbf{X} = \mathbf{V} \hat{\Sigma}^2 \mathbf{V}^T?$$

2. Beschouw de volgende dataset bestaande uit 4 punten:

$$\{(2, 0), (0, -2), (3, -3), (-1, 1)\}$$

- a) Converteer deze dataset naar een gecentreerde dataset, d.w.z. een dataset waarvoor het gemiddelde van elke component gelijk is aan nul.
  - b) Maak een schets van deze gecentreerde dataset.
  - c) Gebruik `numpy` om de zuinige SVD van de gecentreerde datamatrix te vinden.
  - d) Maak een schets van de richting van de eerste principale component.
  - e) Projecteer (op je schets) de 4 datapunten op de richting van de eerste principale component. Wat zijn volgens je schets de coördinaten van deze punten t.o.v. de eerste principale component?
  - f) Verifieer je antwoord door een berekening uit te voeren m.b.v. `numpy`.
  - g) Welk percentage van de variabiliteit wordt er verklaard door het gebruik van de eerste principale component?
3. Voor de matrix  $\mathbf{A}$  uit Voorbeeld 5.19.
- a) Bereken de SVD m.b.v. `numpy`.
  - b) Verifieer de pseudoinverse a.d.h.v. de bekomen SVD.
  - c) Verifieer dat  $\mathbf{A}^+ \mathbf{A} \mathbf{x} = \mathbf{x}$  voor elke rij van  $\mathbf{A}$  (waarbij je de rij moet transponeren om een kolomvector te krijgen.)
  - d) Verifieer dat  $\mathbf{A} \mathbf{A}^+ \mathbf{x} = \mathbf{x}$  voor elke kolom van  $\mathbf{A}$ .
4. Beschouw een verzameling vier punten in  $\mathbb{R}^2$ :  $\{(0, 0), (1, 8), (3, 8), (4, 20)\}$ . Gebruik de pseudoinverse om de volgende vragen op te lossen. Visualiseer telkens de oplossing.
- a) Vind de vergelijking van de best passende rechte bij deze vier punten.
  - b) Vind de vergelijking van de best passende horizontale rechte bij deze vier punten.
  - c) Vind de vergelijking van de best passende rechte door de oorsprong bij deze vier punten.
  - d) Vind de vergelijking van de best passende parabool bij deze vier punten.

5. Schrijf je eigen methode `pseudoinverse` om de pseudoinverse van een matrix te bepalen. Maak hierbij gebruik van de ingebouwde methode in `numpy` om de SVD te bepalen.

- Vergelijk jouw uitkomst met de uitkomst bekomen door `np.linalg.pinv`.
- Schrijf een methode die de pseudoinverse test door te verifiëren dat

$$\mathbf{A}^+ \mathbf{A} \mathbf{x} = \mathbf{x}$$

voor alle rijen  $\mathbf{x}$  van  $\mathbf{A}$  en

$$\mathbf{A} \mathbf{A}^+ \mathbf{x} = \mathbf{x}$$

voor alle kolommen  $\mathbf{x}$  van  $\mathbf{A}$ . Verifiëren of twee vectoren aan elkaar “gelijk” zijn kan m.b.v. de methode `np.allclose()`.

# Deel II

# Analyse

# Reële functies in één veranderlijke

Using the chain rule is like peeling an onion: you have to deal with each layer at a time, and if it is too big you will start crying.  
*Anonymous Professor*<sup>1</sup>

## 6.1 Inleiding reële functies

Een REËLE FUNCTIE is een functie die met elk reëel getal in haar DOMEIN een reëel getal associeert. Het BEELD van een reële functie is de verzameling van die reële getallen die de “uitkomst” zijn van die reële functie. Een getal  $b$  behoort m.a.w. tot het beeld van de reële functie  $f$  als en slechts als er een getal  $a$  bestaat in het domein van  $f$  waarvoor geldt dat  $f(a) = b$ . Wanneer je een functie voorstelt door haar grafiek, dan lees je het domein af op de horizontale  $X$ -as als alle mogelijke  $x$ -waarden waarvoor je een  $y$ -waarde kan berekenen. Het beeld van de functie  $f$  lees je af op de verticale  $Y$ -as als alle mogelijke  $y$ -waarden die je kan bereiken voor alle mogelijke  $x$ -waarden die tot het domein behoren.

Eenvoudige reële functies die je reeds kent zijn:

$$f_1: \mathbb{R} \rightarrow \mathbb{R}: x \mapsto f_1(x) = x, \quad \text{en} \quad f_2: \mathbb{R} \rightarrow \mathbb{R}: x \mapsto f_2(x) = \frac{x^2}{4}.$$

<sup>1</sup>Zie video: <https://www.youtube.com/watch?v=YG15m2VwSjA>

De functie  $f_1$  is een voorbeeld van een lineaire functie, terwijl  $f_2$  een voorbeeld is van een kwadratische functie. Beide zijn voorbeelden van VEELTERMFUNCTIES, die je reeds in het eerste jaar hebt ontmoet. Het domein en beeld van  $f_1$  zijn allebei gelijk aan de volledige verzameling van de reële getallen  $\mathbb{R}$ , terwijl het domein van  $f_2$  ook gelijk is aan  $\mathbb{R}$ , maar haar beeld beperkt is tot de verzameling van de niet-negatieve reële getallen  $\mathbb{R}^+$  (omdat het kwadraat van elk getal niet-negatief is). Je ziet de grafiek van deze functies in Figuur 6.1.

Functies kunnen ook een *samengesteld voorschrift* hebben, dit werkt dan net als een `switch` of `case`-statement in klassieke programmeertalen. Voorbeelden van zo'n functies zijn:

$$f_3(x) = \begin{cases} 0 & \text{als } x \leq 0 \\ x & \text{anders} \end{cases} \quad \text{en} \quad f_4(x) = \begin{cases} 0 & \text{als } x \leq 0 \\ 1 & \text{anders.} \end{cases}$$

De eenvoudige functie  $f_3$ , waarvan je de grafiek kunt zien in Figuur 6.2 speelt een zeer belangrijke rol in het gebied van “deep learning” en wordt in die context de RECTIFIED LINEAR UNIT of kortweg RELU genoemd. De functie  $f_4$  wordt de HEAVISIDE STAPFUNCTIE genoemd en is de activatie-functie<sup>2</sup> bij een perceptron.

**Opmerking 6.1** Je ziet dat we voor de functies  $f_3$  en  $f_4$  een verkorte notatie hebben gebruikt. We hadden ook kunnen schrijven:

$$f_3: \mathbb{R} \rightarrow \mathbb{R}: x \mapsto f_3(x) = \begin{cases} 0 & \text{als } x \leq 0 \\ x & \text{anders} \end{cases}$$

maar dat is een heel stuk langer en aangezien we op dit moment “weten” dat we werken met reële functies levert dit eigenlijk niet veel extra informatie op. ■

Reële functies kunnen op verschillende manieren gecombineerd worden om er nieuwe functies mee te creëren. Je kan functies OPTELLEN, bijvoorbeeld

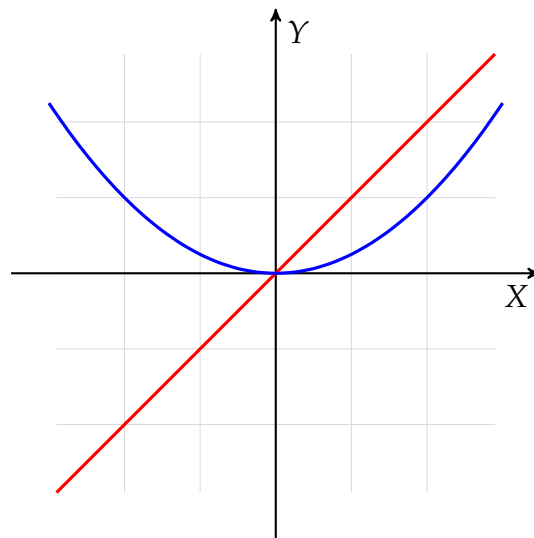
$$f(x) = \sin(x) + x,$$

waarbij  $f$  de som is van de functies  $\sin$  en een lineaire functie. Je kan functies ook VERMENIGVULDIGEN zoals in het voorbeeld hieronder

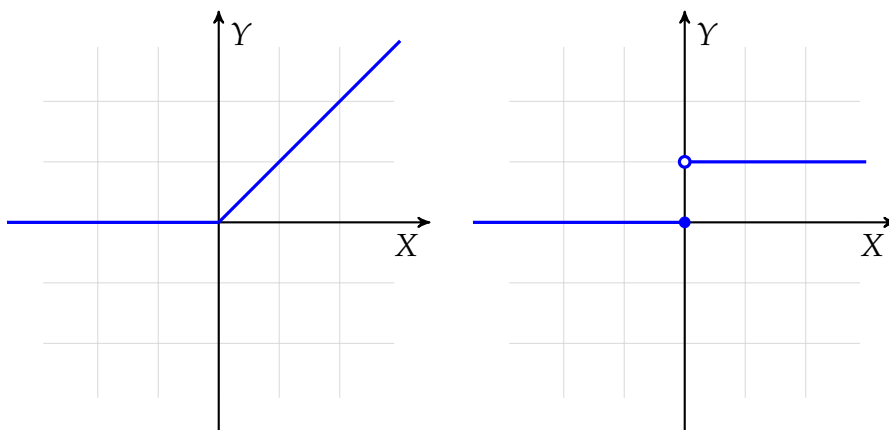
$$f(x) = x \ln(x),$$

<sup>2</sup>In een later opleidingsonderdeel zul je leren wat dit precies is.





**Figuur 6.1:** De grafiek van twee functies: een lineaire functie  $f_1(x) = x$  en een kwadratische functie  $f_2(x) = x^2/4$ .



**Figuur 6.2:** De grafiek van twee functies: de ReLu-functie en de stap-functie.

waarbij deze functie het product is van een lineaire functie en de natuurlijke logaritmische functie, die we later meer in detail zullen bespreken.

In het algemeen geldt dat als  $h$  de som is van functies  $f$  en  $g$ , i.e. als

$$h = f + g,$$

dan betekent dit dat de functiewaarden van  $h$  worden berekend als de som van de functiewaarden van  $f$  en  $g$ . In formulevorm wordt dit:

$$h: \mathbb{R} \rightarrow \mathbb{R}: x \mapsto h(x) = f(x) + g(x).$$

Voor het product van functies werken we op een gelijkaardige manier: als

$$h = f g$$

dan geldt dat

$$h: \mathbb{R} \rightarrow \mathbb{R}: x \mapsto h(x) = f(x)g(x).$$

Een derde manier om nieuwe functies te creëren is door SAMENSTELLING van functies, ook FUNCTIECOMPOSITIE genoemd: hierbij is de uitvoer van een eerste functie de invoer voor een tweede functie. Dit is zeer gelijkaardig als bij programmeren waarbij je de return-waarde van een methode gebruikt als invoerparameter voor een tweede methode. De samenstelling van de functies  $f$  (die als eerste wordt uitgevoerd) en  $g$  (die als tweede wordt uitgevoerd) wordt gegeven door

$$h: \mathbb{R} \rightarrow \mathbb{R}: x \mapsto h(x) = g(f(x)).$$

De functie  $h$  wordt ook genoteerd als  $g \circ f$ , en we lezen dit als “ $g$  na  $f$ ”. Let goed op de volgorde waarin de functies worden genoteerd! Bijvoorbeeld, de functie:

$$h: \mathbb{R} \rightarrow \mathbb{R}: x \mapsto e^{-x^2},$$

is de samenstelling van de functie  $f(x) = -x^2$  en de exponentiële functie  $g(x) = e^x$ .

### 6.1.1 Limieten

In de analyse is het begrip **LIMIET** zeer belangrijk. We geven hier slechts een informele en intuïtieve behandeling en definitie.

We zeggen dat de limiet van de functie  $f$  voor  $x$  naderend naar  $a$  gelijk is aan  $b$  als  $f(x)$  willekeurig dicht bij  $b$  kan komen door  $x$  dicht genoeg, *maar verschillend van*  $a$ , te kiezen. We noteren dit als

$$\lim_{x \rightarrow a} f(x) = b.$$

Het berekenen van limieten kan een heel onderwerp op zichzelf zijn, maar in de praktijk kan men de limiet vaak benaderen door  $x$ -waarden te proberen die almaar dichter bij  $a$  liggen (maar niet gelijk zijn aan  $a$ ) en te kijken of de bijhorende functiewaarden “convergeren”, i.e. of de functiewaarden rond een bepaald getal komen te liggen.

**Voorbeeld 6.2** De volgende limieten liggen voor de hand omdat we eenvoudigweg de functiewaarde kunnen berekenen (en geloven/zien dat dit dan ook de limiet moet zijn omdat het “normale” functies zijn).

- $\lim_{x \rightarrow a} x = a$
- $\lim_{x \rightarrow a} x^2 = a^2$
- $\lim_{x \rightarrow a} \frac{1}{x} = \frac{1}{a}$  als  $a \neq 0$ . ■

**Voorbeeld 6.3** Bij de volgende limiet ligt het al iets moeilijker omdat we in het limietpunt de functiewaarde niet kunnen bepalen omdat het geen deel uitmaakt van het domein van de functie (omdat we niet kunnen delen door 0):

$$\lim_{x \rightarrow -1} \frac{x^2 - 1}{x + 1}.$$

Als je echter de volgende Python-code uitvoert

```
f = lambda x : (x*x - 1)/(x + 1)
[f" {f(-1 - 1/h) :.5f}, {f(-1 + 1/h) :.5f}"
 for h in range(1, 10_000, 1000)]
```

dan krijg je de volgende uitvoer

```
['-3.00000, -1.00000',
 '-2.00100, -1.99900',
 '-2.00050, -1.99950',
```

```
'-2.00033, -1.99967',
'-2.00025, -1.99975',
'-2.00020, -1.99980',
'-2.00017, -1.99983',
'-2.00014, -1.99986',
'-2.00012, -1.99988',
'-2.00011, -1.99989']
```

waaruit je kan afleiden dat de waarde van de gezochte limiet gelijk is aan  $-2$ . Dit is in dit geval dan ook het correcte antwoord. ■

**Voorbeeld 6.4** Een limiet hoeft niet altijd te bestaan. Bekijk bv. de stapfunctie waarbij we de limiet trachten te berekenen in het punt  $a = 0$ . Als we  $a = 0$  naderen van de rechterkant dan is de functiewaarde steeds gelijk aan 1. Als we echter  $a = 0$  naderen van de linkerkant, dan is de functiewaarde gelijk aan 0. Omdat deze twee waarden verschillen van elkaar is de limiet niet gedefinieerd. In dit geval bestaan de zogenaamde linker- en rechterlimiet maar zijn ze verschillend van elkaar, zodat de limiet zelf niet bestaat. ■

**Voorbeeld 6.5** Voor de functie

$$f(x) = \sin\left(\frac{1}{x}\right)$$

bestaat de limiet niet in  $a = 0$ . In dit geval bestaan de linker- en rechterlimiet ook niet. We bewijzen dit niet formeel maar bekijken functiewaarden in de buurt van nul m.b.v. volgende Python-code

```
f = lambda x : np.sin(1/x)
[f"{f(0 - 1/h):.5f}, {f(0 + 1/h):.5f}"
 for h in range(50_000, 100_000, 5000)]
```

met als uitvoer

```
['0.99984, -0.99984',
'0.13698, -0.13698',
'-0.95747, 0.95747',
'-0.43316, 0.43316',
'0.82347, -0.82347',
'0.68789, -0.68789',
'-0.61068, 0.61068',
```

```
'-0.87680, 0.87680',
'0.33946, -0.33946',
'0.98181, -0.98181']
```

De waarden springen min of meer willekeurig heen en weer en lijken in geen van beide kolommen te convergeren naar een bepaalde waarde. ■

### 6.1.2 Continuïteit

Nauw gerelateerd aan het limietbegrip is het begrip continuïteit. We zeggen dat een functie CONTINU IS IN HET PUNT  $a$  (waarbij  $a$  moet behoren tot het domein van de functie) als de limiet van  $f$  voor  $x$  naderend naar  $a$  gelijk is aan de functiewaarde  $f(a)$ .

**Opmerking 6.6** Wanneer  $a$  aan de rand ligt van het domein van  $f$ , dan wordt de gepaste éénzijdige limiet genomen. ■

Als een functie continu is in *alle punten van haar domein* dan zeggen we eenvoudigweg dat de functie continu is. Als dat het geval is dan zal de grafiek van de functie in haar domein geen “sprongen” vertonen.

**Voorbeeld 6.7** De stapfunctie is niet continu in het punt 0, omdat de limiet daar niet bestaat en dus zeker verschillend is van de functiewaarde in 0. Je ziet dat de grafiek van de stapfunctie door ook een “sprong” maakt. ■

**Voorbeeld 6.8** De functie ReLu is continu in haar domein. Je ziet dit ook aan de grafiek die geen sprongen vertoont. Er is weliswaar een “knik” in  $x = 0$  maar dit heeft een andere oorzaak. ■

### 6.1.3 Oefeningen

1. Bekijk de functie die reële getallen afrondt naar beneden. Dit is de *floor*-functie die je reeds kent vanuit de cursus IT Fundamentals. Het functievoorschrift van deze functie is:

$$f: \mathbb{R} \rightarrow \mathbb{R}: x \mapsto f(x) = \lfloor x \rfloor = \text{grootste } z \in \mathbb{Z} \text{ waarvoor } z \leq x.$$

- a) Geef het domein en beeld van deze functie.
- b) Bespreek de continuïteit van deze functie. Is welke punten is ze continu en in welke niet?

2. Bereken de volgende limieten m.b.v. Python. Maak eventueel onderscheid tussen linker- en rechterlimiet.

a)  $\lim_{x \rightarrow 0} \frac{\sin(x)}{x}$

b)  $\lim_{x \rightarrow 0} \frac{\tan(x)}{x}$

c)  $\lim_{x \rightarrow \infty} \frac{3x^2 + x + 1}{x^2 - 2}$

d)  $\lim_{x \rightarrow 1} \frac{3x^2 + x + 1}{x - 1}$

e)  $\lim_{x \rightarrow 1} \frac{x^2 - 3x + 2}{x - 1}$

## 6.2 Afgeleiden

In veel toepassingen in de wiskunde en machinaal leren wenst men het minimum (of maximum) te vinden van een functie (over een bepaald domein), eventueel onder bepaalde restricties. In deze context is het nuttig om te weten of een functie *stijgend* of *dalend* is in een bepaald punt.

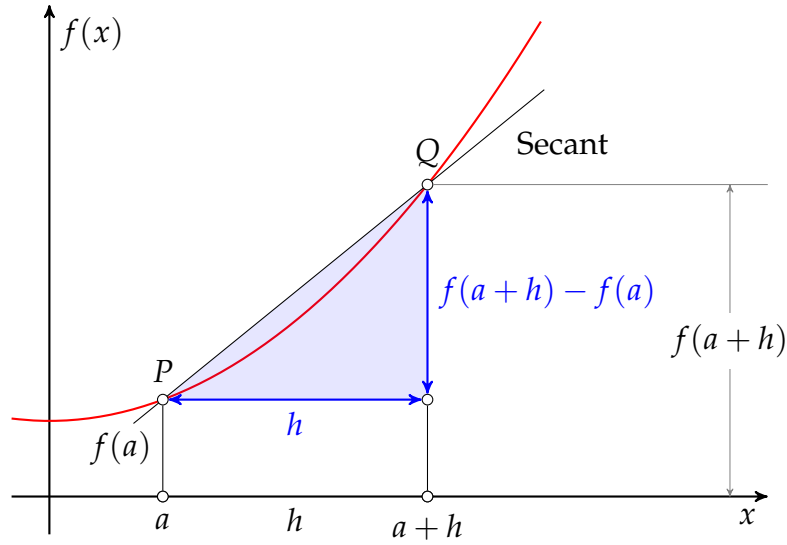
Intuïtief is een functie  $f$  stijgend in het punt  $a$  als de functiewaarden voor  $x$ -waarden net links van  $a$  kleiner zijn dan  $f(a)$  en omgekeerd als de functiewaarden voor  $x$ -waarden net rechts van  $a$  groter zijn dan  $f(a)$ . Dit betekent het zinvol is om te kijken naar het verschil

$$f(a + h) - f(a),$$

waarbij  $h$  klein is in absolute waarde. Bovendien, als we willen weten wat de stijging (of daling) is per eenheid verschil op de  $x$ -as dan delen we dit verschil door  $h$ , nl. door het verschil tussen de punten op de  $x$ -as:

$$\frac{f(a + h) - f(a)}{h}. \quad (6.1)$$

Op die manier krijgen we het quotiënt van de overstaande zijde en de aanliggende zijde in de rechthoekige driehoek gevormd door de punten  $(a, f(a))$ ,  $(a + h, f(a))$  en  $(a + h, f(a + h))$ . Dit is dan meteen ook de *richtingscoëfficiënt* van de rechte door de punten  $(a, f(a))$  en  $(a + h, f(a + h))$ . Zo'n rechte wordt een secant-lijn genoemd. Dit wordt geïllustreerd in Figuur 6.3. Naarmate  $h$  kleiner wordt geeft de verhouding (6.1) beter en beter de informatie



**Figuur 6.3:** Illustratie van het de secant-lijn. De verhouding  $\frac{f(a+h)-f(a)}{h}$  is de richtingscoëfficiënt van de rechte  $PQ$ .

weer m.b.t. het stijgen of dalen van de functie  $f$  in het punt  $a$ . De secantlijnen naderen dan ook naar de raaklijn aan de functie in het punt  $(a, f(a))$ . Dit wordt geïllustreerd in Figuur 6.4. Het is dan ook niet verwonderlijk dat we de limiet nemen waarbij  $h$  naar nul nadert in de formule (6.1) om de afgeleide te definiëren.

**Definitie 6.9** De functie  $f$  is AFLEIDBAAR in  $a$  als

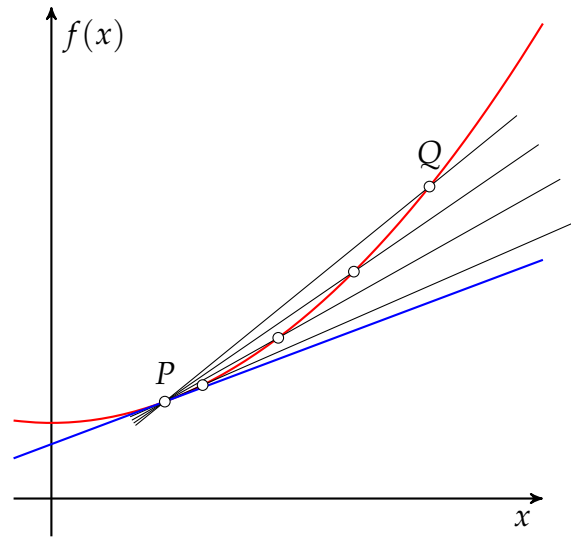
$$\lim_{h \rightarrow 0} \frac{f(a+h) - f(a)}{h}$$

bestaat. In dit geval wordt deze limiet genoteerd als  $f'(a)$  en noemen we deze waarde de AFGELEIDE van  $f$  in  $a$ . ■

**Opmerking 6.10** Wanneer de functie  $f$  afleidbaar is in elk punt van haar domein dan noemen we de functie zelf afleidbaar. De afgeleide functie wordt dan ook aangeduid met  $f'$ . ■

Wanneer een functie  $f$  afleidbaar is in  $a$  dan definiëren we de RAAKLIJN aan de functie in  $(a, f(a))$  als de rechte door dat punt met richtingscoëfficiënt  $f'(a)$ . De vergelijking voor deze rechte is m.a.w.

$$y - f(a) = f'(a)(x - a).$$



**Figuur 6.4:** Illustratie van het feit dat de secant-lijnen almaar meer op de raaklijn lijken als men  $h$  laat naderen naar nul. De blauwe lijn is de raaklijn aan de grafiek in het punt  $P$ . Die gaat in principe *niet* door het punt dat werd aangeduid op de laatste secant-lijn, maar de inkt is al zo dik dat dit wel het geval lijkt! In de buurt van  $P$  gedraagt de rode functie zich min of meer als de blauwe raaklijn.

We berekenen nu de afgeleide van enkele eenvoudige functies.

1. Stel dat  $f(x) = c$ , m.a.w. de grafiek van  $f$  is een horizontale rechte, dan geldt

$$\lim_{h \rightarrow 0} \frac{f(a+h) - f(a)}{h} = \lim_{h \rightarrow 0} \frac{c - c}{h} = \lim_{h \rightarrow 0} \frac{0}{h} = 0.$$

De afgeleide van een constante functie is m.a.w. steeds gelijk aan nul, en de raaklijn in elk punt is gelijk aan de rechte zelf.

2. Stel  $f(x) = mx$ , i.e. de grafiek van  $f$  is een rechte door de oorsprong en de richtingscoëfficiënt van de rechte is gelijk aan  $m$ . De afgeleide van  $f$  in het punt  $a$  wordt gegeven door

$$\lim_{h \rightarrow 0} \frac{f(a+h) - f(a)}{h} = \lim_{h \rightarrow 0} \frac{m(a+h) - ma}{h} = \lim_{h \rightarrow 0} \frac{mh}{h} = \lim_{h \rightarrow 0} m = m.$$

De vergelijking van de raaklijn in  $(a, ma)$  is gelijk aan

$$y - ma = m(x - a) \iff y = mx.$$



Ook in dit geval is de raaklijn aan de rechte gelijk aan de rechte zelf.

3. Stel  $f(x) = x^2$ , i.e. de grafiek van  $f$  is een parabool. De afgeleide van  $f$  in het punt  $a$  wordt gegeven door

$$\begin{aligned}\lim_{h \rightarrow 0} \frac{f(a+h) - f(a)}{h} &= \lim_{h \rightarrow 0} \frac{(a+h)^2 - a^2}{h} \\ &= \lim_{h \rightarrow 0} \frac{(a^2 + 2ah + h^2) - a^2}{h} \\ &= \lim_{h \rightarrow 0} \frac{2ah + h^2}{h} \\ &= \lim_{h \rightarrow 0} 2a + h \\ &= 2a.\end{aligned}$$

Er geldt voor de functie  $f(x) = x^2$  dat

$$f'(a) = 2a, \quad \text{voor alle } a.$$

We kunnen dit ook schrijven m.b.v. de AFGELEIDE FUNCTIE  $f'$ , en we definiëren in dit geval

$$f': \mathbb{R} \rightarrow \mathbb{R}: x \mapsto 2x.$$

4. Voor de ReLu-functie vinden we het volgende: als  $a < 0$  dan is de waarde van de afgeleide gelijk aan nul, want de ReLu-functie is daar gelijk aan een constante functie (met constante gelijk aan nul). Als  $a > 0$  dan is de waarde van de afgeleide gelijk aan 1, want de ReLu-functie is daar gelijk aan een lineaire functie met richtingscoëfficiënt gelijk aan 1.

In het punt  $a = 0$  bestaat de afgeleide echter niet! Inderdaad, als  $h$  nadert naar nul langs de positieve kant, dan vinden we

$$\lim_{h \rightarrow 0^+} \frac{f(0+h) - f(0)}{h} = \lim_{h \rightarrow 0^+} \frac{(0+h) - 0}{h} = 1.$$

Maar, als  $h$  nadert naar nul langs de negatieve kant dan wordt dit:

$$\lim_{h \rightarrow 0^-} \frac{f(0+h) - f(0)}{h} = \lim_{h \rightarrow 0^-} \frac{0 - 0}{h} = 0.$$

Omdat deze twee limieten een verschillend resultaat opleveren is de limiet zelf (en dus de afgeleide in 0) niet gedefinieerd. M.a.w.

$$\text{ReLU}'(x): \mathbb{R} \rightarrow \mathbb{R}: x \rightarrow \begin{cases} 0 & \text{als } x < 0 \\ \text{niet gedefinieerd} & \text{als } x = 0 \\ 1 & \text{als } x > 0 \end{cases}$$

Het feit dat de afgeleide in nul niet bestaat voor de ReLu functie manifesteert zich in het feit dat de grafiek van de ReLu functie in  $a = 0$  een “knik” vertoont. Het functieverloop is er niet vloeiend.

5. Voor de stapfunctie is de afgeleide functie eveneens niet gedefinieerd in  $a = 0$ , en is nul overal anders. De stapfunctie geeft dus zeer weinig bruikbare informatie in haar afgeleide. Dit is ook de reden dat deze functie niet vaak meer wordt gebruikt in moderne toepassingen van machinaal leren.
6. Beschouw de functie  $f(x) = 1/x$ . We bepalen de afgeleide van deze functie in een punt  $a \neq 0$ .

$$\begin{aligned} \lim_{h \rightarrow 0} \frac{f(a+h) - f(a)}{h} &= \lim_{h \rightarrow 0} \frac{1/(a+h) - 1/a}{h} \\ &= \lim_{h \rightarrow 0} \frac{a - (a+h)}{(a+h)ah} \\ &= \lim_{h \rightarrow 0} \frac{-h}{(a+h)ah} \\ &= \lim_{h \rightarrow 0} \frac{-1}{(a+h)a} \\ &= -\frac{1}{a^2}. \end{aligned}$$

We vinden m.a.w. dat

$$f'(x) = -\frac{1}{x^2}, \quad \text{voor } x \neq 0.$$

### 6.2.1 Oefeningen

1. Bereken de afgeleide functie van  $f(x) = \sqrt{x}$  voor  $x > 0$  m.b.v. de definitie van afgeleide. Tip: vermenigvuldig in de limiet teller en noemer met de “toegevoegde term” om de vierkantswortels te verwijderen.

2. Bereken de afgeleide functie van de sinus en cosinus functie m.b.v. de definitie van afgeleide. Maak hiertoe gebruik van de regels van Simpson die worden gegeven door

$$\begin{aligned}\sin(x) - \sin(y) &= 2 \cos\left(\frac{x+y}{2}\right) \sin\left(\frac{x-y}{2}\right) \\ \cos(x) - \cos(y) &= -2 \sin\left(\frac{x+y}{2}\right) \sin\left(\frac{x-y}{2}\right).\end{aligned}$$

In sectie 6.1.3 heb je ook een interessante limiet gezien die kan helpen bij het beantwoorden van deze vraag.

### 6.3 Afgeleide van som en product

Het wordt al heel snel nogal langdradig om afgeleiden te bepalen aan de hand van Definitie 6.9. Er zijn gelukkig regels die het mogelijk maken om snel en volledig procesmatig de waarde van afgeleiden en afgeleide functies te bepalen.

Voor een som van functies ligt de regel voor de hand. Stel dat  $f = g + h$  en dat we de waarde van de afgeleide in  $x = a$  wensen te bepalen. De waarde van de afgeleide van een functie in een bepaald punt zegt hoe *de waarde van de functie verandert als de waarde van de input een heel klein beetje wordt gewijzigd*. Het is dan ook niet moeilijk om in te zien dat

$$f'(a) = g'(a) + h'(a),$$

i.e. “de afgeleide van een som is de som van de afgeleiden” (op voorwaarde dat deze bestaan).

Voor het product van twee functies is de regel echter iets ingewikkelder, daar geldt immers

$$(gh)'(a) = g'(a)h(a) + g(a)h'(a).$$

Dit kan (als je dat zou willen) geverifieerd worden a.d.h.v. de formele definitie voor afgeleiden maar kan ook gezien worden als men het product  $gh$  bekijkt als een *oppervlakte*.

We kunnen de functies  $g$  en  $h$  m.b.v. hun afgeleiden als volgt benaderen door een lineaire functie in de buurt van  $x = a$ :

$$g(a + \Delta a) \approx g(a) + g'(a)\Delta a \quad \text{en} \quad h(a + \Delta a) \approx h(a) + h'(a)\Delta a,$$

waarbij  $\Delta a$  de suggestie geeft dat we ons een heel klein beetje verplaatsen uit de buurt van  $a$ . We wensen nu een lineaire benadering te vinden voor de functie  $gh$  in de buurt van  $a$ . Uit de voorgaande benaderingen volgt:

$$\begin{aligned} g(a + \Delta a)h(a + \Delta a) & \approx (g(a) + g'(a)\Delta a)(h(a) + h'(a)\Delta a) \\ & = g(a)h(a) + (g'(a)h(a) + g(a)h'(a))\Delta a + g'(a)h'(a)(\Delta a)^2. \end{aligned}$$

Tenslotte houden we rekening met het feit dat we een lineaire benadering wensen en dat  $(\Delta a)^2$  *veel kleiner* is dan  $\Delta a$ . De term met  $(\Delta a)^2$  verwaarlozen we en we zien inderdaad de (gekende) formule voor afgeleide van een product verschijnen.

We vatten deze resultaten samen in de volgende eigenschap.

**Eigenschap 6.11** Als  $g$  en  $h$  twee functies zijn die afleidbaar zijn in  $a$  dan geldt dat  $g + h$  en  $gh$  afleidbaar zijn in  $a$  en

$$(g + h)'(a) = g'(a) + h'(a)$$

en

$$(gh)'(a) = g'(a)h(a) + g(a)h'(a). \quad \blacksquare$$

**Eigenschap 6.12** Een belangrijk speciaal geval krijgt men wanneer de functie  $g$  een constante functie is, in dit geval geldt

$$(ch)'(a) = ch'(a), \quad \text{voor alle } c \in \mathbb{R}. \quad \blacksquare$$

We geven nu een aantal voorbeelden van het toepassen van deze eigenschap.

1. Als  $f(x) = x^2 + bx + c$ , dan geldt

$$\begin{aligned} f'(x) &= (x^2 + bx + c)' && \text{afgeleide van een som} \\ &= (x^2)' + (bx)' + (c)' && \text{gekende afgeleiden en Eigenschap 6.12} \\ &= 2x + b(x)' + 0 && \text{gekende afgeleide} \\ &= 2x + b. \end{aligned}$$

2. Als  $f(x) = x^3$ , dan geldt

$$\begin{aligned}
 f'(x) &= (x^3)' && \text{schrijf } x^3 \text{ als een product} \\
 &= (x \times x^2)' && \text{afgeleide product} \\
 &= x' \times x^2 + x \times (x^2)' && \text{toepassing gekende afgeleiden} \\
 &= 1 \times x^2 + x \times (2x) && \text{vereenvoudigen} \\
 &= 3x^2.
 \end{aligned}$$

3. Als we veronderstellen dat we reeds weten dat

$$\sin'(a) = \cos(a) \quad \text{en} \quad \cos'(a) = -\sin(a)$$

voor alle  $a$ , dan kunnen we de afgeleide berekenen van  $f(x) = \cos(x) \sin(x)$ :

$$\begin{aligned}
 f'(x) &= (\cos(x) \sin(x))' \\
 &= (\cos(x))' \sin(x) + \cos(x) (\sin(x))' \\
 &= -\sin(x) \sin(x) + \cos(x) \cos(x) \\
 &= \cos^2(x) - \sin^2(x).
 \end{aligned}$$

4. We kunnen ook producten van drie functies aanpakken. Stel

$$f(x) = (x^2 + x) \cos(x) \sin(x),$$

dan is

$$\begin{aligned}
 f'(x) &= (x^2 + x)' \cos(x) \sin(x) + (x^2 + x) (\cos(x) \sin(x))' \\
 &= (x^2 + x)' \cos(x) \sin(x) \\
 &\quad + (x^2 + x) ((\cos(x))' \sin(x) + \cos(x) (\sin(x))') \\
 &= (2x + 1) \cos(x) \sin(x) + (x^2 + x) (\cos^2(x) - \sin^2(x)).
 \end{aligned}$$

Met de eigenschappen die we nu kennen is het echter onmogelijk om gemakkelijk de afgeleide te bepalen van een functie als  $\sin(x^2)$  omdat dit een *samenstelling* van een aantal functies is.

## 6.4 De kettingregel voor afgeleiden

De allerbelangrijkste rekenregel voor afgeleiden is ongetwijfeld de KETTING-REGEL die zegt hoe we de afgeleide kunnen bepalen van een samenstelling van functies in termen van de afgeleiden van de samenstellende functies.

Stel dat  $f = h \circ g$ , i.e.  $f(a) = h(g(a))$ . Om de afgeleide van  $f$  te bepalen in het punt  $a$  wensen we te weten hoe  $f$  zich gedraagt in de buurt van  $a$ . M.a.w. we wensen  $f$  in de buurt van  $a$  te benaderen door een lineaire functie:

$$f(a + \Delta a) \approx f(a) + (\text{hier komt de afgeleide van } f \text{ in } a)\Delta a.$$

We weten dat

$$f(a + \Delta a) = h(g(a + \Delta a)).$$

We weten hoe de functie  $g$  zich gedraagt in buurt van  $a$ :

$$g(a + \Delta a) \approx g(a) + g'(a)\Delta a.$$

Als we dit invullen in de voorgaande formule dan vinden we

$$f(a + \Delta a) \approx h(\underbrace{g(a)}_b + \underbrace{g'(a)\Delta a}_{\Delta b}) = h(b + \Delta b).$$

We weten echter dat

$$h(b + \Delta b) \approx h(b) + h'(b)\Delta b,$$

zodat we finaal krijgen:

$$f(a + \Delta a) \approx h(g(a)) + h'(g(a))g'(a)\Delta a.$$

De afgeleide van een samengestelde functie is dus het product van de afgeleiden van de samenstellende functies, waarbij de afgeleide van de buitenste functie wordt geëvalueerd in de functiewaarde bekomen door het toepassen van de binnenste functie in het punt waar we de afgeleide wensen te berekenen.

We hebben m.a.w. de volgende eigenschap plausibel gemaakt, zonder die echter formeel aan te tonen.

**Eigenschap 6.13** Als de functie  $g$  afleidbaar is in het punt  $a$  en de functie  $h$  is afleidbaar in  $g(a)$ , dan is de functie  $h \circ g$  afleidbaar in  $a$  en er geldt:

$$(h \circ g)'(a) = h'(g(a)) \times g'(a). \quad \blacksquare$$

We passen de kettingregel toe op een aantal voorbeelden.

1. Stel  $f(x) = (2x + 1)^2$ . Deze functie is de samenstelling van de functies  $g(x) = 2x + 1$  gevolgd door de functie  $h(x) = x^2$ . We weten dat

$$h'(x) = 2x \quad \text{en} \quad g'(x) = 2.$$

Om de afgeleide functie  $f'(x)$  te vinden moeten we het product nemen van deze twee functies, waarbij de de functie  $h'$  moeten evalueren in het punt  $g(x)$ :

$$f'(x) = h'(g(x)) \times g'(x) = 2(2x + 1) \times 2 = 8x + 4.$$

In dit voorbeeld kunnen we de afgeleide functie ook op een andere manier bepalen, nl. door het kwadraat in de functie  $f$  uit te werken. Op die manier krijgen we

$$f(x) = (2x + 1)^2 = 4x^2 + 4x + 1.$$

Als we deze functie nu afleiden m.b.v. Eigenschappen 6.11 en 6.12 dan vinden we onmiddellijk:

$$f'(x) = 8x + 4,$$

wat hetzelfde resultaat is! Dit is een bevestiging van de correctheid van de kettingregel.

2. Stel  $f(x) = \sin(x^2)$ , dan is  $f$  de samenstelling van  $g(x) = x^2$  gevolgd door  $h(x) = \sin(x)$ . We weten reeds dat

$$h'(x) = \cos(x) \quad \text{en} \quad g'(x) = 2x.$$

Opnieuw moeten we de functie  $h'$  evalueren in  $g(x)$ , en we vinden

$$f'(x) = h'(g(x)) \times g'(x) = \cos(x^2) \times 2x = 2x \cos(x^2).$$

3. Net zoals een product of som niet hoeft te bestaan uit twee factoren of termen, hoeft een samenstelling van functies ook niet te stoppen bij de samenstelling van twee functies. Beschouw

$$f(x) = \sin^3(x^2 + \cos(x)).$$

Dit is de samenstelling van drie functies:

- $g_1(x) = x^2 + \cos(x)$ , met afgeleide  $g_1'(x) = 2x - \sin(x)$

- $g_2(x) = \sin(x)$ , met afgeleide  $g_2'(x) = \cos(x)$
- $g_3(x) = x^3$ , met afgeleide  $g_3'(x) = 3x^2$

Om de finale functiewaarde te berekenen gaan we als volgt tewerk:

$$x \xrightarrow{g_1} x^2 + \cos(x) \xrightarrow{g_2} \sin(x^2 + \cos(x)) \xrightarrow{g_3} \sin^3(x^2 + \cos(x))$$

Om de afgeleide functie te berekenen start je van rechts: elke toepassing van een functie geeft een factor in de afgeleide. Deze factor bestaat uit de afgeleide van de functie die werd toegepast, *geëvalueerd in het punt waaruit werd vertrokken*. In dit geval werden er drie functies samengesteld en dus zullen er drie factoren zijn in de afgeleide, de eerste factor is

$$g_3'(\sin(x^2 + \cos(x))) = 3(\sin(x^2 + \cos(x)))^2,$$

de tweede factor is

$$g_2'(x^2 + \cos(x)) = \cos(x^2 + \cos(x))$$

en de derde factor, tenslotte is,

$$g_1'(x) = 2x - \sin(x).$$

De finale afgeleide functie is

$$f'(x) = 3(\sin(x^2 + \cos(x)))^2 \cos(x^2 + \cos(x)) (2x - \sin(x)).$$

### 6.4.1 Oefeningen

1. Bereken de afgeleide functie van de volgende functies m.b.v. de methodes gezien in deze en vorige sectie (en gebruikmakend van reeds gekende afgeleide functies).

- a)  $f(x) = (x + 7)^{10}$
- b)  $f(x) = x(\sin(x) + \cos(x))$
- c)  $f(x) = x(\sin^2(x) + \cos^2(x))$
- d)  $f(x) = \frac{x}{x^2 + 1}$
- e)  $f(x) = \frac{\sin(x)}{\cos(x)}$  (of  $f(x) = \tan(x)$ ).



## 6.5 Het belang van de afgeleide

Wanneer men het maximum (of minimum) van een functie  $f$  over een bepaald interval wenst te bepalen, dan komt dit vaak neer op het zoeken naar die punten waarvoor de afgeleide functie de waarde nul aanneemt.

We starten met een definitie om precies aan te duiden wat we bedoelen met een maximum.

**Definitie 6.14** Veronderstel dat  $f$  een reële functie en dat  $A$  een deelverzameling is van het domein van  $f$ . We zeggen dat het punt  $x$  een MAXIMUM PUNT is voor  $f$  in  $A$ , als geen enkele  $y \in A$  een functiewaarde heeft die strikt groter is, of

$$f(y) \leq f(x), \quad \text{voor alle } y \in A.$$

De waarde  $f(x)$  noemen we de MAXIMUM WAARDE of kortweg het MAXIMUM van  $f$  over  $A$ . ■

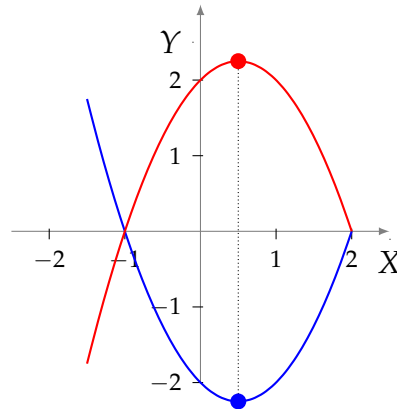
**Opmerking 6.15** Merk op dat een functie meerdere maximum punten kan hebben over  $A$ . ■

Een gelijkaardige definitie kan opgezet worden voor de begrippen MINIMUM PUNT en MINIMUM WAARDE. Merk ook op dat als  $x$  een maximum punt is voor  $f$  in  $A$ , dan is datzelfde punt een minimum punt voor de functie  $-f$  in  $A$ . Dit betekent dat maximaliseren en minimaliseren van een functie in essentie equivalent zijn aan elkaar. Figuur 6.5 geeft hiervan een illustratie. In deze figuur zie je ook dat in het maximum punt  $x = 1/2$  de waarde van de afgeleide van de functie gelijk is aan nul. Dit is geen toeval zoals de volgende eigenschap aangeeft.

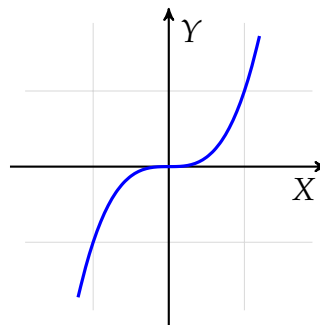
**Eigenschap 6.16** Als  $f$  een functie is die gedefinieerd is over een open interval  $(a, b)$ , en als  $x$  een maximum punt is voor  $f$  in dit interval waarbij  $f$  afleidbaar is in  $x$ , dan is de waarde van de afgeleide in dat maximum punt steeds gelijk aan nul, i.e.

$$f'(x) = 0. \quad \blacksquare$$

Het is hierbij zeer belangrijk om op te merken dat het omgekeerde niet geldt: het kan best zijn dat de waarde van  $f'(x)$  gelijk is aan nul voor een bepaalde  $x$  zonder dat die  $x$  een maximum (of minimum) punt is. Bv. als  $f(x) = x^3$  dan is  $f'(x) = 3x^2$ , zodat  $f'(0) = 0$  maar het punt  $x = 0$  is geen minimum



**Figuur 6.5:** Omzetten van minimalisatie van  $(x + 1)(x - 2)$  over  $[-1, 2]$  naar maximalisatie van  $-(x + 1)(x - 2)$  over hetzelfde interval. Het minimum van  $(x + 1)(x - 2)$  over  $[-1, 2]$  wordt bereikt voor  $x = 1/2$  (het midden van de nulpunten) met bijhorende functiewaarde  $y = -9/4$ . Het maximum van  $-(x + 1)(x - 2)$  wordt bereikt voor dezelfde  $x$ -waarde  $1/2$  maar de bijhorende functiewaarde is nu  $y = 9/4$ .

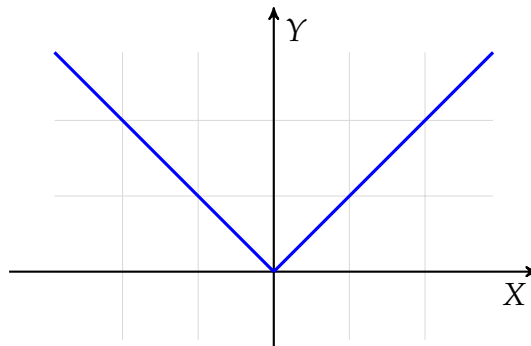


**Figuur 6.6:** De grafiek van functie  $f(x) = x^3$ . Er geldt dat  $f'(0) = 0$  terwijl  $x = 0$  een minimum noch maximum punt is.

en ook geen maximum punt. De functie  $f(x) = x^3$  is een strikt stijgende functie en heeft minima noch maxima

Ook belangrijk om op te merken: maximum en minimum punten kunnen ook gevonden worden waar de functie eventueel niet afleidbaar is. Beschouw bv. de absolute waarde functie:

$$f: \mathbb{R} \rightarrow \mathbb{R}: x \mapsto f(x) = |x| = \begin{cases} -x & \text{als } x \leq 0 \\ x & \text{anders.} \end{cases}$$



**Figuur 6.7:** De grafiek van de absolute waarde functie. Het minimum punt is  $x = 0$ , terwijl de functie daar niet afleidbaar is.

Het minimum punt voor deze functie is  $x = 0$ , terwijl de functie daar niet afleidbaar is.

Dit toont aan dat de maximum punten van een functie  $f$  over een verzameling  $A$  kunnen gevonden worden onder de volgende punten:

1. De KRITISCHE PUNTEN van  $f$ , i.e. de punten  $x$  waarvoor  $f'(x) = 0$ .
2. Punten waarbij  $f$  niet afleidbaar is.
3. Op de randen van  $A$  (bv. de grenspunten van een interval).

### 6.5.1 Oefeningen

1. Gegeven twee vectoren

$$\mathbf{a} = \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \quad \text{en} \quad \mathbf{b} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_n \end{bmatrix}.$$

Vind  $p \in \mathbb{R}$  zodanig dat

$$\|p\mathbf{a} - \mathbf{b}\|$$

minimaal is. Vergelijk met hetgeen in Sectie 1.5 werd gezien. Wat hebben we gevonden?

2. Gegeven een dataset bestaande uit  $m$  koppels

$$(x^{(i)}, y^{(i)}) \quad \text{met } i \in \{1, 2, \dots, m\}.$$

Veronderstel dat we de best passende *horizontale* rechte  $y = c$  willen vinden. Minimaliseer hiertoe de MSE:

$$\frac{1}{m} \sum_{i=1}^m (c - y^{(i)})^2$$

als een functie van  $c$ . Geef een interpretatie aan de gevonden waarde van  $c$ .

3. We wensen een cilindervormig blikje te ontwerpen met een bepaald volume  $V$ . Om de materiaalkosten van het blikje zo klein mogelijk te maken wensen we de hoeveelheid materiaal nodig om het blikje te produceren te minimaliseren. Vind de straal van het grondvlak en de hoogte van het blikje (in functie van het vereiste volume  $V$ ).

## 6.6 De exponentiële en logaritmische functie

### 6.6.1 Exponentiële groei

We starten deze sectie met een (hopelijk) gekend voorbeeld i.v.m. samengestelde intrest, i.e. intrest waarbij je de verdiende intrest opnieuw belegt.

Stel dat je start met een initieel kapitaal  $A$  en dat de intrestvoet gelijk is aan  $r$  bv. voor 5 procent intrest geldt  $r = 0.05$ . Als we  $A_k$  het kapitaal na  $k$  jaar noemen dan vinden we:

$$\begin{aligned} A_0 &= A \\ A_1 &= (1 + r)A_0 = (1 + r)A \\ A_2 &= (1 + r)A_1 = (1 + r)^2 A \\ A_3 &= (1 + r)A_2 = (1 + r)^3 A. \end{aligned}$$

Voor een willekeurige  $k$  zien we dat

$$A_k = (1 + r)A_{k-1}, \quad \text{voor } k \geq 1. \quad (6.2)$$

Je ziet dat in het algemeen zal gelden dat

$$A_k = (1 + r)^k A \quad \text{voor } k \geq 0.$$

Dit is een voorbeeld van EXPONENTIËLE GROEI. Wanneer je de intrest niet zou herbeleggen, dan zou je finale kapitaal (slechts) gelijk zijn aan  $(1 + rk)A$ .

**Voorbeeld 6.17** We geven een cijfervoorbeeld om de kracht van exponentiële groei duidelijk te maken. Stel dat je start met  $A = 100$  EUR en dat je een bank hebt gevonden die 5 procent intrest betaalt. Als je begint te sparen op je 25ste en je neemt het geld op bij je pensioen op je 65ste<sup>3</sup>, dan heb je met samengestelde intrest na 40 jaar een kapitaal van

$$A_{40} = (1 + 0.05)^{40} \times 100 \approx 704 \text{ EUR.}$$

Als je de intresten niet herbelegt dan heb je

$$(1 + 40 \times 0.05) \times 100 = 300 \text{ EUR.}$$

Veronderstel vervolgens dat de bank de intrest niet jaarlijks maar halfjaarlijks uitbetaalt, dus bv. in plaats van 1 keer 5 procent intrest uit te betalen, betaalt de bank 2 keer per jaar 2.5 procent intrest. In dit geval krijg je na 1 jaar

$$\begin{aligned} A_{1/2} &= \left(1 + \frac{r}{2}\right)A && \text{kapitaal na een half jaar} \\ A_1 &= \left(1 + \frac{r}{2}\right)A_{1/2} = \left(1 + \frac{r}{2}\right)^2 A && \text{kapitaal na één jaar.} \end{aligned}$$

De factor waarmee je geld elk jaar aangroeit is in dit geval gelijk aan

$$\left(1 + \frac{r}{2}\right)^2.$$

Als de bank nu *maandelijks* je intrest zou uitbetalen dan is het niet moeilijk om in te zien dat de factor waarmee je geld jaarlijks aangroeit gegeven wordt door

$$\left(1 + \frac{r}{12}\right)^{12}.$$

Bij dagelijkse uitbetaling van de intrest wordt dit

$$\left(1 + \frac{r}{365}\right)^{365}.$$

**Voorbeeld 6.18** We gebruiken opnieuw het cijfervoorbeeld van daarnet, en berekenen ons kapitaal na 40 jaar bij halfjaarlijkse, maandelijks en dagelijkse uitbetaling van de intrest.

<sup>3</sup>Gesteld dat tegen dan de pensioenleeftijd al niet veel hoger is.

frequentie	aangroefactor per jaar	totaal na 40 jaar
jaarlijks	1.05	704 EUR
halfjaarlijks	1.050625	720.96 EUR
maandelijks	1.051162	735.84 EUR
dagelijks	1.051267	738.80 EUR

Je ziet dat de aangroefactor per jaar en het eindbedrag na 40 jaar stijgen naarmate de frequentie waarmee de intrest wordt uitbetaald groter wordt maar dat er toch ook een zekere afvlakking blijkt in te zitten, en dat zelfs al werd de intrest elke nanoseconde (of nog sneller) herberekend de aangroefactor en het eindbedrag een bepaald plafond zouden bereiken. ■

We nemen nu de logische vervolgstap en bekijken

$$\lim_{n \rightarrow \infty} \left(1 + \frac{r}{n}\right)^n.$$

We bekijken eerst en vooral de waarde van deze limiet voor  $r = 1$ . We gebruiken Python om numeriek vast te stellen wat deze waarde is:

```
factor = lambda n : (1+1/n)**n
[(10**k, factor(10**k)) for k in range(1, 12, 2)]
```

Deze code<sup>4</sup> geeft de volgende uitvoer:

```
[(10, 2.5937424601000023),
 (1000, 2.7169239322355936),
 (100000, 2.7182682371922975),
 (10000000, 2.7182816941320818),
 (1000000000, 2.7182820520115603),
 (100000000000, 2.71828205335711)]
```

De waarde van deze limiet is een beroemd getal in de wiskunde en is vernoemd naar de wiskundige Leonhard Euler. We noteren deze limiet met  $e$ . Dit getal is net als  $\pi$  een irrationaal getal, en gaat dus oneindig lang door na de komma zonder zich ooit te herhalen. De eerste paar decimalen van  $e$  zijn:

$$e \approx 2.718281 \dots$$

<sup>4</sup>Opletten: als je  $k$  groter maakt in deze code geeft dit niet noodzakelijk betere resultaten. Dit is zeker niet de beste manier om het getal  $e$  te berekenen!

## 6.6.2 De exponentiële functie

De limiet

$$\lim_{n \rightarrow \infty} \left(1 + \frac{r}{n}\right)^n$$

kan nu gebruikt worden om een functie te definiëren: de EXPONENTIËLE FUNCTIE<sup>5</sup>:

$$\exp: \mathbb{R} \rightarrow \mathbb{R}: x \mapsto \exp(x) = \lim_{n \rightarrow \infty} \left(1 + \frac{x}{n}\right)^n.$$

De exponentiële functie wordt ook dikwijls genoteerd als  $e^x$ , wat suggereert dat we het getal  $e$  tot de macht  $x$  verheffen.

We geven hieronder, zonder bewijs, de belangrijkste eigenschappen van deze exponentiële functie.

**Eigenschap 6.19** De exponentiële functie heeft de volgende eigenschappen:

- De exponentiële functie is steeds *strikt positief*. Het domein is  $\mathbb{R}$  en het beeld is  $\mathbb{R}_0^+$ .
- De exponentiële functie is *strikt stijgend*:

$$x_1 < x_2 \iff \exp(x_1) < \exp(x_2).$$

- De exponentiële functie is overal afleidbaar én *de afgeleide functie van de exponentiële functie is zichzelf*<sup>6</sup>:

$$\exp'(x) = \exp(x) \quad \text{voor alle } x \in \mathbb{R}.$$

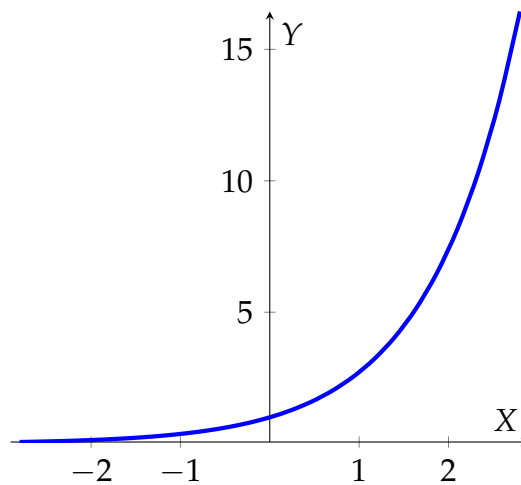
- Er geldt dat

$$\exp(x) \exp(y) = \exp(x + y) \quad \text{voor alle } x, y \in \mathbb{R}. \quad \blacksquare$$

In Figuur 6.8 tonen we een deel van de grafiek van de exponentiële functie. Je ziet dat  $\exp(0) = 1$  en dat de functie aan de linkerkant zeer langzaam nadert naar 0 maar dat de functie aan de rechterkant enorm snel stijgt.

<sup>5</sup>Je hoeft niet te snappen waarom dit wiskundig zinvol is.

<sup>6</sup>Dit wordt, samen met de voorwaarde  $\exp(0) = 1$  soms ook genomen als de *definitie* van de exponentiële functie.



**Figuur 6.8:** Grafiek van de exponentiële functie.

### 6.6.3 De natuurlijke logaritmische functie

Omdat de exponentiële functie een strikt stijgende functie is, is haar inverse eveneens een functie. Deze functie wordt de NATUURLIJKE LOGARITMISCHE FUNCTIE genoemd, en wordt genoteerd met  $\ln(x)$ .

Aangezien het beeld van de exponentiële functie  $\mathbb{R}_0^+$  is, is de natuurlijke logaritmische functie enkel gedefinieerd voor strikt positieve reële getallen. Het beeld van de natuurlijke logaritmische functie is gelijk aan het domein van de exponentiële functie en is m.a.w. gelijk aan de verzameling van de reële getallen  $\mathbb{R}$ . Er geldt bijgevolg dat

$$\ln: \mathbb{R}_0^+ \rightarrow \mathbb{R}: x \mapsto \ln(x).$$

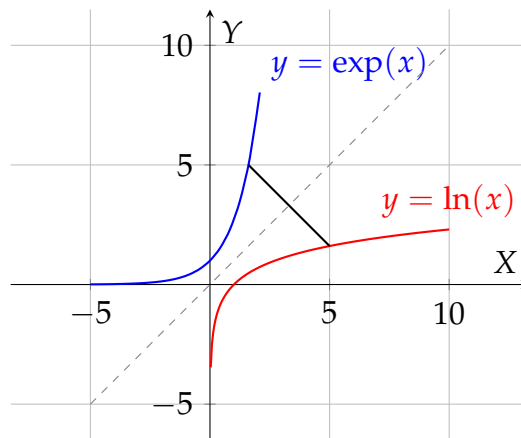
Omdat  $\ln$  en  $\exp$  elkaars inverse functie zijn geldt:

$$y = \ln(x) \iff \exp(y) = x.$$

In Figuur 6.9 worden de natuurlijke logaritmische functie en de exponentiële functie allebei afgebeeld. Je bekomt de grafiek van de logaritmische functie door die van de exponentiële functie loodrecht te spiegelen rond de eerste bissectrice  $y = x$ .

Hieronder worden, zonder bewijs, een aantal eigenschappen gegeven van de natuurlijke logaritmische functie.





**Figuur 6.9:** Grafiek van de exponentiële functie en de logaritmische functie. Omdat deze functies elkaars inverse zijn, krijg je de grafiek van  $\ln(x)$  door de grafiek van  $\exp(x)$  te spiegelen rond de eerste bissectrice (aangeduid in stippellijn). Zo'n spiegeling tussen de punten  $(\ln(5), 5)$  (op de blauwe curve) en  $(5, \ln(5))$  op de rode curve is aangeduid door het zwarte lijnsegment.

**Eigenschap 6.20** De natuurlijke logaritmische functie heeft de volgende eigenschappen:

- De natuurlijk logaritmische functie is afleidbaar in haar domein en er geldt:

$$\ln'(x) = \frac{1}{x}.$$

- De functiewaarde in 1 is gelijk aan nul:  $\ln(1) = 0$ .
- De natuurlijke logaritmische functie is strikt stijgend.
- Er geldt dat

$$\ln(x_1 \times x_2) = \ln(x_1) + \ln(x_2)$$

en

$$\ln(x^p) = p \ln(x), \quad \text{voor elke macht } p \text{ en } x > 0.$$

■

### 6.6.4 Andere grondtallen

Exponentiële functies kunnen ook voor andere *grondtallen* dan  $e$  worden gedefinieerd. Als men bv. praat over een exponentiële functie

$$f: \mathbb{R} \rightarrow \mathbb{R}: x \mapsto f(x) = 2^x,$$

dan heeft men het intuïtief over een functie die machten neemt van het grondtal 2, en dat is ook een goede intuïtieve manier om daarover na te denken<sup>7</sup>. De precieze definitie van  $2^x$  (en van  $a^x$  voor  $a > 0$  en  $a \neq 1$ ) maakt gebruik van het feit dat  $\exp$  en  $\ln$  inverse functies zijn:

$$a^x \stackrel{\text{def}}{=} \exp(x \ln(a)), \quad \text{voor } a > 0 \text{ en } a \neq 1.$$

Gebruikmakend van de kettingregel voor afleiden vinden we dan

$$(a^x)' = \ln(a) \exp(x \ln(a)) = \ln(a) a^x.$$

Aangezien  $a^x$  steeds strikt positief is (want het is het resultaat van het toepassen van de functie  $\exp$ ) is de afgeleide strikt positief wanneer  $a > 1$  en strikt negatief wanneer  $0 < a < 1$ . In het eerste geval hebben we te maken met een strikt stijgende functie, in het tweede geval met een strikt dalende functie. In dit geval spreken we van een EXPONENTIËLE AFNAME.

### 6.6.5 De logistische functie

Omdat de bronnen op aarde eindig zijn kan exponentiële groei in een reële context niet eeuwig doorgaan: er zijn geen personen meer die het virus kan besmetten, de meeste mensen hebben het virale filmpje al gezien, of hebben de “challenge” reeds gedaan. De groei van de konijnenpopulatie vertraagt omdat de voedselvoorraad beperkt is, enzovoort.

Exponentiële groei wordt gekenmerkt door de vergelijking (6.2) die we hier herschrijven in termen van  $x$  i.p.v.  $A$ :

$$x_k = (1 + r)x_{k-1}.$$

We zullen er nu voor zorgen dat de groei vertraagt eens de populatie (i.e.  $x_k$ ) “groot” wordt. We doen dit door een extra factor  $(1 - x_k/C)$  toe te voegen

<sup>7</sup> Alles loopt goed tot men probeert precies te definiëren wat men bedoelt met het nemen van een irrationale macht zoals bv.  $2^\pi$ .

```

import numpy as np
import matplotlib.pyplot as plt

def simuleer(x0, r, C, n_steps=100):
    xs = np.empty(shape=(n_steps,), dtype=np.float32)
    xs[0] = x0
    for k in range(1, n_steps):
        xs[k] = (1 + r) * (1 - xs[k-1]/C) * xs[k-1]
    return xs

res = simuleer(0.001, r=0.30, C=100);

plt.plot(res);

```

**Figuur 6.10:** Python code voor het simuleren van de recursiebetrekking (6.3).

aan de vergelijking. Als we aannemen dat  $0 < x_k < C$  dan is dit een factor tussen 0 en 1. Hoe meer  $x_k$  nadert naar  $C$  hoe kleiner deze factor wordt (i.e. hoe dichter bij nul) en hoe kleiner de groei van de populatie zal zijn in de volgende stap. We bekijken nu de vergelijking:

$$x_k = (1 + r) \left(1 - \frac{x_{k-1}}{C}\right) x_{k-1}. \quad (6.3)$$

We kunnen dit proces eenvoudig in Python implementeren en bekijken hoe de resulterende kromme er uit ziet. De Python code wordt gegeven in Figuur 6.10 en de bijhorende grafiek in Figuur 6.11.

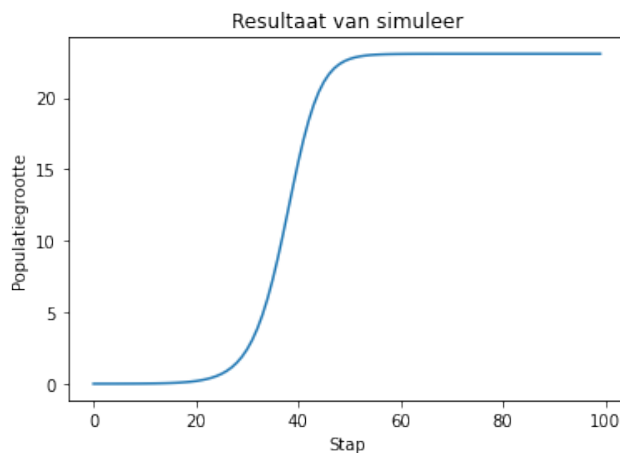
Dit soort van S-vormige kromme wordt een SIGMOÏDE genoemd. De standaardvorm van zo'n S-vormige kromme heeft als functievoorschrift:

$$\sigma: \mathbb{R} \rightarrow \mathbb{R}: x \mapsto \sigma(x) = \frac{1}{1 + \exp(-x)}. \quad (6.4)$$

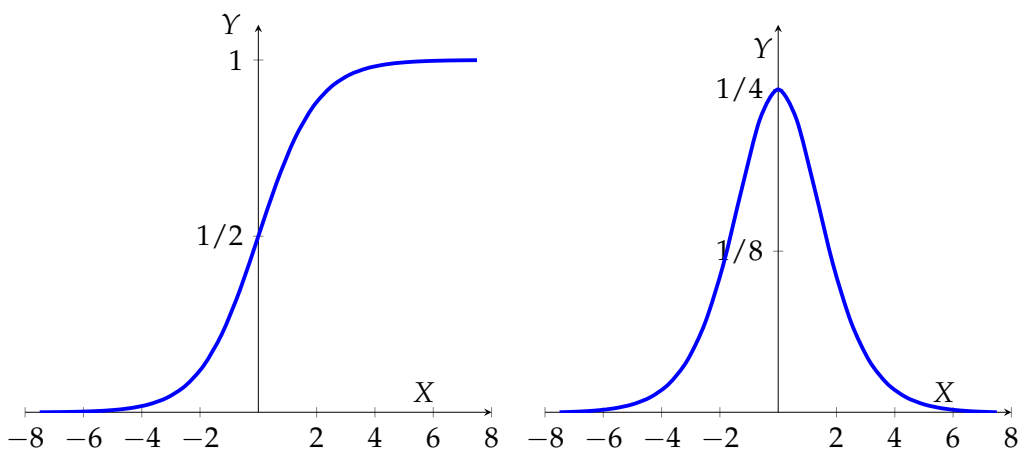
Met de technieken die we gezien hebben in dit hoofdstuk kunnen we gemakkelijk de volgende eigenschappen van de sigmoïde aantonen.

**Eigenschap 6.21** De sigmoïde  $\sigma$  heeft de volgende eigenschappen:

1. Het domein van  $\sigma$  is  $\mathbb{R}$  en het beeld is het open interval  $(0, 1)$ .



**Figuur 6.11:** Resultaat van methode `simuleer`. Er werden 100 stappen gesimuleerd. De initiële grootte van de populatie was  $1/1000$  en  $C = 100$  met  $r = 0.30$



**Figuur 6.12:** Grafiek van de sigmoïde en haar afgeleide functie. De maximumwaarde van de afgeleide wordt bereikt in 0 en is daar gelijk aan  $1/4$ . Dit lijkt niet overeen te komen met de linkse figuur die veel steiler lijkt in  $x = 0$ , maar dit komt omdat de schaal op de X en Y-as verschillend is.

2. De functie is overal afleidbaar en de afgeleide functie wordt gegeven door

$$\sigma'(x) = \sigma(x)(1 - \sigma(x)).$$

3. De functiewaarde in 0 is gelijk aan  $1/2$ :  $\sigma(0) = 1/2$ . ■

**Opmerking 6.22** Soms wordt het functievoorschrift van de sigmoïde als volgt geschreven:

$$\sigma(x) = \frac{\exp(x)}{1 + \exp(x)}.$$

Je kan gemakkelijk verifiëren dat dit een equivalent functievoorschrift is door in het “originele” functievoorschrift (6.4) teller en noemer te vermenigvuldigen met  $\exp(x)$ . ■

In de context van machinaal leren wordt de sigmoïde gebruikt bij logistische regressie, en ook als activatiefunctie bij neurale netwerken, al wordt bij “moderne” neurale netwerken nu vaker de ReLu-functie gebruikt.

### 6.6.6 Oefeningen

1. Beschouw de functie

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

- Gebruik Python om een plot te maken van deze functie.
- Wat is het domein en beeld van deze functie volgens de plot?
- Bereken de afgeleide functie van  $\tanh$ . Kan je het resultaat schrijven in termen van  $\tanh$ ?

2. Plot de functies

$$f(x) = -\ln(x), \quad \text{en} \quad g(x) = -\ln(1 - x).$$

voor  $x$  in het open interval  $(0, 1)$ . Deze functies worden gebruikt bij de kostfunctie voor logistische regressie<sup>8</sup>.

<sup>8</sup>Dit zal je zien in het opleidingsonderdeel Machine Learning

3. Beschouw functies van de vorm

$$f(x) = \exp(-\gamma(x - a)^2),$$

waarbij  $\gamma \in \mathbb{R}_0^+$  en  $a \in \mathbb{R}$  twee parameters zijn. Deze functies zijn een ééndimensionale vorm van *radiale basis functies* die bv. worden gebruikt in de context van *support vector machines*<sup>9</sup>.

- a) Wat is het domein en beeld van deze functie?
- b) Gebruik Python om een plot te maken van de grafiek van deze functie voor een aantal waarden van  $\gamma$  en  $a$ .
- c) Gebruik de afgeleide functie om te bepalen waar het maximum van deze functie wordt bereikt.
- d) Wat is het effect van de parameter  $\gamma$  op de vorm van de functie? Beschrijf dit effect kwalitatief.
- e) Kan je het effect van de parameter  $\gamma$  kwantitatief beschrijven? Waar stijgt/daalt de functie het snelst. Wat is daar de richtingscoëfficiënt van de raaklijn?

## 6.7 Newton-Raphson methode voor nulpunten

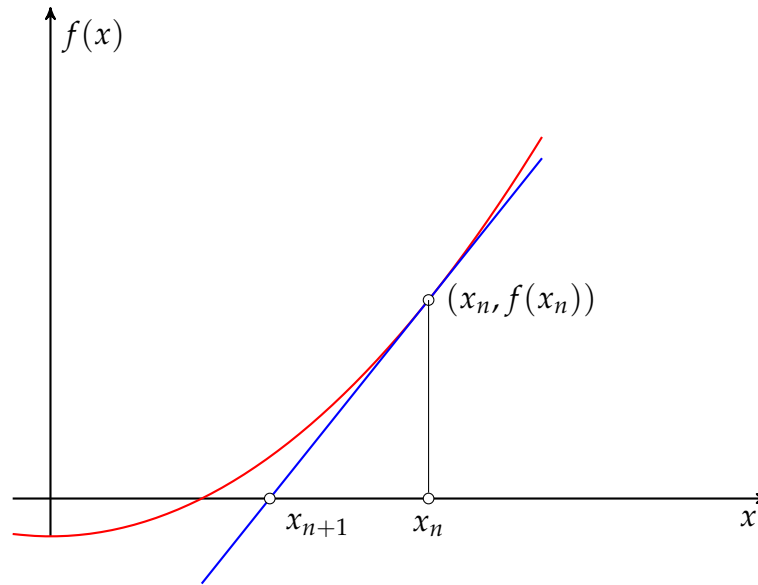
In deze sectie beschouwen we een toepassing van het gebruik van afgeleiden, waarbij we de eerste afgeleide gebruiken om een *nulpunt van een afleidbare functie* te benaderen.

De Newton-Raphson methode is een *iteratieve methode* waarbij er wordt gestart met een iniële schatting (of gok) voor een mogelijk nulpunt. Vervolgens wordt er een iteratief proces toegepast, waarbij elke iteratie ons hopelijk dichter bij het gewenste nulpunt brengt.

Stel dat de huidige iteratie ons bij een  $x$ -waarde heeft gebracht die gelijk is aan  $x_n$ . De methode van Newton-Raphson werkt als volgt: bekijk de raaklijn aan de functie  $f$  in het punt  $(x_n, f(x_n))$ . Als deze raaklijn niet horizontaal is (i.e.  $f'(x_n) \neq 0$ ), dan beschouwen we het snijpunt van deze raaklijn met de  $X$ -as als onze volgende benadering. De vergelijking van deze raaklijn is

$$y - f(x_n) = f'(x_n)(x - x_n).$$

<sup>9</sup>Een methode van machine learning die je later zal bespreken.



**Figuur 6.13:** Illustratie van één stap in het Newton-Raphson algoritme. We berekenen het snijpunt van de (blauwe) raaklijn aan de rode grafiek in  $(x_n, f(x_n))$  met de  $X$ -as. Dat wordt de nieuwe benadering  $x_{n+1}$ .

Als we nu  $y = 0$  stellen in deze vergelijking en oplossen naar  $x$  dan vinden we

$$x = x_n - \frac{f(x_n)}{f'(x_n)}.$$

We vertalen dit nu naar een update-regel:

$$x_{n+1} \leftarrow x_n - \frac{f(x_n)}{f'(x_n)}. \quad (6.5)$$

**Voorbeeld 6.23 (Vierkantswortel van 2)** We gebruiken de methode van Newton-Raphson om de vierkantswortel uit 2 te bepalen. De methode van Newton-Raphson wordt gebruikt om nulpunten van functies te vinden en  $\sqrt{2}$  is het nulpunt van de functie:

$$f(x) = x^2 - 2.$$

Om de methode van Newton-Raphson toe te passen hebben we de afgeleide functie nodig:

$$f'(x) = 2x$$

```

step = lambda x : (x*x + 2) / (2*x)
prev, x = 0, 2
while abs(prev - x) > 10**-6:
    prev, x = x, step(x)
    print(f"{x:.7f}")

```

**Figuur 6.14:** Python-code om een benadering te vinden voor de vierkantswortel uit 2 m.b.v. de methode van Newton-Raphson.

We bepalen het rechterlid van de update-regel (6.5):

$$x_n - \frac{x_n^2 - 2}{2x_n} = \frac{x_n^2 + 2}{2x_n}.$$

We starten in  $x_0 = 2$  en we vinden m.b.v. het stukje Python-code in Figuur 6.14 de volgende uitvoer:

```

1.5000000
1.4166667
1.4142157
1.4142136
1.4142136

```

### 6.7.1 Oefeningen

1. Schrijf een Python-methode `bepaal_vierkantswortel` om de vierkantswortel uit een willekeurig getal  $a$  te vinden tot op een bepaald aantal decimale cijfers nauwkeurig.
2. Beschouw de vergelijking  $2x^2 + 5 = e^x$ .
  - a) Gebruik Python om vast te stellen dat er een oplossing van deze vergelijking ligt in het interval  $[3, 4]$ .
  - b) Gebruik de methode van Newton-Raphson om deze oplossing te benaderen tot op 6 decimale cijfers nauwkeurig.
3. Vind een benadering voor alle nulpunten van de functie  $f(x) = x^3 - x^2 - 15x + 1$ .



- a) Gebruik Python om een plot te maken van de grafiek van  $f$ .
- b) Identificeer geschikte startwaarden voor de methode van Newton-Raphson voor elk van de nulpunten en voer de methode uit tot je een benadering voor het nulpunt krijgt tot 6 decimalen nauwkeurig.

# Reële functies in meerdere veranderlijken

## 7.1 Definitie en voorstelling functies

In het deel over lineaire algebra hebben we al functies gezien van  $\mathbb{R}^m$  naar  $\mathbb{R}^n$ . Inderdaad, elke matrix  $\mathbf{A} \in \mathbb{R}^{n \times m}$  beeldt een kolomvector  $\mathbf{x} \in \mathbb{R}^m$  af op een kolomvector  $\mathbf{Ax} \in \mathbb{R}^n$ . In dat geval zijn de functies echter beperkt tot lineaire transformaties. In dit hoofdstuk bekijken we meer algemene functies in meerdere veranderlijken.

In het algemeen is een REËLE FUNCTIE IN MEERDERE VERANDERLIJKEN een afbeelding die met de tupels in haar domein een tupel (of een reëel getal) associeert:

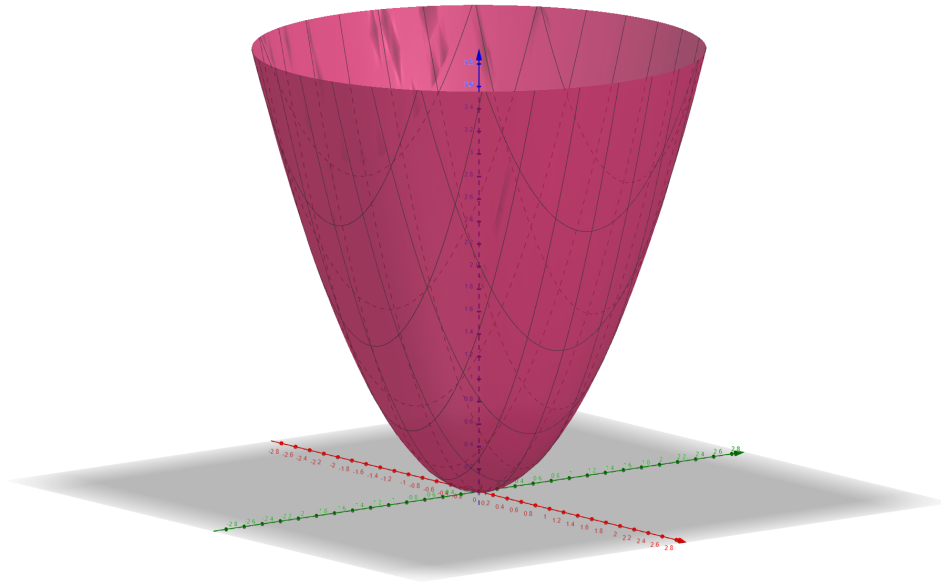
$$f: \mathbb{R}^m \rightarrow \mathbb{R}^n: (x_1, \dots, x_m) \mapsto (f_1(x_1, \dots, x_m), f_2(x_1, \dots, x_m), \dots, f_n(x_1, \dots, x_m)).$$

Hierbij is elke  $f_i$  met  $i \in \{1, 2, \dots, n\}$  een functie die een tupel afbeeldt op een reëel getal. De functie  $f$  zelf bestaat dus uit  $n$  componenten.

In dit hoofdstuk zullen we heel vaak functies van  $\mathbb{R}^m$  naar  $\mathbb{R}$  beschouwen, i.e. functies die een reëel getal als uitvoer hebben.

**Voorbeeld 7.1** We bekijken een functie in twee veranderlijken met een reëelwaardige uitvoer:

$$f: \mathbb{R}^2 \rightarrow \mathbb{R}: (x, y) \mapsto f(x, y) = x^2 + y^2. \quad \blacksquare$$



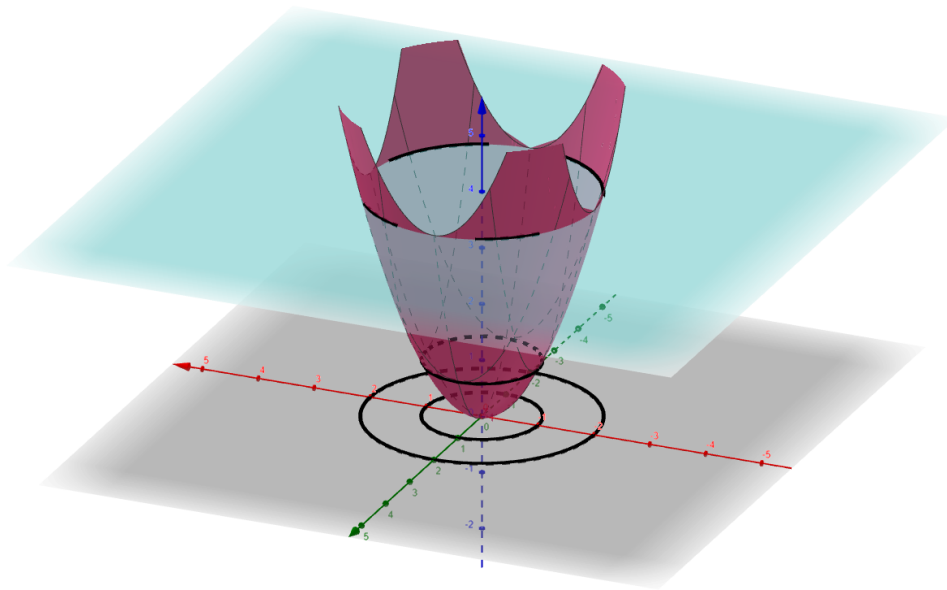
**Figuur 7.1:** Grafiek van functie  $f(x, y) = x^2 + y^2$ .

Visualiseren van functies in meerdere veranderlijken is heel wat ingewikkelder dan het visualiseren van reële functies in één veranderlijke. Dat komt natuurlijk omdat papier tweedimensionaal is, én omdat de meeste menselijke hersens het zeer moeilijk hebben om zich ruimtes in 4 of meer dimensies voor te stellen.

Wanneer we te maken hebben met functies met twee argumenten en één reële uitvoer, dan kunnen we toch nog trachten een visuele voorstelling te maken. We beelden ons in dat het XY-vlak de inputwaarden bevat en dat er een verticale as is die er loodrecht uitsteekt, de Z-as, die gebruikt wordt om de functiewaarden voor te stellen. Figuur 7.1 toont een voorstelling van de functie  $f(x, y) = x^2 + y^2$  in de buurt van de oorsprong.

Zo'n driedimensionale voorstelling is nuttig en interessant maar zeker niet gemakkelijk om te creëren en al helemaal niet zonder speciale software.

Een andere manier om functies met twee reële invoerwaarden en een reële uitvoerwaarde voor te stellen is aan de hand van een CONTOURPLOT. Om een contourplot te creëren moet je je voorstellen dat je de originele driedimensionale grafiek snijdt met een vlak  $z = C$  evenwijdig met het XY-



**Figuur 7.2:** De functie  $f(x, y) = x^2 + y^2$ , met de niveau curves voor  $C = 1$  en  $C = 4$ . Ter illustratie is ook het vlak  $z = 4$  getekend. Dit geeft aanleiding tot een cirkel met straal 2 als niveau kromme.

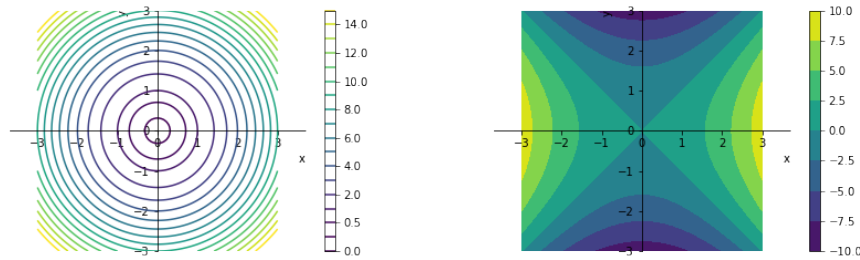
vlak. In het algemeen krijg je dan een lijn in het vlak  $z = C$  die alle paren  $(x, y)$  voorstelt waarvoor  $f(x, y) = C$ . Deze lijn kunnen we ook tekenen in het originele  $XY$ -vlak; zo'n lijn noemen we een NIVEAU CURVE of NIVEAU KROMME (Eng. *level curve*).

**Voorbeeld 7.2** De niveau curves voor de functie  $f(x, y) = x^2 + y^2$  zijn van de vorm:

$$x^2 + y^2 = C.$$

Als  $C < 0$  dan is de niveau curve leeg, er zijn immers geen reële waarden voor  $x$  en  $y$  waarvoor  $x^2 + y^2 < 0$ . Als  $C \geq 0$ , dan is de niveau curve voor deze functie een cirkel voor met straal  $\sqrt{C}$  en middelpunt in de oorsprong. In Figuur 7.2 zie je een illustratie. ■

We kunnen ook enkel naar de niveau krommes in het  $XY$ -vlak kijken en die eventueel inkleuren volgens de hoogte. Op die manier krijgen we een beeld



**Figuur 7.3:** Links een niet-ingevulde contour plot van  $f(x, y) = x^2 + y^2$ . Rechts een ingevulde contour plot van het zadeloppervlak  $f(x, y) = x^2 - y^2$ .

van de grafiek van de functie, net zoals we op een landkaart informatie m.b.t. het reliëf kunnen afleiden. In Figuur 7.3 zie je een tweetal illustraties.

## 7.2 Partiële afgeleiden

Concepten zoals limieten en continuïteit kunnen ook worden gedefinieerd voor functies in meerdere veranderlijken, maar we focussen ons nu vooral op het begrip “afgeleide” in meerdere dimensies.

Net als bij afgeleiden bij reële functies in één veranderlijke geven “afgeleiden” aan hoe de functie wijzigt wanneer men zich een heel klein beetje verplaatst vanaf het huidige punt. Het grote verschil is echter dit: bij een functie in één veranderlijke is er slechts één richting waarin men kan bewegen, nl. naar links en rechts over de  $X$ -as. Wanneer we echter bv. werken met een functie in twee veranderlijken (en één reële uitvoer) dan kunnen we in principe in elke richting in het  $XY$ -vlak bewegen en we kunnen ons voorstellen dat de helling die we zullen ondergaan afhankelijk is van de richting waarin we bewegen.

Om eenvoudig te starten kunnen we een richting kiezen die evenwijdig is met één van de assen, bv. evenwijdig met de  $X$ -as. Dit betekent dat we de  $x$ -inputwaarde kunnen laten variëren, maar dat *de  $y$ -waarde vast blijft*. Stel dat we dit doen in het punt  $(a, b)$ , dan beschouwen we de volgende verhouding:

$$\frac{f(a + h, b) - f(a, b)}{h},$$

en meer in het bijzonder beschouwen we opnieuw de limiet waarbij  $h$  naar nul gaat:

$$\lim_{h \rightarrow 0} \frac{f(a+h, b) - f(a, b)}{h}.$$

Als deze limiet bestaat dan noemen we deze waarde de PARTIËLE AFGELEIDE NAAR  $x$  van  $f$  in het punt  $(a, b)$ . We gebruiken hiervoor de volgende notatie:

$$\frac{\partial f}{\partial x}(a, b) = \lim_{h \rightarrow 0} \frac{f(a+h, b) - f(a, b)}{h}.$$

We kunnen uiteraard eveneens bewegen evenwijdig met de  $Y$ -as, d.w.z. dat we de  $x$ -waarde vast laten maar dat de  $y$ -waarde varieert. De partiële afgeleide naar  $y$  in het punt  $(a, b)$  is dan de limiet

$$\frac{\partial f}{\partial y}(a, b) = \lim_{h \rightarrow 0} \frac{f(a, b+h) - f(a, b)}{h},$$

op voorwaarde dat deze limiet bestaat.

**Opmerking 7.3 (Notatie  $\partial$  vs.  $d$ )** De notatie met de “gekrulde  $d$ ” nl.  $\partial$  is diegene die wordt gebruikt om een *partiële* afgeleide aan te duiden. Daarmee maken we duidelijk dat er nog andere argumenten zijn waardoor de functie kan wijzigen.

In Hoofdstuk 6 hebben we notatie

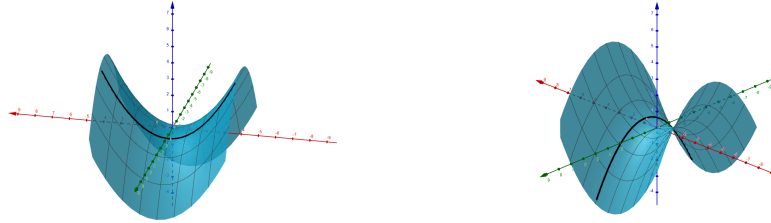
$$f'(x)$$

gebruikt om een afgeleide aan te geven. We zullen later in dit hoofdstuk ook de notatie

$$\frac{df}{dx}$$

gebruiken om dezelfde afgeleide weer te geven, waarbij een “gewone  $d$ ” wordt gebruikt. ■

We bekijken de functie  $f(x, y) = (x^2 - y^2)/4$ . We wensen de partiële afge-



**Figuur 7.4:** Een zadeloppervlak  $f(x, y) = (x^2 - y^2)/4$ . In de linkerfiguur is de snijlijn aangeduid met het vlak  $y = 1$ , zodat men evenwijdig met de  $X$ -as beweegt. In de rechterfiguur (met een andere oriëntatie) ziet men de snijlijn met het vlak  $x = 1$ .

leide naar  $x$  te berekenen in  $(x, y) = (1, 1)$ . We bepalen de limiet:

$$\begin{aligned}
 \frac{\partial f}{\partial x}(1, 1) &= \lim_{h \rightarrow 0} \frac{f(1+h, 1) - f(1, 1)}{h} \\
 &= \lim_{h \rightarrow 0} \frac{((1+h)^2 - 1^2)/4 - (1^2 - 1^2)/4}{h} \\
 &= \lim_{h \rightarrow 0} \frac{h^2 + 2h}{4h} \\
 &= \lim_{h \rightarrow 0} \frac{h}{4} + \frac{1}{2} \\
 &= \frac{1}{2}.
 \end{aligned}$$

De raaklijn aan de curve in het driedimensionale punt  $(1, 1, 0)$  in de  $X$ -richting is m.a.w. gelijk aan  $1/2$ . Als  $x$  stijgt (en  $y$  blijft gelijk), dan stijgt ook de functiewaarde.

Op dezelfde manier kan je vinden dat

$$\frac{\partial f}{\partial y}(1, 1) = -\frac{1}{2}.$$

Dit kan je ook zien op Figuur 7.4: op de linkerfiguur zie je dat de zwarte kromme stijgt boven het punt  $(1, 1)$ , terwijl dat voor de zwarte kromme op de rechterfiguur niet het geval is.

Laat ons nog eens een andere functie beschouwen die duidelijk kan maken hoe men partiële afgeleiden eenvoudig kan berekenen met de methodes die

we reeds kennen voor het afleiden van reële functies in één veranderlijke. Beschouw de functie

$$f(x, y) = x^2y + xy^2.$$

We bepalen de partiële afgeleide naar  $x$  in een willekeurig punt  $(a, b)$ :

$$\begin{aligned} \frac{\partial f}{\partial x}(a, b) &= \lim_{h \rightarrow 0} \frac{f(a+h, b) - f(a, b)}{h} \\ &= \lim_{h \rightarrow 0} \frac{(a+h)^2b + (a+h)b^2 - (a^2b + ab^2)}{h} \\ &= \lim_{h \rightarrow 0} \frac{(a+h)^2b - a^2b}{h} + \lim_{h \rightarrow 0} \frac{(a+h)b^2 - ab^2}{h} \\ &= 2ab + b^2. \end{aligned}$$

Hier kunnen we nu een belangrijke observatie maken: om de waarde van de partiële afgeleide naar  $x$  te bepalen van een functie, leiden we deze functie eenvoudigweg af naar  $x$  waarbij we veronderstellen dat alle andere variabelen een constante waarde aannemen!

**Eigenschap 7.4** Stel dat  $f$  een reële functie is van  $\mathbb{R}^n$  naar  $\mathbb{R}$ : de partiële afgeleide van  $f$  naar de variabele  $x_i$  kan berekend worden als de afgeleide van een functie  $g(x_i)$  die hetzelfde functievoorschrift heeft als  $f$  maar waarbij alle andere variabelen (verschillend van  $x_i$ ) geïnterpreteerd worden als constanten. ■

### 7.2.1 Oefeningen

1. Bereken de partiële afgeleiden van de volgende functies:

- a)  $f(x, y) = \cos(x^2 + 2y)$
- b)  $f(x, y) = \exp(x^2 + y^2)$
- c)  $f(s, t, v) = t^2 \ln(s + 2t) - \ln(3v)(s^3 + t^2 - 4v)$
- d)  $f(x, y, z) = \exp(-z)(x^2y + 2)$
- e)  $f(x, y, z) = \frac{\exp(z)}{\exp(x) + \exp(y) + \exp(z)}.$

## 7.3 De gradiënt

We hebben gezien hoe men voor een functie van  $\mathbb{R}^n$  naar  $\mathbb{R}$  in elk punt  $n$  partiële afgeleiden kan bepalen die aangeven hoe de functie varieert wan-



neer men een kleine beweging maakt in de richting van één van de assen van het assenstelsel.

Wanneer men al deze partiële afgeleiden verzamelt in een kolomvector, dan noemen we deze vector de **GRADIËNT** van de functie in dat punt. Deze gradiënt wordt dikwijls aangeduid met Griekse symbool “nabla”, ofte  $\nabla$ .

**Voorbeeld 7.5** Voor de functie  $f(x, y) = x^2 + y^2$  wordt de gradiënt in het punt  $(1, 2)$  gegeven door

$$\nabla f(1, 2) = \begin{bmatrix} \frac{\partial f}{\partial x}(1, 2) \\ \frac{\partial f}{\partial y}(1, 2) \end{bmatrix} = \begin{bmatrix} 2 \\ 4 \end{bmatrix}.$$

Hierbij hebben we gebruikgemaakt van het feit dat

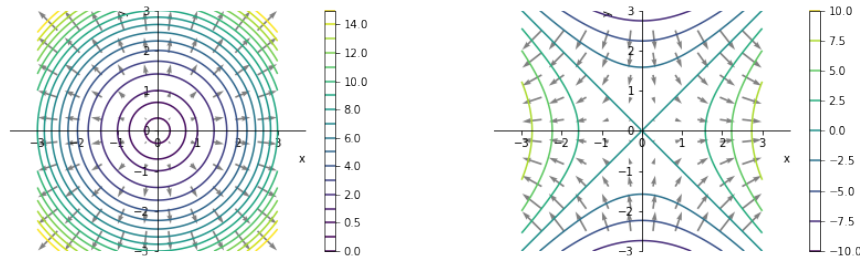
$$\frac{\partial f}{\partial x}(x, y) = 2x \quad \text{en} \quad \frac{\partial f}{\partial y}(x, y) = 2y. \quad \blacksquare$$

De gradiënt in elk punt is een vector en kunnen we dus voorstellen als een pijltje in het  $XY$ -vlak. We kunnen deze vectoren tekenen voor een aantal verschillende punten, samen met de contourplot van de functie. In Figuur 7.5 doen we dit voor twee verschillende functies. We merken de volgende zeer belangrijke zaken op:

1. De pijltjes van de gradiënt staan steeds loodrecht op de niveau krommen.
2. De gradiënt wijst steeds in de richting waarin de functie stijgt.
3. De pijltjes zijn langer, i.e. de norm van de gradiënt is groter, daar waar de functie steiler is.

Deze observaties laten ons geloven dat de volgende belangrijke eigenschap waar is.

**Eigenschap 7.6** Als  $f: \mathbb{R}^n \rightarrow \mathbb{R}$  een functie is zodat de gradiënt van  $f$  bestaat, dan wijst de gradiënt in elk punt *de richting aan waarin de functie  $f$  het sterkst stijgt*. Hoe groter deze stijging, hoe groter de norm van de gradiënt.  $\blacksquare$



**Figuur 7.5:** Contourplots van de functies  $f(x,y) = x^2 + y^2$  (links) en  $f(x,y) = x^2 - y^2$  aangevuld met een aanduiding van de gradiënt in een aantal verschillende punten. Merk op: de pijltjes op de figuur zijn geschaald om de figuur beter leesbaar te maken maar grotere pijlen duiden op het feit dat de gradiënt een grotere lengte heeft.

**Voorbeeld 7.7** We voeren een experiment uit om Eigenschap 7.6 te bevestigen. Voor een functie  $f$  van  $\mathbb{R}^2$  naar  $\mathbb{R}$  kan men de afgeleide in een punt  $\mathbf{a}$  in de richting

$$\begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix} \quad \text{met } \theta \in [0, 2\pi]$$

gaan benaderen door een vector  $\mathbf{h}$  te bepalen met een zeer kleine norm, bv.  $10^{-3}$ , i.e.

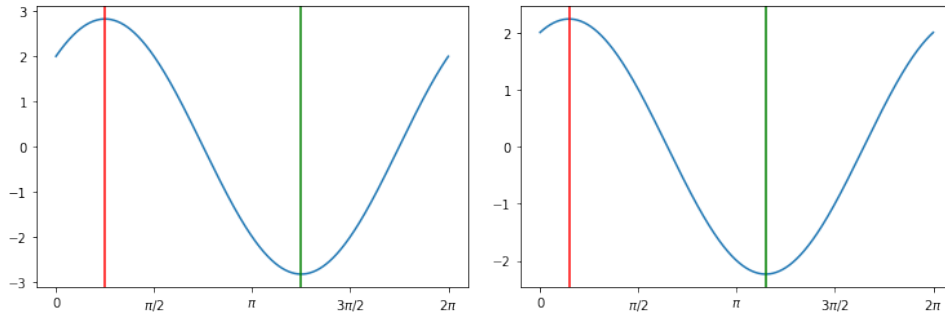
$$\mathbf{h} = 10^{-3} \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \end{bmatrix}.$$

Met deze vector berekenen we dan de volgende verhouding:

$$\frac{f(\mathbf{a} + \mathbf{h}) - f(\mathbf{a})}{\|\mathbf{h}\|}.$$

Deze verhouding geeft een benadering voor de manier waarop de functie  $f$  verandert (i.e. voor de afgeleide) in de gegeven richting. Als men dit doet voor alle richtingen, i.e. voor een groot aantal hoeken tussen 0 en  $2\pi$  radialen, dan kan men een plot maken waarop men visueel kan aflezen in welke richting  $f$  het sterkst toeneemt, afneemt en niet verandert. Figuur 7.6 geeft een illustratie.

We zien inderdaad dat de sterkste stijging wordt bereikt in de richting van de gradiënt. De richting tegengesteld aan die van de gradiënt geeft de sterkste daling. ■



**Figuur 7.6:** Grafieken die tonen hoe afgeleide voor  $f(x, y) = x^2 + y^2$  varieert in verschillende richtingen. De figuur links bekijkt de afgeleide in het punt  $(1, 1)$ . De figuur rechts in  $(1, 1/2)$ . Je ziet dat het maximum bereikt wordt voor verschillende hoeken. De rode lijn geeft de richting weer van de gradiëntvector in dat punt. De groene lijn is dezelfde hoek maar 180 graden ( $\pi$  radialen) verder, en wijst dus in de tegengestelde richting van de gradiënt. Je ziet dat deze richtingen overeenkomen met de richtingen waarin de functie het sterkst stijgt en daalt.

```
import numpy as np
import matplotlib.pyplot as plt

def compute_directional_derivative(f, current):
    thetas = np.linspace(0, 2*np.pi, 720)
    size = 10**-3
    ds = []
    for theta in thetas:
        direction = np.array([np.cos(theta), np.sin(theta)]) * size
        ds.append((f(current + direction) - f(current))/size)

    return thetas, ds

f = lambda x : x[0]**2 + x[1]**2
current = np.array([1, 1])
thetas, derivatives = compute_directional_derivative(f, current)
plt.plot(thetas, derivatives);
```

**Figuur 7.7:** Basiscode om de (linker)grafiek in Figuur 7.6 te reproduceren. Additionele code is nodig om de verticale lijnen te plotten en de labels mooier te maken, maar dit is de essentie van het experiment.

Tot slot van deze sectie nog een opmerking over het belang van de gradiënt. Voor functies in één veranderlijke vinden we het minimum typisch daar waar de afgeleide gelijk is aan nul, al zijn er uitzonderingen mogelijk zoals gezien in Hoofdstuk 6. Als we het minimum willen vinden van een functie in  $n$  veranderlijken maar met een reële uitvoer, dan vertaalt zich dit in het feit dat de gradiënt in dat punt gelijk moet zijn aan nul, op voorwaarde dat de gradiënt bestaat in dat punt.

Let op: het feit dat de gradiënt gelijk is aan nul is geen garantie dat het punt een minimum is. Het kan ook een maximum zijn of een zadelpunt.

## 7.4 Gradient descent

Veel toepassingen in machinaal leren vereisen dat we een bepaalde “kost-functie” zo klein mogelijk maken. Deze kostfunctie meet typisch hoe goed ons model het doet, en kleinere waarden wijzen op een beter model<sup>1</sup>. De “Mean Squared Error” is een typisch voorbeeld van zo’n kostfunctie en die hebben we reeds gezien in de context van lineaire regressie. We komen later op dit voorbeeld terug, maar starten met de beschrijving van een algemeen algoritme dat in het Engels GRADIENT DESCENT wordt genoemd en dat je kan gebruiken om, hopelijk, een minimum van een (kost)functie te vinden.

Intuïtief wordt gradient descent soms als volgt omschreven: je staat ergens op een berglandschap maar het is zeer mistig, dus je kan enkel in je onmiddellijke nabijheid waarnemen in welke richting het landschap stijgt of daalt. Je wil terug naar het dal (dus naar een laag punt). Je zoekt in elke stap de richting van de sterkste daling, en dan zet je een zeer klein pasje in die richting. Dit blijf je herhalen totdat alles rondom jou vlak is. Op dat moment ben je (hopelijk) in het dal aangekomen.

De manier waarop precies tewerk wordt gegaan in gradient descent is als volgt. Er is een functie  $f: \mathbb{R}^n \rightarrow \mathbb{R}$  gegeven waarvan we veronderstellen dat de gradiënt overal bestaat en gegeven wordt door  $\nabla f$ . We starten in een initieel punt  $\mathbf{x}_0 \in \mathbb{R}^n$  en we zullen deze positie iteratief verbeteren door gebruik te maken van de gradiënt in het huidige punt. Meer in het bijzonder voeren we de volgende update uit:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha \nabla f(\mathbf{x}_n).$$

<sup>1</sup>Het belangrijke probleem van overfitting zal in een latere cursus in de opleiding besproken worden.

Hierbij is  $\alpha$  een parameter die je zelf kan kiezen en die de STAPGROOTTE (Eng. *step size* of ook *learning rate*) wordt genoemd. Deze keuze van deze parameter is in de praktijk zeer belangrijk. Merk op dat het in principe niet nodig is om de voorgaande posities te bewaren. In dit geval kan je de update regel ook schrijven als:

$$\mathbf{x} \leftarrow \mathbf{x} - \alpha \nabla f(\mathbf{x}). \quad (7.1)$$

**Voorbeeld 7.8** Beschouw een functie  $f$  met volgend functievoorschrift:

$$f(x, y) = x^2 + 2y^2.$$

We wensen deze functie te minimaliseren m.b.v. het gradient descent algoritme. Men berekent de gradiënt als volgt:

$$\nabla f(x, y) = \begin{bmatrix} 2x \\ 4y \end{bmatrix}.$$

Veronderstel nu dat we starten in het punt  $(1, 1)$  en dat we als stapgrootte  $\alpha = 0.1$  nemen. In de eerste iteratie van het algoritme evalueert gradient descent de gradiënt in het punt  $(1, 1)$  en bekomt:

$$\nabla f(1, 1) = \begin{bmatrix} 2 \\ 4 \end{bmatrix}.$$

De volgende aanpassing gebeurt:

$$\mathbf{x} \leftarrow \begin{bmatrix} 1 \\ 1 \end{bmatrix} - 0.1 \begin{bmatrix} 2 \\ 4 \end{bmatrix} = \begin{bmatrix} 0.8 \\ 0.6 \end{bmatrix}.$$

Het nieuwe huidige punt is dus  $(0.8, 0.6)$ . We berekenen de gradiënt  $\nabla f$  in dit punt:

$$\nabla f(0.8, 0.6) = \begin{bmatrix} 1.6 \\ 2.4 \end{bmatrix}.$$

We passen de update regel (7.1) toe en we vinden:

$$\mathbf{x} \leftarrow \begin{bmatrix} 0.8 \\ 0.6 \end{bmatrix} - 0.1 \begin{bmatrix} 1.6 \\ 2.4 \end{bmatrix} = \begin{bmatrix} 0.64 \\ 0.36 \end{bmatrix}.$$

De volgende iteraties worden samengevat in Tabel 7.1 waarbij we voor de eenvoud van notatie de coördinaten van de punten apart hebben genoteerd.

$i$	$x$	$x$	$f(x, y)$
1	1	1	3
2	0.8	0.6	1.36
3	0.64	0.36	0.6688
4	0.512	0.216	0.3555
5	0.4096	0.1296	0.2014
6	0.3277	0.0778	0.1195
7	0.2621	0.0467	0.0731
8	0.2097	0.0280	0.0455
9	0.1678	0.0168	0.0287
10	0.1342	0.0101	0.0182

**Tabel 7.1:** Tien iteraties van gradient descent voor de functie  $f(x, y) = x^2 + 2y^2$ , startend in  $(1, 1)$  en met stapgrootte  $\alpha = 0.1$ .

Men ziet dat men na 10 iteraties al relatief dicht genaderd is bij het punt  $(0, 0)$  waarvan men (voor deze eenvoudige functie) op het zicht ziet dat dit het globale minimum is.

Deze resultaten worden bevestigd door de eenvoudige Python-code en de uitvoer ervan in Figuur 7.8. ■

**Voorbeeld 7.9 (Lokale minima)** In Figuur 7.9 zie je drie toepassingen van gradient descent voor een functie met twee minima. Het globale minimum bevindt zich in  $(0, 0)$ , het lokale minimum in  $(2, 2)$ . Afhankelijk van de startpositie eindigt gradient descent in één van deze twee minima. Dit voorbeeld toont aan dat gradient descent niet gegarandeerd eindigt in een globaal minimum! ■

**Voorbeeld 7.10 (Gradient descent met afgeplatte ellips)** Om aan te geven dat gradient descent niet steeds in de “globale” juiste richting stapt geven we een voorbeeld waarbij we de volgende kostfunctie wensen te minimaliseren:

$$f(x, y) = x^2 + \frac{y^2}{b},$$

waarbij  $b$  een strikt positieve parameter is die we kunnen laten variëren. Merk dat als  $b = 1$  de niveau krommen van deze functie cirkels zijn, ter-

```
import numpy as np

f = lambda x : x[0]*x[0] + 2*x[1]*x[1]
g = lambda x : np.array([2*x[0], 4*x[1]])
alpha = 0.1
curr = np.array([1,1])
for _ in range(10):
    print(f"{curr[0]:.5f}, {curr[1]:.5f}, {f(curr):.5f}")
    curr = curr - alpha * g(curr)
```

Deze code geeft de volgende uitvoer:

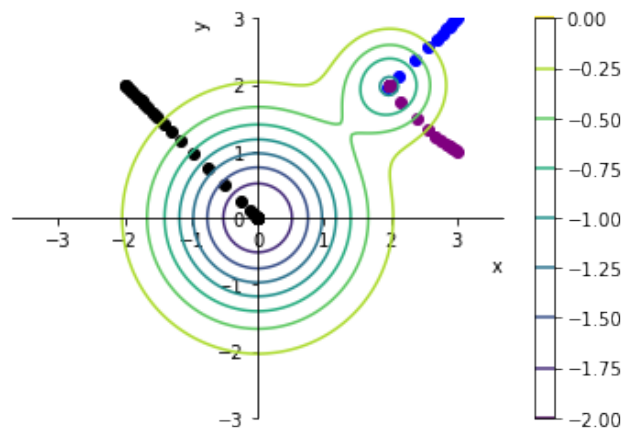
```
1.00000, 1.00000, 3.00000
0.80000, 0.60000, 1.36000
0.64000, 0.36000, 0.66880
0.51200, 0.21600, 0.35546
0.40960, 0.12960, 0.20136
0.32768, 0.07776, 0.11947
0.26214, 0.04666, 0.07307
0.20972, 0.02799, 0.04555
0.16777, 0.01680, 0.02871
0.13422, 0.01008, 0.01822
```

**Figuur 7.8:** Eenvoudige Python code voor gradient descent toegepast op de functie  $f(x, y) = x^2 + 2y^2$ , samen met de uitvoer van deze code.

wijl de niveau krommen voor  $b > 1$  ellipsen zijn. Hoe groter  $b$  wordt, hoe meer “uitgerekt” de ellips wordt in de  $Y$ -richting. Zie Figuur 7.10 voor een illustratie.

We doen een experiment waarbij het gradient descent uitvoeren startend in het punt  $(2, 2)$  met stapgrootte  $\alpha = 1/10$ . We voeren voldoende iteraties uit totdat het huidige punt een norm heeft die kleiner is dan  $10^{-6}$  (omdat we weten dat het enige minimum zich bevindt in de oorsprong).

Het aantal benodigde iteraties dat hiervoor nodig was zie je in de onderstaande tabel:



**Figuur 7.9:** Een contourplot van een functie met twee minima. Er worden drie toepassingen van gradient descent gevisualiseerd. Twee eindigen in een lokaal minimum, enkel de toepassing van gradient descent aangegeven door de zwarte posities eindigt in het globale minimum.

$b$	aantal iteraties
1	68
10	720
100	7249
1000	72538

Je ziet duidelijk dat het aantal nodige iteraties sterk stijgt naarmate  $b$  stijgt.

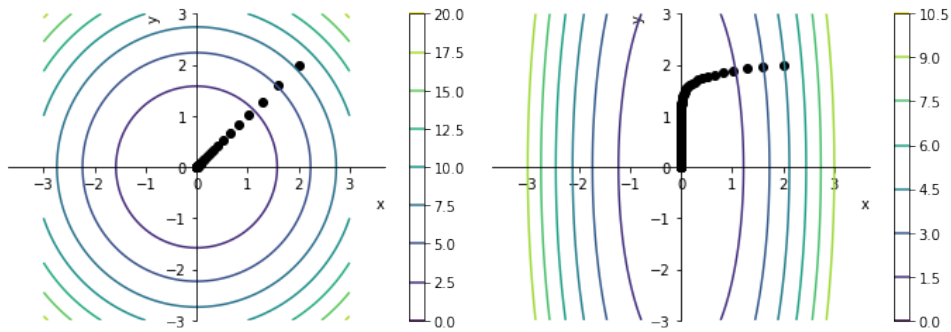
Wanneer de attributen bij machinaal leren zeer verschillend zijn dan heeft dit een effect gelijkaardig aan een grote waarde van  $b$ . Zoals je later ook zal zien<sup>2</sup> is het daarom zeer belangrijk dat de *attributen een gelijkaardige schaal hebben* om gradient descent efficiënt te laten werken. ■

### 7.4.1 Lineaire regressie met gradient descent

We keren terug naar een probleem in machinaal leren dat we ook al hebben aangehaald in de context van lineaire algebra. De oplossingsmethode die we in deze sectie zullen aanrijken ligt echter al heel wat dichterbij de manier waarop bv. problemen in deep learning, i.e. met grote neurale netwerken worden aangepakt.

<sup>2</sup>In het opleidingsonderdeel Machine Learning





**Figuur 7.10:** Gradient descent toegepast op  $x^2 + y^2$  (links) en  $x^2 + y^2/10$  (rechts). Je ziet dat bij de cirkel de aanpassingen steeds gebeuren in de richting van het globale minimum (de oorsprong), terwijl dat voor de ellips niet het geval is. De eerste aanpassingen gaan daar niet de goede richting uit. Bijgevolg heeft gradient descent veel meer iteraties nodig om in de buurt van de oorsprong te komen.

Gegeven een dataset bestaande uit  $m$  gelabelde voorbeelden

$$(\mathbf{x}^{(i)}, y^{(i)}) \quad \text{met} \quad i \in \{1, 2, \dots, m\}$$

waarbij elke  $\mathbf{x}^{(i)}$  een  $n$ -dimensionale (kolom)vector is en elk label  $y^{(i)}$  een reëel getal is. Anders gezegd

$$\mathbf{x}^{(i)} = \begin{bmatrix} \mathbf{x}_1^{(i)} \\ \mathbf{x}_2^{(i)} \\ \vdots \\ \mathbf{x}_n^{(i)} \end{bmatrix} \in \mathbb{R}^n \quad \text{en} \quad y^{(i)} \in \mathbb{R}.$$

De bedoeling is om voor een nieuwe vector  $\mathbf{x} \in \mathbb{R}^n$  te voorspellen wat de bijhorende waarde voor  $y$  is. We hebben hier m.a.w. te maken met een *regressieprobleem*.

De voorspellingen die we willen maken hangen af van een aantal *parameters* nl. een vector  $\mathbf{w} \in \mathbb{R}^n$  en een reëel getal  $b$ . Het zijn deze parameters die door het algoritme moeten worden gevonden.

De manier waarop een voorspelling wordt gemaakt voor een vector  $\mathbf{x}$  wanneer de parameters  $\mathbf{w}$  en  $b$  gekend zijn is:

$$\begin{aligned} h_{\mathbf{w},b}(\mathbf{x}) &= \mathbf{w} \cdot \mathbf{x} + b \\ &= w_1 x_1 + w_2 x_2 + \dots + w_n x_n + b. \end{aligned}$$

Zo'n voorspelling noteren we ook kortweg als  $\hat{y}$ .

Het idee achter lineaire regressie is heel eenvoudig. Met elke mogelijke combinaties van parameters  $\mathbf{w}$  en  $b$  associëren we een *kost* (gebaseerd op de beschikbare voorbeelden). We zoeken dié parameters  $\mathbf{w}$  en  $b$  die de kost (voor de gegeven dataset) minimaliseren. De kost is, zoals ook reeds vroeger gezien, (in essentie) niets anders dan de *gemiddelde kwadratische afwijking* tussen de voorspelde labels  $\hat{y}^{(i)}$  en de werkelijke labels  $y^{(i)}$ . In formulevorm vinden we

$$\begin{aligned} J(\mathbf{w}, b) &= \frac{1}{2m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)})^2 \\ &= \frac{1}{2m} \sum_{i=1}^m (\mathbf{w} \cdot \mathbf{x}^{(i)} + b - y^{(i)})^2 \\ &= \frac{1}{2m} \sum_{i=1}^m (w_1 x_1^{(i)} + w_2 x_2^{(i)} + \cdots + w_n x_n^{(i)} + b - y^{(i)})^2. \end{aligned} \quad (7.2)$$

**Opmerking 7.11** Het is zeer belangrijk om in te zien dat de functie  $J$  een functie is van de parameters  $\mathbf{w}$  en  $b$ . Hoewel we traditioneel de letter  $x$  gebruiken om de onbekenden voor te stellen, zijn de  $x$ -en en  $y$ -s in dit geval constanten en zijn de variabelen waarvoor we moeten optimaliseren  $\mathbf{w}$  en  $b$ . ■

**Opmerking 7.12** In formule (7.2) is een factor  $\frac{1}{2}$  toegevoegd. Dit zorgt ervoor dat straks de gradiënt geen factor 2 heeft. Merk op dat het minimumpunt van  $J$  precies gelijk is aan het minimumpunt van de MSE. ■

### Analytische bepaling gradiënt

De oplossingsmethode die we voorop stellen om  $J$  te minimaliseren is gradient descent. We starten bijgevolg met het bepalen van de afgeleiden, meer bepaald van alle partiële afgeleiden (in analytische vorm):

$$\frac{\partial}{\partial w_j} J(\mathbf{w}, b) = \frac{1}{m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)}) x_j^{(i)} \quad \text{met } j \in \{1, \dots, n\} \quad (7.3)$$

en

$$\frac{\partial}{\partial b} J(w, b) = \frac{1}{m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)}).$$

Als men dit op een voor de hand liggende manier zou gaan berekenen dan schrijft men een expliciete lus over alle voorbeelden en telt men de individuele bijdragen op. Veel moderne programmeertalen beschikken echter over

methodes die zeer snel matrixvermenigvuldigingen kunnen uitvoeren. We trachten hiervan gebruik te maken samen met de kennis van lineaire algebra die we reeds hebben uit het eerste deel van de cursus. Zoals voorheen definiëren de datamatrix  $\mathbf{X}$  (ditmaal zonder kolom van 1-en)

$$\mathbf{X} = \begin{bmatrix} \text{---} & (\mathbf{x}^{(1)})^T & \text{---} \\ \text{---} & (\mathbf{x}^{(2)})^T & \text{---} \\ & \vdots & \\ \text{---} & (\mathbf{x}^{(m)})^T & \text{---} \end{bmatrix} \in \mathbb{R}^{m \times n},$$

en twee vectoren  $\mathbf{y}$  en  $\hat{\mathbf{y}}$ :

$$\mathbf{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \quad \text{en} \quad \hat{\mathbf{y}} = \begin{bmatrix} \hat{y}^{(1)} \\ \hat{y}^{(2)} \\ \vdots \\ \hat{y}^{(m)} \end{bmatrix},$$

De gradiënt  $\nabla_{\mathbf{w}} J \in \mathbb{R}^n$  is, zoals reeds gezien, de kolomvector die alle partiële afgeleiden verzamelt.

Als we kijken naar formule (7.3), dan herkennen we daarin duidelijk het verschil tussen  $\hat{\mathbf{y}}$  en  $\mathbf{y}$ . Anderzijds speelt ook de  $j$ -de kolom van  $\mathbf{X}$  een rol. Echter, de  $j$ -de kolom van  $\mathbf{X}$  is de  $j$ -de rij van  $\mathbf{X}^T$ . Bovendien kunnen  $\mathbf{X}^T \in \mathbb{R}^{n \times m}$  en  $\mathbf{y} \in \mathbb{R}^m$  met elkaar vermenigvuldigd worden en is het resultaat een kolomvector in  $\mathbb{R}^n$ , dat is precies de dimensie van  $\nabla_{\mathbf{w}} J$ . Je kan inderdaad verifiëren dat

$$\nabla_{\mathbf{w}} J = \frac{1}{m} \mathbf{X}^T (\hat{\mathbf{y}} - \mathbf{y}) \quad (7.4)$$

Voor de partiële afgeleide van de kost naar  $b$  vindt men op gelijkaardige manier dat

$$\frac{\partial J}{\partial b} = \frac{1}{m} \mathbf{1}_{1 \times m} (\hat{\mathbf{y}} - \mathbf{y}),$$

waarbij  $\mathbf{1}$  een rijmatrix is volledig gevuld met 1-en. Dit is een ingewikkelde manier om uit te drukken dat men de som moet nemen van alle waarden in het verschil  $\hat{\mathbf{y}} - \mathbf{y}$  (en dat men daarna nog moet delen door  $m$  om het gemiddelde te bepalen).

**Voorbeeld 7.13 (Voorbeeld berekening gradiënt)** We bekijken een dataset bestaande uit 5 datapunten:

$i$	$x^{(i)}$	$y^{(i)}$
1	1.0	1.0
2	2.0	4.5
3	3.0	9.5
4	4.0	15.0
5	5.0	24.0

In dit geval willen we echter de *best passende parabool* vinden, m.a.w. we doen voorspellingen van de vorm:

$$\hat{y} = w_1 x + w_2 x^2 + b.$$

Veronderstel dat  $w_1 = 1$ ,  $w_2 = 0.5$  en  $b = -0.4$ , dan is

$i$	$x_1^{(i)} = x^{(i)}$	$x_2^{(i)} = (x^{(i)})^2$	$y^{(i)}$	$\hat{y}^{(i)} = -0.4 + x^{(i)} + 0.5(x^{(i)})^2$	$\hat{y}^{(i)} - y^{(i)}$
1	1.0	1.0	1.0	$-0.4 + 1.0 + 0.5 \times 1.0 = 1.1$	0.1
2	2.0	4.0	4.5	$-0.4 + 2.0 + 0.5 \times 4.0 = 3.6$	-0.9
3	3.0	9.0	9.5	$-0.4 + 3.0 + 0.5 \times 9.0 = 7.1$	-2.4
4	4.0	16.0	15.0	$-0.4 + 4.0 + 0.5 \times 16.0 = 11.6$	-3.4
5	5.0	25.0	24.0	$-0.4 + 5.0 + 0.5 \times 25.0 = 17.1$	-6.9

de partiële afgeleide van  $J$  naar  $w_1$  volgens formule (7.3) gelijk aan:

$$\begin{aligned} \frac{\partial}{\partial w_1} J(\mathbf{w}, b) &= \frac{1}{m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)}) x_1^{(i)} \\ &= \frac{1}{5} (0.1 \times 1.0 - 0.9 \times 2.0 - 2.4 \times 3.0 - 3.4 \times 4.0 - 6.9 \times 5.0) \\ &= -11.4. \end{aligned}$$

Voor de partiële afgeleide van  $J$  naar  $w_2$  vinden we op dezelfde manier

$$\begin{aligned} \frac{\partial}{\partial w_2} J(\mathbf{w}, b) &= \frac{1}{m} \sum_{i=1}^m (\hat{y}^{(i)} - y^{(i)}) x_2^{(i)} \\ &= \frac{1}{5} (0.1 \times 1.0 - 0.9 \times 4.0 - 2.4 \times 9.0 - 3.4 \times 16.0 - 6.9 \times 25.0) \\ &= -50.4. \end{aligned}$$

Als we de partiële afgeleiden berekenen m.b.v. (7.4) dan vinden we:

$$\mathbf{X} = \begin{bmatrix} 1.0 & 1.0 \\ 2.0 & 4.0 \\ 3.0 & 9.0 \\ 4.0 & 16.0 \\ 5.0 & 25.0 \end{bmatrix} \quad \text{en} \quad \hat{\mathbf{y}} - \mathbf{y} = \begin{bmatrix} 0.1 \\ -0.9 \\ -2.4 \\ -3.4 \\ -6.9 \end{bmatrix}$$

waaruit volgt dat

$$\frac{1}{m} \mathbf{X}^T (\hat{\mathbf{y}} - \mathbf{y}) = \frac{1}{5} \begin{bmatrix} 1.0 & 2.0 & 3.0 & 4.0 & 5.0 \\ 1.0 & 4.0 & 9.0 & 16.0 & 25.0 \end{bmatrix} \begin{bmatrix} 0.1 \\ -0.9 \\ -2.4 \\ -3.4 \\ -6.9 \end{bmatrix} = \begin{bmatrix} -11.4 \\ -50.4 \end{bmatrix}.$$

Dit zijn precies dezelfde waarden als diegene die men bekomt door de sommatie (7.3) uit te voeren. In de praktijk zal de manier m.b.v. lineaire algebra echter sneller zijn, zeker wanneer men geoptimaliseerde implementaties gebruikt voor het uitvoeren van het matrixproduct. ■

### 7.4.2 Oefeningen

1. Bekijk opnieuw de functie uit Voorbeeld 7.8.
  - a) Start in het punt  $(1, 1)$  en neem  $\alpha = 0.00001$ . Bereken de eerste twee updates met de hand. Gebruik Python om te bekijken waar men is na 10 updates. Wat na 100 updates? Wat na 1000 updates? En na 10000? *Dit toont aan dat een te kleine waarde voor  $\alpha$  leidt tot een trage convergentie.*
  - b) Start in het punt  $(1, 1)$  en gebruik  $\alpha = 1$ . Bereken de eerste twee updates met de hand. Wat gebeurt er met de waarde van doelfunctie? Stijgt of daalt deze? Gebruik Python of andere software om de volgende tien updates te berekenen. Wat is de waarde van de doelfunctie na 10 updates? *Dit toont aan dat een te grote waarde voor  $\alpha$  kan leiden tot divergentie.*
2. Men kan gradient descent ook gebruiken om het minimum te vinden voor reële functies in één veranderlijke. In dit geval is de gradiënt

uiteraard gelijk aan de “gewone” afgeleide. Pas gradient descent toe op de functie

$$f(x) = (x - 3)(x - 1)(x + 2)(x + 3).$$

Voer hiertoe de volgende stappen uit.

- a) Bepaal de afgeleide van deze functie.
- b) Plot deze functie m.b.v. Python. Hoeveel minima heeft deze functie.
- c) Schrijf een eenvoudige methode om gradient descent toe te passen voor deze functie. Kom je steeds in het globale minimum uit? Wat zijn goede waarden voor  $\alpha$ ?

## 7.5 Kettingregel in meerdere veranderlijken

Veronderstel dat  $f$  een functie is van twee veranderlijken:

$$f(x, y) = x^2 y$$

maar dat elke variabele  $x$  en  $y$  ook nog eens afhangen van een reële veranderlijke  $t$  op de volgende manier:

$$x(t) = \cos(t) \quad \text{en} \quad y(t) = \sin(t).$$

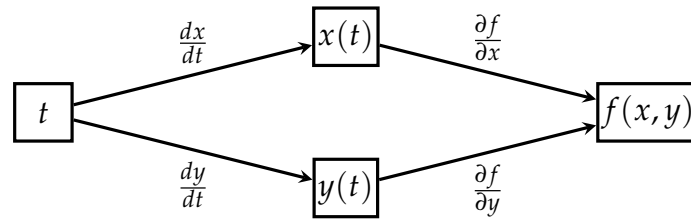
We zoeken nu een manier om de afgeleide van  $f$  naar  $t$  te bepalen zonder noodzakelijk het functievoorschrift (in termen van  $t$ ) expliciet te moeten berekenen. In Figuur 7.11 zie je een graaf die weergeeft hoe deze berekening (in het algemeen) wordt gerealiseerd.

Om te starten bepalen we toch  $f$  expliciet in termen van  $t$  en we vinden:

$$f: \mathbb{R} \rightarrow \mathbb{R}: t \mapsto f(t) = \cos^2(t) \sin(t).$$

Het is nu eenvoudig om de afgeleide van  $f$  naar  $t$  te berekenen m.b.v. de rekenregels voor afgeleiden uit hoofdstuk 6:

$$\begin{aligned} \frac{df}{dt} &= \frac{d \cos^2(t) \sin(t)}{dt} \\ &= \cos^2(t) \frac{d \sin(t)}{dt} + \sin(t) \frac{d \cos^2(t)}{dt} \\ &= \cos^2(t) \cos(t) + 2 \cos(t) (-\sin(t)) \sin(t). \end{aligned}$$



**Figuur 7.11:** Om te berekenen wat de afgeleide is van  $f$  naar  $t$ , bekijk je elk pad van  $t$  naar  $f$  en je vermenigvuldigt alle (partiële) afgeleiden langs dit pad. Tenslotte tel je al deze producten op.

We kunnen dit nu ook als volgt herschrijven:

$$\frac{df}{dt} = \underbrace{2 \cos(t) \sin(t)}_{\frac{\partial f}{\partial x}} \underbrace{(-\sin(t))}_{\frac{dx}{dt}} + \underbrace{\cos^2(t)}_{\frac{\partial f}{\partial y}} \underbrace{\cos(t)}_{\frac{dy}{dt}}.$$

In deze formule zijn nu verschillende onderdelen aangeduid:

$$\frac{\partial f}{\partial x} = 2xy = 2 \cos(t) \sin(t) \quad \text{en} \quad \frac{\partial f}{\partial y} = x^2 = \cos^2(t).$$

Dit suggereert de volgende formule voor  $\frac{df}{dt}$ :

$$\frac{df}{dt} = \frac{\partial f}{\partial x} \frac{dx}{dt} + \frac{\partial f}{\partial y} \frac{dy}{dt}.$$

Dit noemt men de **KETTINGREGEL** voor functies in meerdere veranderlijken.

**Voorbeeld 7.14** We bekijken nog een tweede voorbeeld om deze kettingregel te illustreren. Stel

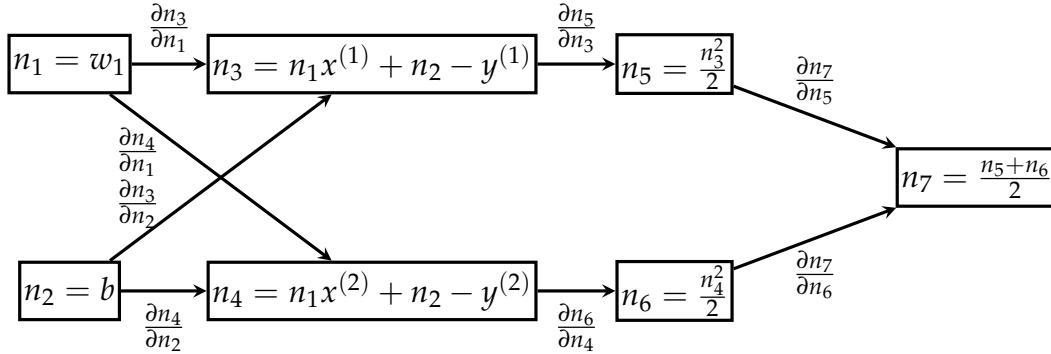
$$f(x, y) = x^2 y - y^2$$

en

$$x(t) = t^2 \quad \text{en} \quad y(t) = 2t.$$

We bepalen de afgeleide van  $f$  naar  $t$  zonder het functievoorschrift van  $f$  in termen van  $t$  expliciet te bepalen. We starten met de partiële afgeleiden van  $f$  naar  $x$  en  $y$ :

$$\frac{\partial f}{\partial x} = 2xy \quad \text{en} \quad \frac{\partial f}{\partial y} = x^2 - 2y.$$



**Figuur 7.12:** Berekenen van  $J = \frac{1}{2} \left( \frac{(w_1 x^{(1)} + b - y^{(1)})^2}{2} + \frac{(w_1 x^{(2)} + b - y^{(2)})^2}{2} \right)$ , de kostfunctie voor lineaire regressie in één veranderlijke  $x$  wanneer het aantal voorbeelden gelijk is aan 2.

Vervolgens bepalen we de afgeleiden van  $x$  en  $y$  naar  $t$ :

$$\frac{dx}{dt} = 2t \quad \text{en} \quad \frac{dy}{dt} = 2.$$

Toepassen van de kettingregel betekent dat we deze resultaten nu twee aan twee vermenigvuldigen en vervolgens de factoren optellen:

$$\begin{aligned} \frac{df}{dt} &= \frac{\partial f}{\partial x} \frac{dx}{dt} + \frac{\partial f}{\partial y} \frac{dy}{dt} \\ &= (2xy)(2t) + (x^2 - 2y)(2) \\ &= (2t^2 2t)(2t) + (t^4 - 4t)(2) \\ &= 8t^4 + 2t^4 - 8t \\ &= 10t^4 - 8t. \end{aligned}$$

Als we, om het resultaat te verifiëren,  $f$  toch expliciet uitschrijven in termen van  $t$  dan vinden we

$$f(t) = (t^2)^2(2t) - (2t)^2 = 2t^5 - 4t^2.$$

Als men deze functie afleidt naar  $t$  dan zie je zo goed als onmiddellijk dat we het gekende resultaat terug bekomen. ■

We bekijken nu het voorbeeld van het berekenen van de gradiënt van de kostfunctie voor lineaire regressie in één veranderlijke in het geval van 2



voorbeelden. In het algemeen werd deze kostfunctie gegeven door vergelijking (7.2) maar in dit bijzonder geval wordt ze:

$$J = \frac{1}{2} \left( \frac{(w_1 x^{(1)} + b - y^{(1)})^2}{2} + \frac{(w_1 x^{(2)} + b - y^{(2)})^2}{2} \right).$$

In Figuur 7.12 staat een mogelijke manier om deze berekening voor te stellen m.b.v. een gerichte graaf.

Om nu de partiële afgeleide van  $J$  m.b.t.  $w_1$  te bepalen moeten we alle paden bekijken van  $n_1$  (i.e. de knoop die  $w_1$  voorstelt) tot aan de knoop  $n_7$  (i.e. de knoop die het finale resultaat voorstelt.) Er zijn twee zo'n paden, namelijk

$$n_1 \rightarrow n_3 \rightarrow n_5 \rightarrow n_7 \quad \text{en} \quad n_1 \rightarrow n_4 \rightarrow n_6 \rightarrow n_7.$$

Langs elk pad moet je alle partiële afgeleiden vermenigvuldigen:

$$\frac{\partial n_3}{\partial n_1} \times \frac{\partial n_5}{\partial n_3} \times \frac{\partial n_7}{\partial n_5} \quad \text{en} \quad \frac{\partial n_4}{\partial n_1} \times \frac{\partial n_6}{\partial n_4} \times \frac{\partial n_7}{\partial n_6}.$$

In concreto wordt dit

$$x^{(1)} \times n_3 \times \frac{1}{2} \quad \text{en} \quad x^{(2)} \times n_4 \times \frac{1}{2}.$$

De volledige partiële afgeleide vind je door deze twee resultaten op te tellen. Als we nu  $n_3$  en  $n_4$  uitdrukken in termen van de originele variabelen  $w_1$  en  $b$  dan vinden we:

$$\begin{aligned} \frac{\partial J}{\partial w_1} &= x^{(1)}(w_1 x^{(1)} + b - y^{(1)}) \frac{1}{2} + x^{(2)}(w_1 x^{(2)} + b - y^{(2)}) \frac{1}{2} \\ &= \frac{1}{2} ((\hat{y}^{(1)} - y^{(1)})x^{(1)} + (\hat{y}^{(2)} - y^{(2)})x^{(2)}) \end{aligned}$$

Dit komt overeen met hetgeen vroeger werd gevonden in formule (7.3).

### 7.5.1 Oefeningen

1. Veronderstel dat  $g: \mathbb{R} \rightarrow \mathbb{R}^n$  en  $h: \mathbb{R}^n \rightarrow \mathbb{R}$  functies zijn, en dat

$$f: \mathbb{R} \rightarrow \mathbb{R} : t \mapsto f(t) = h(g(t))$$

de samenstelling is van  $h$  en  $g$ .

a) Veronderstel dat gegeven is dat:

$$g(5) = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \quad g'(5) = \begin{bmatrix} -3 \\ -3 \end{bmatrix} \quad \nabla h(1,2) = \begin{bmatrix} 4 \\ 1 \end{bmatrix}$$

Gevraagd: wat is  $f'(5)$ ?

b) Veronderstel dat gegeven is dat:

$$g(2) = \begin{bmatrix} 4 \\ 1 \\ 0 \end{bmatrix} \quad g'(2) = \begin{bmatrix} 3 \\ 5 \\ -1 \end{bmatrix} \quad \nabla h(4,1,0) = \begin{bmatrix} 0 \\ 4 \\ 2 \end{bmatrix}$$

Gevraagd: wat is  $f'(2)$ ?

2. Veronderstel dat  $f(x, y) = x^2y$  en dat  $x(t) = 2t$  en  $y(t) = t$ .

- Bepaal  $x'(t)$  en  $y'(t)$ .
- Bepaal de partiële afgeleiden van  $f$  naar  $x$  en  $y$ .
- Bepaal tenslotte  $\frac{df}{dt}$  door gebruik te maken van de kettingregel in meerdere veranderlijken.

3. Veronderstel dat  $f(x, y) = \ln(xy)$  en  $x(t) = \cos(t)$  en  $y(t) = \sin(t)$ .

- Bepaal  $x'(t)$  en  $y'(t)$ .
- Bepaal de partiële afgeleiden van  $f$  naar  $x$  en  $y$ .
- Bepaal tenslotte  $\frac{df}{dt}$  door gebruik te maken van de kettingregel in meerdere veranderlijken.

## 7.6 Exacte numerieke bepaling gradiënt

Bij machinaal leren en zeker bij “deep learning” is de kostfunctie die we wensen te minimaliseren van de form

$$J: \mathbb{R}^n \rightarrow \mathbb{R}: \mathbf{w} \rightarrow J(\mathbf{w}),$$

waarbij de vector  $\mathbf{w}$  conceptueel alle aanpasbare parameters van het model bevat. In de context van “deep learning” is  $n$  gemakkelijk in de orde van een paar honderdduizend, miljoenen en soms zelfs miljarden<sup>3</sup>. Gradiënt

<sup>3</sup>GPT-3, een groot model voor natuurlijke taal, bevat 175 miljard trainbare parameters.

gebaseerde methoden hebben bij iedere update de gradiënt nodig van de kostfunctie  $J$  (typisch geëvalueerd op een subset van de trainingsvoorbeelden).

In de context van deep learning is men er meestal niet op uit om een symbolische uitdrukking te vinden voor de gradiënt, maar men zoekt de *numerieke waarde van de gradiënt in het huidige punt*, i.e. voor de huidige waarden van de parameters  $\mathbf{w}$ .

Om de gradiënt van  $\nabla_{\mathbf{w}}J$  te bepalen wordt in eerste instantie de graaf opgesteld die de berekening van  $J$  voorstelt. In deze graaf stelt elke knoop een (deel van de totale) berekening voor. Er is een boog van knoop  $n_i$  naar  $n_j$  als  $n_j$  de uitvoer van  $n_i$  gebruikt om zijn berekening uit te voeren. Deze graaf noemen we de *BEREKENINGSGRAAF* van de functie. De berekeningsgraaf is steeds een gerichte, en acyclische graaf, want als er een cykel aanwezig was dan zou het onmogelijk zijn om de berekening uit te voeren. Een gerichte en acyclische graaf heeft steeds een topologische sortering<sup>4</sup>, i.e. een volgorde van de knopen van de graaf zodanig dat alle pijlen vooruit wijzen. *De topologische sortering is m.a.w. een volgorde waarin de berekening kan worden uitgevoerd.*

Het idee is dat elke knoop in de berekeningsgraaf een “eenvoudige” berekening uitvoert op zijn invoerwaarden, zoals optellen, vermenigvuldigen, machtsverheffing of toepassing van een eenvoudige functie. Elke knoop in de berekeningsgraaf stelt m.a.w. een (eenvoudige) functie voor met nul<sup>5</sup>, één of meerdere argumenten en één enkele reële retourwaarde.

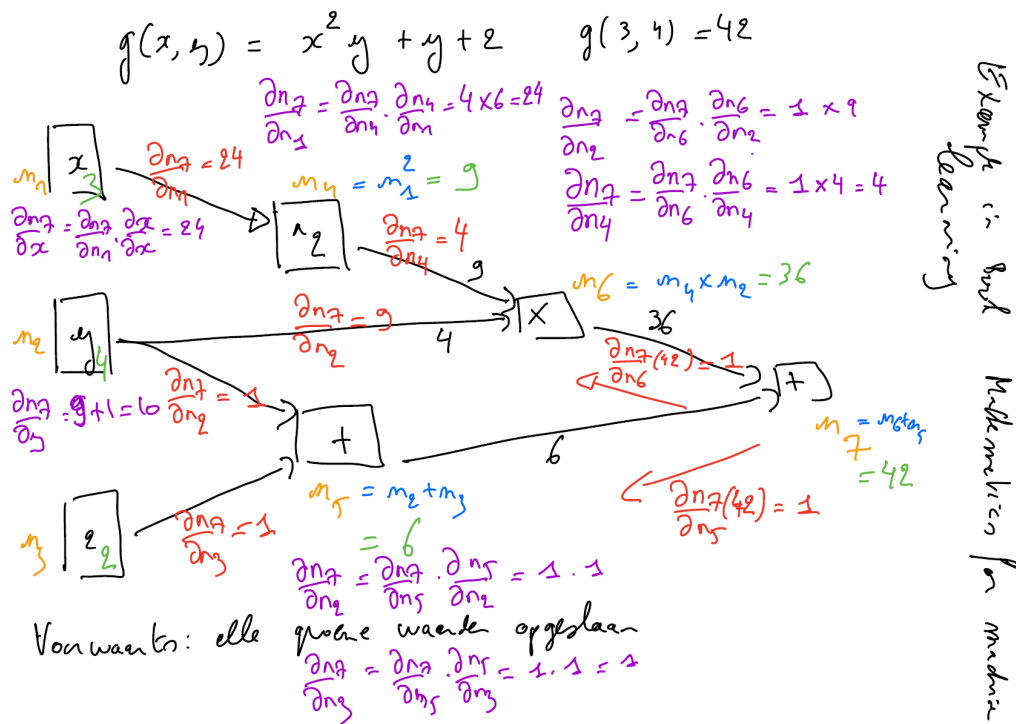
Zo’n knoop moet uiteraard in staat zijn om de voorwaartse berekening uit te voeren, maar moet ook over een tweede methode beschikken die uitgevoerd wordt wanneer de berekeningsgraaf achterstevoren wordt uitgevoerd! Het is immers zo dat *alle partiële afgeleiden in het huidig punt kunnen worden bepaald door één voorwaartse en één achterwaartse berekening op de berekeningsgraaf!* Een voorwaarde hierbij is wel dat elke knoop de intermediaire waarden opslaat tijdens de voorwaartse berekening.

De tweede methode waarover een knoop moet beschikken is het berekenen van de waarde van alle partiële afgeleiden m.b.t. zijn verschillende invoerparameters.

---

<sup>4</sup>Zie cursus Classic Computer Science Algorithms.

<sup>5</sup>Deze knoop stelt dan een constante of een variabele voor



**Figuur 7.13:** Berekeningsgraaf voor  $f(x, y) = x^2 y + y + 2$ , samen met de berekeningen nodig voor het uitvoeren van reverse mode automatic differentiation.

We illustreren de volledige procedure a.d.h.v. een uitgewerkt voorbeeld.

Beschouw de computationele graaf in Figuur 7.13. Deze stelt de berekening voor van  $f(x, y) = x^2 y + y + 2$ . We wensen de waarde te vinden van de gradiënt van  $f$  in het punt  $(3, 4)$ . De knopen zijn zodanig genoemd dat  $n_1$  t.e.m.  $n_7$  een geldige topologische sortering voorstelt.

In de voorwaartse stap wordt het volgende berekend.

- De knoop  $n_1$  heeft als returnwaarde 3.
- De knoop  $n_2$  heeft als returnwaarde 4.
- De knoop  $n_3$  heeft als returnwaarde 2.
- De knoop  $n_4 = n_1^2$  heeft als returnwaarde 9, en onthoudt dat  $n_1 = 3$ .

- De knoop  $n_5 = n_2 + n_3$  heeft als returnwaarde  $4 + 2 = 6$  en onthoudt dat  $n_2 = 4$  en  $n_3 = 2$ .
- De knoop  $n_6 = n_2 \times n_4$  heeft als returnwaarde  $4 \times 9 = 36$  en onthoudt dat  $n_2 = 4$  en  $n_4 = 9$ .
- De knoop  $n_7 = n_5 + n_6$  heeft als returnwaarde  $6 + 36 = 42$  en onthoudt dat  $n_5 = 6$  en  $n_6 = 36$ .

De achterwaartste berekening gaat nu als volgt:

- De knoop  $n_7$  heeft ingebouwd dat  $\frac{\partial n_7}{\partial n_6} = 1$ . Deze waarde wordt van  $n_7$  naar  $n_6$  gestuurd. Op dezelfde manier is  $\frac{\partial n_7}{\partial n_5} = 1$  en deze waarde wordt van  $n_7$  naar  $n_5$  gestuurd.
- De knoop  $n_6$  heeft ingebouwd dat  $\frac{\partial n_6}{\partial n_4} = n_2$  en omdat  $n_2 = 4$  weet  $n_6$  dat  $\frac{\partial n_6}{\partial n_4} = 4$ . We willen echter afgeleiden van  $n_7$  doorheen de graaf sturen. Door de kettingregel geldt

$$\frac{\partial n_7}{\partial n_4} = \frac{\partial n_7}{\partial n_6} \frac{\partial n_6}{\partial n_4} = 1 \times 4 = 4.$$

Deze waarde wordt door  $n_6$  naar de knoop  $n_4$  gestuurd.

De knoop  $n_6$  heeft ingebouwd dat  $\frac{\partial n_6}{\partial n_2} = n_4$  en omdat  $n_4 = 9$  weet  $n_6$  dat  $\frac{\partial n_6}{\partial n_2} = 9$ . Opnieuw wordt de kettingregel toegepast:

$$\frac{\partial n_7}{\partial n_2} = \frac{\partial n_7}{\partial n_6} \frac{\partial n_6}{\partial n_2} = 1 \times 9 = 9.$$

Deze waarde wordt van  $n_6$  naar  $n_2$  gestuurd.

- De knoop  $n_5$  heeft van  $n_7$  ontvangen dat  $\frac{\partial n_7}{\partial n_5} = 1$ . De knoop  $n_5$  heeft ingebouwd dat  $\frac{\partial n_5}{\partial n_3} = 1$ , en dus is

$$\frac{\partial n_7}{\partial n_3} = \frac{\partial n_7}{\partial n_5} \frac{\partial n_5}{\partial n_3} = 1 \times 1.$$

Deze waarde wordt door  $n_5$  naar  $n_3$  gestuurd. Op dezelfde manier geldt:

$$\frac{\partial n_7}{\partial n_2} = \frac{\partial n_7}{\partial n_5} \frac{\partial n_5}{\partial n_2} = 1 \times 1.$$

Deze waarde wordt door  $n_5$  naar  $n_2$  gestuurd.

- De knoop  $n_4$  heeft van  $n_6$  ontvangen dat  $\frac{\partial n_7}{\partial n_4} = 4$ . De knoop  $n_4$  heeft ingebouwd dat  $\frac{\partial n_4}{\partial n_1} = 2n_1$ . Omdat  $n_4$  heeft onthouden dat  $n_1 = 3$ , krijgen we  $\frac{\partial n_4}{\partial n_1} = 6$ . Door toepassen van de kettingregel vindt knoop  $n_4$  dat

$$\frac{\partial n_7}{\partial n_1} = \frac{\partial n_7}{\partial n_4} \frac{\partial n_4}{\partial n_1} = 4 \times 6 = 24,$$

waarbij de eerste factor  $\frac{\partial n_7}{\partial n_4}$  naar  $n_4$  werd gestuurd door  $n_6$ . De waarde 24 wordt door  $n_4$  naar  $n_1$  gezonden.

- De knoop  $n_3$  heeft geen invoer. Deze knoop ontvangt van  $n_5$  dat  $\frac{\partial n_7}{\partial n_3} = 1$ . Dat is meteen ook het finale antwoord als we knoop  $n_3$  zouden beschouwen als veranderlijk. (In dit geval beschouwen we knoop  $n_3$  echter als een constante en is de berekening van deze afgeleide niet van belang.)
- In knoop  $n_2$  gebeurt er iets interessants! Deze knoop ontvangt boodschappen van 2 andere knopen, nl. van  $n_6$  en  $n_5$ . De waarde van deze boodschappen was 9 en 1 respectievelijk. De knoop  $n_2$  telt de waarde van deze boodschappen op om  $\frac{\partial n_7}{\partial n_2}$  te bekomen. Dit betekent dat  $\frac{\partial n_7}{\partial n_2} = \frac{\partial n_7}{\partial y} = 9 + 1 = 10$ .
- In knoop  $n_1$  wordt van  $n_4$  de boodschap ontvangen dat  $\frac{\partial n_7}{\partial n_1} = 24$ . Dit is dan meteen ook de waarde van de afgeleide.

De conclusie van heel deze berekening is dus dat

$$\frac{\partial f}{\partial x}(3,4) = 24 \quad \text{en} \quad \frac{\partial f}{\partial y}(3,4) = 10.$$

De functie  $f$  is in dit geval zodanig eenvoudig dat we de afgeleiden ook heel snel analytisch kunnen bepalen

$$\frac{\partial f}{\partial x}(x,y) = 2xy \quad \text{en} \quad \frac{\partial f}{\partial y}(x,y) = x^2 + 1.$$

Als we de getalwaarden  $x = 3$  en  $y = 4$  invullen dan vinden we

$$\frac{\partial f}{\partial x}(3,4) = 2 \times 3 \times 4 = 24 \quad \text{en} \quad \frac{\partial f}{\partial y}(3,4) = 3^2 + 1 = 10. \quad (7.5)$$

We vinden dus (zoals gehoopt en verwacht) dezelfde getalwaarden terug!

Bibliotheken voor deep learning hebben deze functionaliteit ingebouwd (zij het op een efficiëntere manier). Hieronder een kort voorbeeld m.b.v. de bibliotheek Tensorflow:

```
import tensorflow as tf
import numpy as np

# Definieer twee variabelen
x = tf.Variable(3.0)
y = tf.Variable(4.0)

# Neem de berekening op tape op
with tf.GradientTape() as tape:
    g = x**2 * y + y + 2

# Print de uitvoer van de berekening
print(f"g = {g}")
# Bereken de afgeleiden
[dg_dx, dg_dy] = tape.gradient(g, [x, y])

print(f"dg_dx = {dg_dx.numpy()}, dg_dy = {dg_dy.numpy()}")
```

Dit stukje code heeft als uitvoer

```
g = 42.0
dg_dx = 24.0, dg_dy = 10.0
```

### 7.6.1 Oefeningen

1. Beschouw de functie  $f(x, y) = xy + \exp(xy)$ . Bepaal de exacte waarde van gradiënt van deze functie in het punt  $(-1, 2)$  met deze methode besproken in deze sectie. Beantwoord hiertoe volgende vragen.
  - a) Stel de berekeningsgraaf op voor deze functie.
  - b) Voer de voorwaartse berekening uit. Houd de opgeslagen waarden bij in een tabel.
  - c) Voer de achterwaartse berekening uit om de gradiënt te bepalen. Houd je berekeningen bij in een tabel.
2. Python **Implementatie van micrograd**  
 De bedoeling van dit practicum is om een Python-klasse `Value` te

schrijven die ons in staat stelt om “willekeurige” expressies te definiëren. Deze expressies kunnen geëvalueerd worden (i.e. hun waarde kan worden berekend), en bovendien kunnen deze expressies ook de waarde van de afgeleiden

$$\frac{\partial \text{out}}{\partial \text{in}}$$

bepalen waarbij `out` de finale uitvoer is van de expressie en `in` een willekeurige variabele is die deelneemt aan de expressie.

We starten met de volgende (onvolledige) code voor een `Value` object:

```
class Value:

    def __init__(self, data):
        self.data = data

    def __repr__(self):
        return f"Value(data={self.data})"
```

a) **Implementeer de optelling en de vermenigvuldiging**

Implementeer `__add__(self, other)` zodat twee `Value` objecten kunnen worden opgeteld.

Implementeer `__mul__(self, other)` zodat twee `Value` objecten kunnen worden vermenigvuldigd.

Na deze opdracht zou het volgende moeten werken:

```
a = Value(2.0)
b = Value(-3.0)
c = Value(10)
d = a * b + c
d
```

met als waarde

```
Value(data=4.0)
```

b) **Onthoud de voorgangers, de bewerking en een label**

Op dit moment onthoudt een `Value` object niet op welke manier het werd geconstrueerd. We moeten de voorgangers onthouden bij de creatie van een `Value` object. De implementatie van de `__init__` methode wordt



```
def __init__(self, data, prev=(), op='', label=''):
    self.data = data

    self._prev = set(prev)
    self._op = op
    self.label = label
```

Pas de methodes `__add__` en `__multiply__` aan zodat de voorgangers correct worden bijgehouden én zodat we weten welke bewerking gebruikt werd om een `Value` te construeren. Deze bewerking houden we bij als een korte string `+` en `*` voor de optelling en de vermenigvuldiging respectievelijk. Dit label dient later enkel om de berekeningsgraaf “mooier” te kunnen voorstellen.

De volgende code:

```
a = Value(2.0)
b = Value(-3.0)
c = Value(10)
d = a * b + c
d, d._prev, d._op
```

zou nu de volgende uitvoer moeten geven

```
(Value(data=4.0), {Value(data=-6.0), Value(data=10)}, '+')
```

### c) Visualiseer expressie en bereken (handmatig) de gradiënt

We voegen een veld `self.grad` toe dat de afgeleide van de uitvoer m.b.t. de (huidige) waarde voorstelt.

```
def __init__(self, data, prev=(), op='', label=''):
    self.data = data
    self.grad = 0.0 # derivative of output w.r.t. this value

    self._prev = set(prev)
    self._op = op
    self.label = label
```

In de notebook bij deze oefening wordt code gegeven om een netwerk te visualiseren.

Bouw de volgende expressie op en visualiseer ze met de volgende code:

```

a = Value(2.0, label='a')
b = Value(-3.0, label='b')
c = Value(10, label='c')
e = a * b; e.label = 'e'
d = e + c; d.label = 'd'
f = Value(-2.0, label='f')
L = d * f; L.label = 'l'

```

en

```
draw_dot(L)
```

Bereken (handmatig) de afgeleide van  $L$  m.b.t. elk van de `Value` objecten. Begin achteraan (dus bij  $L$ ) en doorloop de berekeningsgraaf van achter naar voor.

Tracht eventueel numeriek te verifiëren dat de waarden die je hebt berekend correct zijn. Stel dat je

$$\frac{\partial L}{\partial a}$$

wil bepalen, dan kan je tweemaal de waarde van  $L$  berekenen. Eén keer voor de huidige waarde van  $a$  nl.  $-2$  en dan nog een tweede keer voor  $-2 + h$  waarbij  $h$  bv. gelijk is aan  $0.0001$ . Vervolgens deel je het verschil van de tweede en de eerste waarde voor  $L$  door  $h$ . Dit geeft je een benadering voor de afgeleide van  $L$  naar  $a$ .

d) **Implementeer een `_backward` methode voor de optelling en de vermenigvuldiging**

We geven elk `Value` object een `_backward` functie die lokaal de kettingregel toepast. De opdracht van deze methode bestaat erin om de gradiënt van een `Value` object (i.e. `self.grad`) door te geven aan de gradiënten van voorgangers van deze knoop. Hierbij wordt dan lokaal één stap van de kettingregel toegepast.

```

def __init__(self, data, prev=(), op='', label=''):
    # Bestaande code hier

    self._backward = lambda : None # No-op by default

```

Wanneer een nieuw `Value` object wordt gecreëerd, wordt ook de `_backward` methode ingesteld, bv.

```
def __add__(self, other):
    out = Value(self.data + other.data,
                prev=(self, other), op='+')

    def _backward():
        # out = self + other
        # dus, lokale afgeleide d out / d self = 1
        # en d out / d other = 1
        # d L / d self = d L / d out * d out / d self
        # d L / d other = d L / d out * d out / d other
        self.grad = out.grad * 1.0
        other.grad = out.grad * 1.0

    out._backward = _backward

    return out
```

Opmerking: deze code heeft nog een bug die we later zullen oplossen!

Doe nu iets gelijkaardigs voor `__mul__` maar bedenk wat er in de plaats moet komen van de factoren 1.0 bij de optelling.

e) **Roep `_backward` in de juiste volgorde op**

Neem de expressie die we voordien hadden en roep `_backward` op om voor elke `Value` de correcte gradiënt te berekenen. Vergeet niet om `L.grad` te initialiseren op 1.0.

Merk op hoe je de `_backward()` methode in essentie “achterstevoren” oproept. Je begint achteraan en je werkt naar voren toe.

f) **Schrijf een methode `backward`**

De methode `backward()` wordt enkel opgeroepen voor het finale `Value` object dat de waarde is van de expressie. Oproepen van deze methode zal er voor zorgen dat het `.grad` veld van elk `Value` object in de expressie zal ingevuld worden met de correcte waarde.

De methode `backward` bouwt een topologische sortering op van de expressiegraaf. Daarna wordt deze lijst in omgekeerde volgorde doorlopen en wordt van elk `Value` object de methode `_backward` opgeroepen. De code voor de topologische sortering is reeds gegeven.

```
def backward(self):
    topo = []
    visited = set()
    def build_topo(v):
        if v not in visited:
            visited.add(v)
            for child in v._prev:
                build_topo(child)
            topo.append(v)

    ### JOUW CODE HIER
    ### EINDE JOUW CODE HIER
```

g) **Value die meerdere malen wordt gebruikt**

Bekijk de volgende code:

```
a = Value(3.0, label='a')
b = a + a ; b.label = 'b'
b.backward()
draw_dot(b)
```

Met de huidige implementatie geeft je code de waarde 1.0 voor de afgeleide

$$\frac{db}{da'}$$

terwijl de juiste waarde voor deze afgeleide gelijk is aan 2.0.

Haal deze bug uit de code die de gradiënt ten onrechte overschrijft wanneer een Value meerdere malen wordt gebruikt.

Test je code op de volgende expressie:

```
a = Value(-2.0, label='a')
b = Value(3.0, label='b')
d = a * b ; d.label = 'd'
e = a + b ; e.label = 'e'
f = d * e ; f.label = 'f'

f.backward()

draw_dot(f)
```

h) **Maak bewerkingen robuuster**

Op dit moment werkt de volgende code niet:

```
a = Value(2.0)
b = a + 1
```

Zorg ervoor dat de code voor `__add__` controleert of `other` een instantie is van `Value`. Als dat zo is, dan werkt de bestaande code reeds, anders gaan we er van uit dat `other` een “getal” is en dan maken we er een `Value` object van.

Pas de code voor vermenigvuldiging op dezelfde manier aan. Op dit moment zou de volgende code moeten werken:

```
a = Value(2.0)
b = a * 2 + 1
```

Het volgende werkt echter niet, terwijl het leuk zou zijn als het wel zou werken:

```
a = Value(2.0)
b = 2 * a
```

De reden hiervoor is dat Python nu de `*` van een integer oproept en die weet (uiteraard) niets af van onze `Value`-objecten. Los dit op door `__rmul__` te definiëren, zie [https://docs.python.org/3/reference/datamodel.html#object.\\_\\_rmul\\_\\_](https://docs.python.org/3/reference/datamodel.html#object.__rmul__).

Doe nu hetzelfde voor de optelling. De volgende code zou nu moeten werken:

```
a = Value(2.0)
1 + 2 * a
```

We maken de berekeningsgraaf uit Figuur 7.13.

```
x = Value(3.0)
y = Value(4.0)
z = x * x * y + y + 2
z
```

Dit geeft als uitvoer

```
Value(data=42.0)
```

Dit is inderdaad de juiste waarde. We bepalen de gradiënt m.b.v. `z.backward()`.

```
z.backward()
x.grad, y.grad
```

Dit geeft als uitvoer:

```
(24.0, 10.0)
```

Dit zijn inderdaad ook de waarden van de partiële afgeleiden die gevonden werden in vergelijking (7.5).

i) **Voeg extra bewerkingen toe**

In dit deel van de opdracht zullen we een paar extra bewerkingen toevoegen zodat de expressies die we kunnen opbouwen interessanter worden.

We starten met de tanh functie:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}},$$

met afgeleide

$$\tanh'(x) = 1 - (\tanh(x))^2$$

Voeg deze methode toe aan de `Value` klasse en zorg dat `_backward()` juist wordt ingevuld. Gebruik `math.exp` om de waarde te berekenen van de exponentiële functie toegepast op een floating-point getal.

Voeg de volgende extra methoden toe:

- de exponentiële functie: `exp`.
- machtsverheffing: `__pow__`. In dit geval beperken we `other` tot getallen (nl. integer of float)
- deling: `__truediv__`
- verschil: `__sub__`

Verifieer nu dat de `tanh` bewerking dezelfde gradiënt geeft als wanneer je andere bewerkingen gebruikt om stap voor stap de `tanh` te berekenen.

## Inleiding tot numpy

### A.1 Basisdatastructuur

Numpy (kort voor *numeric Python*) is een fundamentele Python-bibliotheek voor numerieke berekeningen. De basisdatastructuur in numpy is een  $N$ -dimensionale array `ndarray`. Daarnaast bevat numpy ook nuttige functies voor lineaire algebra, het genereren van random getallen, ...

Om numpy te kunnen gebruiken moet je de bibliotheek importeren. Dat gebeurt steeds met het volgende idioom:

```
import numpy as np
```

Je kan wat wij vectoren hebben genoemd in Hoofdstuk 1 op verschillende manieren aanmaken.

```
np.array([1, 2, 3, 4, 5]) # array([1, 2, 3, 4, 5])
np.zeros(5)               # array([0., 0., 0., 0., 0.])
np.ones(4)                # array([1., 1., 1., 1.])
np.arange(3, 8)           # array([3, 4, 5, 6, 7])
np.linspace(0, 1, 5)      # array([0., 0.25, 0.5, 0.75, 1.])
np.empty(2)               # array([8.7...e-312, 2.4...e-312])
np.full(4, 42)            # array([42, 42, 42, 42])
```

Bovenstaand codevoorbeeld toont de verschillende opties die je hebt bij het aanmaken van numpy `ndarrays`:

- Je kan een numpy array initialiseren op basis van een bestaande Python-lijst van getallen.
- Je kan de numpy array initialiseren met nullen of enen.
- Numpy heeft een equivalent van de Python `range` methode: `np.arange`. Merk op dat de rechtergrens niet is inbegrepen.
- Je kan heel snel een interval onderverdelen in stukjes van gelijke lengte met `np.linspace`.
- Je kan ook geheugen alloceren zonder dit een waarde te geven met `np.empty`. De precieze waarden in deze array zijn onbepaald: die hangen af van wat dat deel van het hoofdgeheugen op dat moment bevat.
- Je kan de array vullen met een constante waarde m.b.v. `np.full`.

Een aantal van deze methoden kan je ook gebruiken om matrices (zie Hoofdstuk 2) aan te maken. Bv.

```
np.zeros((3,4))
```

maakt een matrix aan met drie rijen en vier kolommen volledig gevuld met nullen. Op dezelfde manier kan je ook `np.ones`, `np.empty` en `np.full` gebruiken om matrices aan te maken. Een matrix kan ook geïnitieerd worden m.b.v. een geneste Python-lijst:

```
np.array([[1,2,3],[4,5,6]]) # array([[1, 2, 3],  
#                               [4, 5, 6]])
```

Het datatype voor numpy arrays is `numpy.ndarray`:

```
type(np.empty(5)) # numpy.ndarray
```

Numpy arrays hebben een aantal belangrijke attributen, zoals het datatype van de objecten (meestal getallen) die ze bevatten. Verder is ook de vorm (`shape`) van de arrays uitermate belangrijk.



```

a = np.array([[1.0, 2.0, 3.0], [4.0, 5.0, 6.0]])
a.dtype           # dtype('float64')
a.shape           # (2, 3)
np.zeros(5).dtype # dtype('float64')
np.zeros(5, dtype=np.int32).dtype # dtype('int32')
np.zeros(5).shape # (5,)
np.arange(3, 8).dtype # dtype('int32')

```

Het datatype van een numpy array kan je opvragen m.b.v. het attribuut `dtype`. Het default datatype van een numpy array is `float64`, behalve wanneer je expliciet een datatype opgeeft bij constructie, of wanneer de initiële waarden allemaal gehele getallen zijn. De `shape` van een array is steeds een tuple.

**Opmerking A.1** Alle elementen in een bepaalde `numpy.ndarray` hebben steeds hetzelfde type. Dit zorgt ervoor dat ze véél efficiënter zijn voor numerieke berekeningen dan Python-lijsten. ■

**Opmerking A.2** In deze cursus zullen we ons beperken tot vectoren en matrices maar de `shape` van numpy arrays kan in principe om het even wat zijn. Als je bv. een kleurenafbeelding wil opslaan van 200 pixels hoog en 400 pixels breed dan zal je een numpy array nodig hebben met `shape (200, 400, 3)`, omdat je voor een kleurenafbeelding 3 waarden nodig hebt voor elke pixel (de zogenaamde RGB-waarden). Stel dat je duizend zo'n afbeeldingen wil opslaan dan kan dat in aan numpy array met `shape (1000, 200, 400, 3)`. ■

In numpy heet elke dimensie een *as*, of in het Engels een *AXIS*. Het aantal assen noemt men in numpy de rang van een array. Om verwarring te vermijden met het begrip rang van een matrix (nl. de dimensie van de kolomruimte van een matrix) zoals gezien in Hoofdstuk 2 zullen wij spreken van de numpy-rang van een array. De grootte (of `SIZE`) van een numpy array is het aantal elementen in deze array.

```

a = np.array([[1, 1, 1], [0, 1, 1], [0, 0, 1]]) # array([[1, 1, 1],
                                                    #          [0, 1, 1],
                                                    #          [0, 0, 1]])
a.shape           # (3, 3)
a.size            # 9
a.ndim            # 2

```

De array in bovenstaande code heeft numpy-rang 2 terwijl de rang van de matrix gelijk is aan 3 (aangezien de drie kolommen lineair onafhankelijk zijn).

## A.2 Random numpy arrays genereren

Soms kan het nuttig zijn om random data te genereren, bv. om snel testdata te genereren.

In numpy doe je dit door eerst `default_rng` aan te roepen om een instantie te verkrijgen van een `Generator`. Daarna kan je de methoden van deze `Generator` aanroepen om (pseudo-)random waarden te verkrijgen.

Dit is bv. de idiomatische manier om twee keer 3 waarden te genereren die verdeeld zijn volgens een standaard normale verdeling<sup>1</sup>.

```
rng = np.random.default_rng()
vals = rng.standard_normal(3)
vals      # array([ 1.55260998, -0.38395445,  0.45356297])
more_vals = rng.standard_normal(3)
more_vals # array([-1.28386608, -0.87713461, -0.45373589])
```

Je kan uiteraard ook random waarden genereren in de vorm van een matrix. De volgende code genereert 28 getallen die verdeeld zijn volgens een standaard normale verdeling, georganiseerd in een matrix met 7 rijen en 4 kolommen (en vraagt vervolgens enkel de `shape` van deze matrix op).

```
rng.standard_normal((7,4)).shape # (7,4)
```

## A.3 Rekenkundige bewerkingen

Rekenkundige bewerkingen worden in numpy steeds *elementsgewijs* uitgevoerd.

---

<sup>1</sup>Je zal in het opleidingsonderdeel “Data Science and AI” leren wat dit precies wil zeggen, maar voor nu betekent dit eenvoudigweg dat je waarden verkrijgt die rond nul liggen en die zeer zelden groter dan drie zullen zijn in absolute waarde.

```

a = np.array([14, 23, 32])
b = np.array([5, 4, 3])
a + b      # array([19, 27, 35])
a - b      # array([ 9, 19, 29])
a * b      # array([70, 92, 96])
a / b      # array([ 2.8,  5.75, 10.66666667])
a // b     # array([ 2,  5, 10])
a % b      # array([4, 3, 2])
a ** b     # array([537824, 279841, 32768])

```

Het is hierbij belangrijk om op te merken dat de vermenigvuldiging m.b.v. de `*`-operator *niet* de matrixvermenigvuldiging is zoals gedefinieerd in Hoofdstuk 2. De getallen in de twee arrays worden eenvoudigweg twee aan twee vermenigvuldigd.

### A.3.1 Broadcasting

Wanneer de twee arrays niet dezelfde shape hebben dan zal numpy typisch toch proberen om een resultaat te berekenen aan de hand van de regels voor BROADCASTING. Meestal gebeurt er wat je zou verwachten. Als je een bewerking uitvoert met een numpy array en een getal dan wordt de bewerking uitgevoerd op elk element van de array en het getal.

```

a = np.array([14, 23, 32])
a + 1      # array([15, 24, 33])
2 * a      # array([28, 46, 64])
a ** 2     # array([ 196,  529, 1024])

```

Een ander typisch geval is wanneer men een bewerking uitvoert op een matrix en een vector.

```

a = np.array([[1, 2, 3], [4, 5, 6]])
b = np.array([-1, 0, 1])
a + b      # array([[0, 2, 4],
                  #       [3, 5, 7]])
a * b      # array([[-1,  0,  3],
                  #       [-4,  0,  6]])

```

Wat er conceptueel<sup>2</sup> gebeurt is het volgende:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \rightsquigarrow \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 2 & 4 \\ 3 & 5 & 7 \end{bmatrix}.$$

Een ander voorbeeld is het volgende:

```
a = np.array([[1],[2],[3]])
b = np.array([5,6])
a.shape      # (3,1)
b.shape      # (2,)
a + b # array([[6, 7],
              # [7, 8],
              # [8, 9]])
```

Conceptueel gebeurt er hier het volgende:

$$\begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} + \begin{bmatrix} 5 & 6 \end{bmatrix} \rightsquigarrow \begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \end{bmatrix} + \begin{bmatrix} 5 & 6 \\ 5 & 6 \\ 5 & 6 \end{bmatrix} = \begin{bmatrix} 6 & 7 \\ 7 & 8 \\ 8 & 9 \end{bmatrix}.$$

De regels voor broadcasting zijn de volgende:

1. Als de twee arrays niet dezelfde numpy-rang hebben dan worden er extra dimensies met lengte 1 toegevoegd vooraan de array met de kleinste numpy-rang. Bv.  $(5,3) + (3,) \rightsquigarrow (5,3) + (1,3)$ .
2. Arrays met lengte 1 in een bepaalde dimensie gedragen zich alsof ze de lengte hebben van de andere array in die dimensie. Het enige element in die dimensie wordt herhaald. Bv.  $(5,3) + (1,3) \rightsquigarrow (5,3) + (5,3)$ .
3. Nu moeten de twee arrays dezelfde shape hebben, anders zijn ze niet compatibel voor broadcasting.

**Voorbeeld A.3** Stel dat we een binaire operatie willen uitvoeren op twee arrays met shape  $(7,2,1)$  en  $(2,3)$  dan wordt wegens regel 1 voor broadcasting de tweede shape omgezet naar  $(1,2,3)$ . Vervolgens wordt de eerste shape  $(7,2,3)$  wegens regel 2 en de tweede shape wordt  $(7,2,3)$  wegens regel 2. Aangezien de twee shapes nu hetzelfde zijn kan de binaire operatie worden uitgevoerd op deze twee arrays. ■

<sup>2</sup>In de numpy implementatie wordt er geen extra geheugen gebruikt.

**Voorbeeld A.4 (Incompatibele arrays voor broadcasting)** Stel dat we een binaire operatie willen uitvoeren op twee arrays met shape  $(7, 2, 2)$  en  $(2, 3)$  dan wordt wegens regel 1 voor broadcasting de tweede shape omgezet naar  $(1, 2, 3)$ . Wegens regel twee wordt  $(1, 2, 3)$  omgezet naar  $(7, 2, 3)$  maar nu is  $(7, 2, 2) \neq (7, 2, 3)$ , en de binaire operatie kan niet worden uitgevoerd op deze twee arrays. ■

## A.4 Indexeren van arrays

### A.4.1 Indexeren van ééndimensionale arrays

Indexeren van ééndimensionale arrays werkt zeer gelijkaardig aan het indexeren van Python-lijsten:

```
a = np.array([1, 5, 3, 19, 13, 7, 3])
a[3]      # 19
a[1:3]    # array([5, 3])
a[2:-1]   # array([ 3, 19, 13,  7])
a[:3]     # array([1, 5, 3])
a[2::2]   # array([ 3, 13,  3])
a[::-1]   # array([ 3,  7, 13, 19,  3,  5,  1])
```

Via indexeren kan je eveneens elementen wijzigen.

```
a = np.array([1, 5, 3, 19, 13, 7, 3])
a[3] = 42
a # array([ 1,  5,  3, 42, 13,  7,  3])
```

Je kan ook meerdere elementen tegelijkertijd aanpassen.

```
a = np.array([1, 5, 3, 19, 13, 7, 3])
a[3:6] = [42, 43, 44]
a # array([ 1,  5,  3, 42, 43, 44,  3])
```

In tegenstelling tot Python-lijsten kan men ook één enkele waarde toewijzen aan een *slice*. Via broadcasting wordt deze dan toegekend aan alle elementen van de slice.

```
a = np.array([1, 5, 3, 19, 13, 7, 3])
a[3:6] = 42
a # array([ 1,  5,  3, 42, 42, 42,  3])
```

Je kan de numpy array echter niet groter of kleiner maken via assignatie aan een slice. M.a.w. dit werkt *niet*!

```
# Werkt niet
a = np.array([1, 5, 3, 19, 13, 7, 3])
a[2:5] = [1, 2, 3, 4, 5] # ValueError could not broadcast input
                        # array from shape (5,) into shape (3,)
```

Het volgende werkt eveneens niet.

```
# Werkt niet
a = np.array([1, 5, 3, 19, 13, 7, 3])
del a[2:5] # ValueError: cannot delete array elements
```

Let ook op: slices van een ndarray zijn *views op dezelfde onderliggende data*. M.a.w. als je wijzigingen aanbrengt aan de slice, dan wijzig je ook de originele array én omgekeerd:

```
a = np.array([1, 5, 3, 19, 13, 7, 3])
a_slice = a[2:6]
a_slice[1] = 1000
a # array([1, 5, 3, 1000, 13, 7, 3])
a[5] = 42
a_slice # array([3, 1000, 13, 42])
```

M.b.v. de `copy`-methode creëer je een kopie van de data, zodat wijzigingen aan de ene array (of slice) de andere niet langer beïnvloeden.

```
a = np.array([1, 5, 3, 19, 13, 7, 3])
a_slice = a[2:6].copy() # Maak een kopie van de data
a_slice[1] = 1000
a # array([1, 5, 3, 19, 13, 7, 3])
```

**Opmerking A.5** Let op met de datatypes van de array. Als je een array aanmaakt met een datatype dat gehele getallen bevat (zoals bv. `int64`) dan zal bij toewijzing aan de array een eventueel fractioneel deel van een floating-

point getal worden “afgekapt”.

```
a = np.array([1,2,3,4,5])
a.dtype      # dtype('int64')
a[2] = 2.75
a             # array([1, 2, 2, 4, 5])
```

Dit kan problemen geven wanneer je bv. elementaire rij-operaties probeert uit te voeren op een matrix die initieel enkel uit gehele waarden bestond. In dit geval moet je ervoor zorgen dat de matrix wordt aangemaakt met een datatype dat floating-point getallen kan bevatten. ■

### A.4.2 Indexeren van meerdimensionale arrays

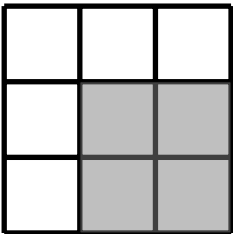
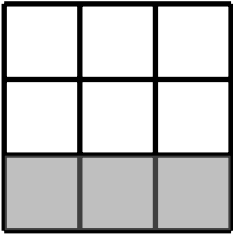
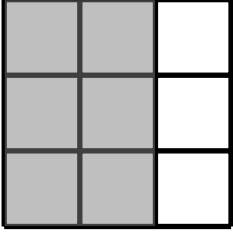
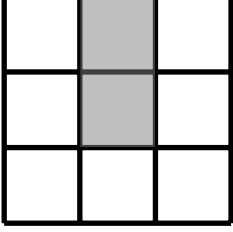
Indexeren in meerdimensionale arrays werkt op een gelijkaardige manier door een index of slice te speciëren voor elke as, gescheiden door komma's. Met een slice `:` geef je aan dat je alle elementen van die as wenst te selecteren. Bekijk het volgende voorbeeld, waarbij we ook gebruikmaken van de methode `reshape` om een array een nieuwe shape te geven. Opnieuw geeft dit, net zoals bij een slice, een view terug op dezelfde onderliggende data.

```
b = np.arange(12).reshape(3,4)
b  # array([[ 0,  1,  2,  3],
#          [ 4,  5,  6,  7],
#          [ 8,  9, 10, 11]])
b[1, 2]  # 6
b[1, :]  # array([4, 5, 6, 7])
b[:, 1]  # array([1, 5, 9])
```

Voor elke index die je aangeeft daalt de numpy-rang van het resultaat met 1: `b[1,2]` is een getal en `b[1,:]` is een array van numpy-rang 1. Zie het verschil met `b[1:2,:]` waarbij het resultaat nog steeds numpy-rang 2 heeft.

```
b = np.arange(12).reshape(3,4)
b[1:2, :]  # array([[4, 5, 6, 7]])
b[1:2, :].shape  # (1, 4)
```

Met *fancy indexing* kan je ook een lijst (of tuple) opgeven van de rijen (of

	expressie	shape
	<code>arr[1:, 1:]</code>	<code>(2, 2)</code>
	<code>arr[2]</code> <code>arr[2, :]</code> <code>arr[2:, :]</code>	<code>(3,)</code> <code>(3,)</code> <code>(1, 3)</code>
	<code>arr[:, :2]</code>	<code>(3, 2)</code>
	<code>arr[:, 1]</code> <code>arr[:, 1:2]</code>	<code>(2,)</code> <code>(2, 1)</code>

**Figuur A.1:** Illustratie van indexeren in een tweedimensionale numpy array.

kolommen) waarin je geïnteresseerd bent. Bv. hieronder selecteren we de laatste en de eerste rij en de middelste twee kolommen

```
b = np.arange(12).reshape(3, 4)
b[(-1, 0), 1:3] # array([[ 9, 10],
#                [ 1,  2]])
```

In Figuur A.1 zie je een overzicht van hoe men kan indexeren in een tweedimensionale numpy array.



## A.5 Aggregatiemethoden in numpy

Numpy beschikt eveneens over een heleboel methoden die één of andere aggregatie (i.e. samenvatting) berekenen van de elementen in een numpy array.

Hieronder een voorbeeldje m.b.v. de methode `sum`, we berekenen de som van alle elementen in de matrix en de som over de twee assen. De as waarover gesommeerd wordt verdwijnt dan uit het resultaat. We tonen ook hoe je de optie kan meegeven om de as waarover gesommeerd wordt te behouden.

```
a = np.arange(6).reshape(2,3)
a          # array([[0, 1, 2],
              #       [3, 4, 5]])
np.sum(a)   # 15
# Bereken som in de richting van axis 0, houd axis 1 over
np.sum(a, axis=0) # array([3, 5, 7])
# Bereken som in de richting van axis 1, houd axis 0 over
np.sum(a, axis=1) # array([ 3, 12])
np.sum(a, axis=1, keepdims=True) # array([[ 3],
                                      #       [12]])
```

Gelijkaardige methoden zijn `max`, `min` en `mean`. Bij wijze van voorbeeld de code waarmee de mean squared error (MSE) uit vergelijking (3.9) kan berekend worden. We geven ook onmiddellijk de berekening van de MSE uit Voorbeeld 3.11 mee.

```
def mse(yhat, y):
    return np.mean((yhat - y) ** 2)

yhat = np.array([0.6, 1.6, 2.6, 3.6, 4.6])
y = np.array([1.0, 2.0, 1.3, 3.75, 2.25])
mse(yhat, y) # 1.510999
```

Bemerk hoe compact je deze formule kan uitdrukken m.b.v. numpy, ook door de kracht van broadcasting.

## A.6 Lineaire algebra

Je kan een matrix transponeren door het `T`-attribuut te gebruiken. Dit geeft een view op de onderliggende data, dus veranderingen aan de ene array reflecteren zich in de andere array.

```
a = np.arange(6).reshape(2,3)
a          # array([[0, 1, 2],
              #       [3, 4, 5]])
a.T        # array([[0, 3],
              #       [1, 4],
              #       [2, 5]])

b = a.T
b[0,1] = 42
a          # array([[ 0,  1,  2],
              #       [42,  4,  5]])
```

Het inwendig product van twee vectoren wordt berekend m.b.v. de numpy-methode `dot`.

```
a = np.array([2.0, 0, -1.0])
b = np.array([3.0, 2.0, 1.0])
np.dot(a, b) # 5.0
```

Het matrix-vectorproduct kan eveneens via de `dot`-methode worden berekend, maar ook via de `matmul`-methode:

```
a = np.arange(6).reshape(2,3)
b = np.array([-1.0, 0, 1.0])
np.dot(a, b)      # array([2., 2.])
np.matmul(a, b)   # array([2., 2.])
```

Voor matrixvermenigvuldiging gebruik je de `matmul`-methode of de equivalente `@`-operator. Hieronder berekenen we het matrixproduct van een matrix en zijn getransponeerde (in twee volgordes).

```
a = np.arange(6).reshape(2,3)
a @ a.T      # array([[ 5, 14],
                #      [14, 50]])
np.matmul(a.T, a) # array([[ 9, 12, 15],
                #          [12, 17, 22],
                #          [15, 22, 29]])
```

Andere, meer geavanceerde, methoden voor lineaire algebra zijn aanwezig in `np.linalg`. Zo kan je de SVD en de pseudo-inverse van een matrix berekenen met `np.linalg.svd` en `np.linalg.pinv`. De QR-decompositie wordt bepaald door `np.linalg.qr`. De determinant van een matrix bereken je met `np.linalg.det`.

**Opmerking A.6** De LU-decompositie uit Sectie 3.3 is niet rechtstreeks beschikbaar in numpy. Hiervoor kan je `scipy.linalg.lu` gebruiken. ■

## A.7 Vergelijken van arrays

Je weet reeds uit het opleidingsonderdeel IT Fundamentals dat floating-point getallen slechts met een eindige precisie worden voorgesteld in het hoofdgeheugen van de computer. Daardoor is het niet aangeraden om floating-point getallen te vergelijken met de `==`-operator. Zo is volgens Python  $0.1 + 0.2 \neq 0.3$  zoals blijkt uit het codefragment hieronder.

```
0.1 + 0.2 == 0.3 # False
```

Vergelijken van numpy arrays die bestaan uit floating-point getallen kan je doen met de methode `np.allclose`. De code hieronder verifieert dat  $\mathbf{A}\mathbf{A}^{-1} = \mathbf{I}_2$  voor een bepaalde matrix  $\mathbf{A}$ . Wanneer je werkt met `==` dan krijg je niet het gewenste resultaat, maar met `np.allclose` wel.

```
A = np.array([[1,2],[3,4]])
A @ np.linalg.inv(A) == np.eye(2) # array([[False, False],
                #                          [ True, False]])
np.allclose(A @ np.linalg.inv(A), np.eye(2)) # True
```

Merk op dat de twee bij twee eenheidsmatrix werd aangemaakt met de methode `np.eye`.