HO
GENT

# Mathematics for Machine Learning

## Oplossingen oefeningen

Stijn Lievens en Koen Mertens

Professionele Bachelor in de Toegepaste Informatica

# Inhoudsopgave

# Deel I

# Lineaire Algebra

HOOFDSTUK 1

# Vectoren

## 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?

## 1.7 Oplossingen

1. a) De vectoren zijn lineair afhankelijk want
$$2\mathbf{a}-\mathbf{b}=\begin{bmatrix}0\\0\end{bmatrix}.$$
   b) De vectoren zijn lineair onafhankelijk.
   c) De vectoren zijn lineair onafhankelijk.
   d) De vectoren zijn lineair afhankelijk want
$$\mathbf{a}+\mathbf{b}+\mathbf{c}=\begin{bmatrix}0\\0\\0\end{bmatrix}.$$

2. $\mathbf{v}=2\mathbf{b_1}+3\mathbf{b_2}$. De coördinaten t.o.v. de basis bepaald door $\mathbf{b_1}$ en $\mathbf{b_2}$ worden dus gegeven door 2 en 3.

3. Nee, want de nulvector behoort niet tot deze verzameling en behoort tot elke deelruimte.

   Een tweede argument kan zijn dat de som van twee vectoren niet langer tot de deelverzameling behoort (want de middelste component is dan gelijk aan 2 i.p.v. 1).

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 $\mathbf{b}=\begin{bmatrix}3\\4\\-1\end{bmatrix}$ Oplossing: $\frac{1}{5}\mathbf{a}$

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

   c) $\mathbf{a}=\begin{bmatrix}1\\0\\2\end{bmatrix}$ en $\mathbf{b}=\begin{bmatrix}4\\-3\\-2\end{bmatrix}$ Oplossing: $\frac{0}{5}\mathbf{a}=\begin{bmatrix}0\\0\\0\end{bmatrix}$

```python
# Oefening Cosinussimilariteit
import numpy as np
import math

def cosinus_similariteit(x, y):
  assert x.shape == y.shape, "Verschillende lengte"
  assert len(x.shape) == 1, "Geen vector"

  return np.dot(x, y)/np.linalg.norm(x)/np.linalg.norm(y)

if __name__ == "__main__":
  x = np.array([1,2,3])
  assert math.isclose(cosinus_similariteit(x, x), 1.0)
  assert math.isclose(cosinus_similariteit(x, 2*x), 1.0)

  assert math.isclose(cosinus_similariteit(x, -x), -1.0)
  assert math.isclose(cosinus_similariteit(x, -2*x), -1.0)

  y = np.array([-1,-1,1])
  assert math.isclose(cosinus_similariteit(x, y), 0.0)
  assert math.isclose(cosinus_similariteit(x, -y), 0.0)
  assert math.isclose(cosinus_similariteit(x, 2*y), 0.0)

  x = np.array([1,0])
  y = np.array([np.cos(np.pi/3), np.sin(np.pi/3)])
  assert math.isclose(cosinus_similariteit(x, y), np.cos(np.pi/3))

  print("Einde")
```

**Figuur 1.1:** Code voor cosinus similariteit

5. Zie Figuur 1.1.

6. Zie Figuur 1.2.

7. We moeten aantonen dat $\mathbf{u}+\mathbf{v}$ en $\mathbf{u}-\mathbf{v}$ orthogonaal zijn, d.w.z. dat we moeten aantonen dat

$$(\mathbf{u}+\mathbf{v})\cdot(\mathbf{u}-\mathbf{v})=0.$$

```python
import numpy as np
import math

AANTAL_CONTROLES=100
LENGTE_VECTOR=10

if __name__ == "__main__":
  rng = np.random.default_rng()
  for _ in range(AANTAL_CONTROLES):
    u = rng.standard_normal(size=LENGTE_VECTOR) * 10
    v = rng.standard_normal(size=LENGTE_VECTOR) * 10
    w = rng.standard_normal(size=LENGTE_VECTOR) * 10

    alpha = (rng.uniform(1)-0.5) * 10
    beta  = (rng.uniform(1)-0.5) * 10

    assert math.isclose(np.dot(v,w), np.dot(w, v))

    links = np.dot(u, alpha * v + beta * w)
    rechts = alpha * np.dot(u, v) + beta * np.dot(u, w)

    assert math.isclose(links, rechts)

  print("Einde")
```

**Figuur 1.2:** Code om eigenschappen inwendig product te verifiëren.

Er geldt:

$$\begin{aligned}(\mathbf{u}+\mathbf{v})\cdot(\mathbf{u}-\mathbf{v}) &= \mathbf{u}\cdot\mathbf{u}-\mathbf{u}\cdot\mathbf{v}+\mathbf{v}\cdot\mathbf{u}-\mathbf{v}\cdot\mathbf{v}\\ &= \mathbf{u}\cdot\mathbf{u}-0+0-\mathbf{v}\cdot\mathbf{v}\\ &= \mathbf{u}\cdot\mathbf{u}-\mathbf{v}\cdot\mathbf{v}\\ &= 1-1\\ &= 0.\end{aligned}$$

Anders gezegd: het inwendig product $(\mathbf{u}+\mathbf{v})\cdot(\mathbf{u}-\mathbf{v})$ is gelijk aan $\sum_{i=1}^{n} u_i^2 - v_i^2$ waarbij zowel $\sum_{i=1}^{n} u_i^2$ als $\sum_{i=1}^{n} v_i^2$ gelijk zijn aan 1, dus is het inwendig product 0.

Om de lengte van $\mathbf{u}+\mathbf{v}$ te bepalen berekenen we eerst het inproduct van deze vector met zichzelf:

$$\begin{aligned}(\mathbf{u}+\mathbf{v})\cdot(\mathbf{u}+\mathbf{v}) &= \mathbf{u}\cdot\mathbf{u}+\mathbf{u}\cdot\mathbf{v}+\mathbf{v}\cdot\mathbf{u}+\mathbf{v}\cdot\mathbf{v}\\ &= 1+0+0+1\\ &= 2.\end{aligned}$$

De lengte van $\mathbf{u} + \mathbf{v}$ is m.a.w. gelijk aan $\sqrt{2}$. Dezelfde redenering geeft hetzelfde resultaat voor $\mathbf{u} - \mathbf{v}$.

Je kan ook onmiddellijk werken met een rechthoekige driehoek en volgens de stelling van Pythagoras is de gevraagde lengte dan $\sqrt{2}$.

HOOFDSTUK **2**

# Matrices

## 2.6 Oefeningen

1. Bereken het matrixproduct **AB** van de volgende matrices. Bereken ook **BA** indien dit mogelijk is en vergelijk met **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 matrix-product in Numpy.

3. `Numpy`. Verifieer de eigenschappen van het matrixproduct zoals vermeld in Eigenschap **??** 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?

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

### 2.6.1 Oplossingen

1. a) $\mathbf{A} = I_2$ dus $\mathbf{AB} = \mathbf{BA} = \mathbf{B}$
   b) $\mathbf{A} = \alpha I_2$ dus $\mathbf{AB} = \mathbf{BA} = \alpha\mathbf{B}$
   c) $\mathbf{AB} = \begin{bmatrix} a & b \\ 2c & 2d \end{bmatrix}$ en $\mathbf{BA} = \begin{bmatrix} a & 2b \\ c & 2d \end{bmatrix}$
   d) $\mathbf{AB} = \begin{bmatrix} 6 & 8 & 12 \\ 6 & 8 & 12 \\ 3 & 4 & 6 \end{bmatrix}$ en $\mathbf{BA} = \begin{bmatrix} 20 \end{bmatrix}$

2. Zie Figuur 2.1.

3. Zie Figuur 2.2.

4. Een matrix $\mathbf{S}$ is symmetrisch als $\mathbf{S}$ een vierkante matrix is als geldt dat

$$\mathbf{S}_{i,j} = \mathbf{S}_{j,i},$$

```python
import numpy as np

def matrix_vector_product(A, x):
  assert len(A.shape) == 2, "A is geen matrix"
  assert len(x.shape) == 1; "x is geen vector"
  assert A.shape[1] == x.shape[0], "A en x niet compatibel"

  product = A[:,0] * x[0]
  assert product.shape == (A.shape[0],)

  for kol_index in range(1, x.shape[0]):
    product += A[:, kol_index] * x[kol_index]

  return product

def matrix_vermenigvuldiging(A, B):
  assert len(A.shape) == 2, "A is geen matrix"
  assert len(B.shape) == 2, "A is geen matrix"
  assert A.shape[1] == B.shape[0], "A en B niet compatibel"

  product = np.empty(shape=(A.shape[0], B.shape[1]))

  for kol_index in range(B.shape[1]):
    product[:, kol_index] = matrix_vector_product(A, B[:, kol_index])

  return product


if __name__ == "__main__":
  rng = np.random.default_rng()
  for _ in range(100): # 100 controles
    A = rng.standard_normal(size=(10,7)) * 10
    x = rng.standard_normal(size=(7,)) * 10
    assert np.allclose(np.dot(A, x), matrix_vector_product(A, x))

    A = rng.standard_normal(size=(10,7)) * 10
    B = rng.standard_normal(size=(7,5)) * 10

    assert np.allclose(A @ B, matrix_vermenigvuldiging(A, B))

  print("Einde")
```

**Figuur 2.1:** Code voor matrixvermenigvuldiging als een reeks van matrix-vectorproducten.

```python
import numpy as np

AANTAL_CONTROLES=100

if __name__ == "__main__":
  rng = np.random.default_rng()
  for _ in range(AANTAL_CONTROLES):

    # Associativiteit
    A = rng.standard_normal(size=(10,7))*10
    B = rng.standard_normal(size=(7,5))*10
    C = rng.standard_normal(size=(5,8))*10

    links = A @ (B @ C)
    rechts = (A @ B) @ C
    assert np.allclose(links, rechts)

    # Distributiviteit
    A = rng.standard_normal(size=(10,7))*10
    B = rng.standard_normal(size=(7,5))*10
    C = rng.standard_normal(size=(7,5))*10

    links = A @ (B + C)
    rechts = A @ B + A @ C
    assert np.allclose(links, rechts)

    A = rng.standard_normal(size=(10,7))*10
    B = rng.standard_normal(size=(5,10))*10
    C = rng.standard_normal(size=(5,10))*10

    links = (B + C) @ A
    rechts = B @ A + C @ A
    assert np.allclose(links, rechts)

    # Interactie met scalair product
    A = rng.standard_normal(size=(10,7))*10
    B = rng.standard_normal(size=(7,5))*10

    alpha = 0.75
    links = (alpha * A) @ B
    rechts = A @ (alpha  * B)
    assert np.allclose(links, rechts)

    # Interactie met transponeren
    links = (A @ B).T
    rechts = B.T @ A.T
    assert np.allclose(links, rechts)

  print("Einde")
```

**Figuur 2.2:** Controleren van eigenschappen matrices

m.a.w. als het element in rij $i$ en kolom $j$ gelijk is aan het element in rij $j$ en kolom $i$.

We verifiëren enkel dat $\mathbf{A}^T\mathbf{A}$ een vierkante symmetrische matrix is. De redenering voor $\mathbf{A}\mathbf{A}^T$ is volledig analoog.

Als $\mathbf{A} \in \mathbb{R}^{n\times m}$ dan is $\mathbf{A}^T \in \mathbb{R}^{m\times n}$ en $\mathbf{A}^T\mathbf{A} \in \mathbb{R}^{m\times m}$ is een vierkante matrix. Om te verifiëren dat het ook een symmetrische matrix is maken we gebruik van de "klassieke" formule voor matrixvermenigvuldiging: als $\mathbf{C} = \mathbf{AB}$ dan geldt dat

$$\mathbf{C}_{i,j} = \sum_k \mathbf{A}_{i,k}\mathbf{B}_{k,j}.$$

We passen dit nu toe op

$$\mathbf{C} = \mathbf{A}^T\mathbf{A}$$

en we vinden

$$\begin{aligned}\mathbf{C}_{i,j} &= \sum_k \mathbf{A}^T_{i,k}\mathbf{A}_{k,j}.\\ &= \sum_k \mathbf{A}_{k,i}\mathbf{A}_{k,j}.\end{aligned}$$

Hierbij hebben we in de laatste stap gebruikgemaakt van het feit dat transponeren de rol van rijen en kolommen omwisselt. Merk op dat we in het rechterlid $i$ en $j$ kunnen omwisselen zonder dat er iets verandert! Dit betekent dat er in het linkerlid ook niets verandert, m.a.w.

$$\mathbf{C}_{i,j} = \mathbf{C}_{j,i}.$$

Dit is precies wat we wilden aantonen.

5. Aangezien $\mathbf{A}$ uit slechts twee kolommen bestaat is de rang van $\mathbf{A}$ kleiner dan 2 wanneer de ene kolom een veelvoud is van de andere kolom.

   We weten uit het gegeven dat $ad - bc = 0$.

   - Veronderstel dat $a \neq 0$, dan is $d = \frac{b}{a}c$ zodat

     $$\begin{bmatrix} b \\ d \end{bmatrix} = \frac{b}{a}\begin{bmatrix} a \\ c \end{bmatrix}.$$

     In dit geval is de tweede kolom een veelvoud van de eerste kolom.

   - Veronderstel dat $a = 0$, dit betekent dat $bc = 0$.

- Als $c = 0$, dan is de eerste kolom een kolom met enkel nullen en zijn de kolommen zeker lineair afhankelijk.
- Als $c \neq 0$, dan is $b = 0$ en dan geldt

$$\begin{bmatrix} b \\ d \end{bmatrix} = \begin{bmatrix} 0 \\ d \end{bmatrix} = \frac{d}{c} \begin{bmatrix} 0 \\ c \end{bmatrix} = \frac{d}{c} \begin{bmatrix} a \\ c \end{bmatrix}.$$

Opnieuw zijn de twee kolommen veelvouden van elkaar en de rang van $\mathbf{A}$ is hoogstens twee.

6. De projectiematrix wordt gegeven door

$$\mathbf{P} = \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T,$$

waarbij de matrix $\mathbf{A}$ de twee gegeven vectoren als kolommen heeft.

We berekenen in eerste instantie $\mathbf{A}^T\mathbf{A}$:

$$\mathbf{A}^T\mathbf{A} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} = \begin{bmatrix} 3 & 6 \\ 6 & 14 \end{bmatrix}.$$

De inverse hiervan wordt gegeven door

$$\frac{1}{6} \begin{bmatrix} 14 & -6 \\ -6 & 3 \end{bmatrix}.$$

We kunnen nu de projectiematrix berekenen:

$$\begin{aligned} \mathbf{P} &= \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T \\ &= \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \frac{1}{6} \begin{bmatrix} 14 & -6 \\ -6 & 3 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \end{bmatrix} \\ &= \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix} \begin{bmatrix} 4/3 & 1/3 & -2/3 \\ -1/2 & 0 & 1/2 \end{bmatrix} \\ &= \frac{1}{6} \begin{bmatrix} 5 & 2 & -1 \\ 2 & 2 & 2 \\ -1 & 2 & 5 \end{bmatrix}. \end{aligned}$$

De projectie van $\mathbf{b}$ wordt nu gegeven door

$$\mathbf{Pb} = \frac{1}{6} \begin{bmatrix} 5 & 2 & -1 \\ 2 & 2 & 2 \\ -1 & 2 & 5 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 2 \end{bmatrix} = \frac{1}{6} \begin{bmatrix} 7 \\ 10 \\ 13 \end{bmatrix}.$$

De coördinaten van de projectie van **b** t.o.v. de gegeven vectoren worden gegeven door de getallen $\alpha$ en $\beta$ zodat

$$\frac{1}{6}\begin{bmatrix}7\\10\\13\end{bmatrix} = \alpha\begin{bmatrix}1\\1\\1\end{bmatrix} + \beta\begin{bmatrix}1\\2\\3\end{bmatrix} = \mathbf{A}\begin{bmatrix}\alpha\\\beta\end{bmatrix}.$$

We zouden deze coëfficiënten kunnen vinden door een stelsel lineaire vergelijkingen op te lossen (zie later), maar eigenlijk kennen we het antwoord al bijna. De laatste bewerking die we hebben gedaan alvorens **P** te vinden was een matrix vermenigvuldigen met **A**. Als we deze matrix, nl.

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

vermenigvuldigen met **b** (aan de rechterkant) dan vinden we het antwoord:

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

Inderdaad, je kan verifiëren dat

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

Om 100 procent zeker te zijn van ons antwoord kunnen we verifiëren dat het verschil tussen **b** en de projectie **Pb** loodrecht staat op de gegeven vectoren $\mathbf{a_1}$ en $\mathbf{a_2}$. We vinden:

$$\mathbf{b} - \mathbf{Pb} = \begin{bmatrix}-1/6\\2/6\\-1/6\end{bmatrix}.$$

Je gaat nu zelf eenvoudig na dat het inwendig product van dit verschil met zowel $\mathbf{a_1}$ als $\mathbf{a_2}$ gelijk is aan nul.

7. **Eerste manier** We zoeken twee lineair onafhankelijke vectoren in het gegeven vlak, i.e. we zoeken twee drietallen die aan de vergelijking $x + y - z = 0$ voldoen. Het is eenvoudig om in te zien dat

$$\mathbf{a_1} = \begin{bmatrix}1\\-1\\0\end{bmatrix} \quad \text{en} \quad \mathbf{a_2} = \begin{bmatrix}1\\0\\1\end{bmatrix}$$

twee zo'n vectoren zijn. Stel

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

De projectiematrix $\mathbf{P}$ wordt dan gegeven door

$$\mathbf{P} = \mathbf{A}(\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T$$

We berekenen eerst

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

De inverse wordt dan gegeven door

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

Als we nu de berekening maken om $\mathbf{P}$ te berekenen dan vinden we:

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

Opmerking: merk op dat dit antwoord niet afhangt van de basis waarmee we gestart zijn. Als je zou starten met met een andere basis voor het vlak $x + y - z = 0$ dan zou je nog steeds dezelfde projectiematrix krijgen! (Zonder bewijs.)

Tweede opmerking: de vector

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

staat loodrecht op het vlak (dit wordt een normaalvector genoemd). De matrix $\mathbf{P}$ beeldt deze normaalvector af op de nulvector. Dit is precies wat we verwachten.

**Tweede manier** We kunnen starten met de projectie op de normaalvector

$$\mathbf{n} = \begin{bmatrix} 1 \\ 1 \\ -1 \end{bmatrix}$$

Deze projectiematrix noteren we $\mathbf{P}_{\text{rechte}}$ en is gelijk aan

$$\mathbf{P}_{\text{rechte}} = \frac{\mathbf{n}\mathbf{n}^T}{\mathbf{n}\cdot\mathbf{n}} = \frac{1}{3}\begin{bmatrix}1\\1\\-1\end{bmatrix}\begin{bmatrix}1 & 1 & -1\end{bmatrix} = \frac{1}{3}\begin{bmatrix}1 & 1 & -1\\1 & 1 & -1\\-1 & -1 & 1\end{bmatrix}$$

Bereken nu $\mathbf{I} - \mathbf{P}_{\text{rechte}}$ en we krijgen

$$\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix} - \frac{1}{3}\begin{bmatrix}1 & 1 & -1\\1 & 1 & -1\\-1 & -1 & 1\end{bmatrix} = \frac{1}{3}\begin{bmatrix}2 & -1 & 1\\-1 & 2 & 1\\1 & 1 & 2\end{bmatrix},$$

wat hetzelfde antwoord is als voorheen.

Het feit dat dit steeds werkt komt omdat we elke vector $\mathbf{b}$ kunnen schrijven als

$$\mathbf{b} = \mathbf{P}\mathbf{b} + \mathbf{P}_{\text{rechte}}\mathbf{b},$$

wat betekent dat

$$(\mathbf{I} - \mathbf{P} - \mathbf{P}_{\text{rechte}})\mathbf{b} = \mathbf{0}$$

voor *elke* vector $\mathbf{b}$, wat op zijn beurt betekent dat de matrix $\mathbf{I} - \mathbf{P} - \mathbf{P}_{\text{rechte}}$ gelijk is aan de nulmatrix.

8.

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

De fout is dat de inverse $\mathbf{A}^{-1}$ mogelijks niet bestaat. Wanneer $\mathbf{A}$ inverteerbaar is, dan is de redenering wel correct. Een inverteerbare idempotente matrix is inderdaad gelijk aan de eenheidsmatrix. Vergelijk dit met de reële getallen: als $x^2 = x$ dan is $x = 1$ op voorwaarde dat $x \neq 0$.

9. De eigenschap

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

is enkel geldig wanneer $\mathbf{A}$ en $\mathbf{B}$ allebei inverteerbare matrices zijn, dus de eerste herschrijving is reeds fout.

Wanneer $\mathbf{A}$ effectief inverteerbaar is, dan is de projectiematrix gelijk aan de eenheidsmatrix. Dit is logisch want als $\mathbf{A} \in \mathbb{R}^{n \times n}$ inverteerbaar is spant deze de hele ruimte $\mathbb{R}^n$ op, en elke vector wordt op zichzelf afgebeeld onder de projectie, dus is de projectiematrix gelijk aan de eenheidsmatrix.

HOOFDSTUK 3

# Stelsels lineaire vergelijkingen

### 3.1.1 Oefeningen

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

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

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) $\sqrt{x} + \sqrt{y} + \sqrt{z} = 1$

### 3.1.2 Oplossingen

1. We ordenen de variabelen als $x$, $y$, $z$ en $u$, dan kan het stelsel geschreven worden als $\mathbf{Ax} = \mathbf{b}$ met

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & -1 & 2 \\ 1 & 2 & -3 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} x \\ y \\ z \\ u \end{bmatrix} \quad \text{en} \quad \mathbf{b} = \begin{bmatrix} 3 \\ -7 \\ 1 \end{bmatrix}.$$

2. Stel $m$ de huidige leeftijd van de moeder en $z$ de leeftijd van de zoon, uitgedrukt in jaren. Dan stelt de eerste zin dat

$$m = z + 21$$

en de tweede zin dat

$$m + 6 = 5(z + 6).$$

We kunnen deze twee vergelijkingen als het volgende stelsel schrijven:

$$\begin{cases} m - z = 21 \\ m - 5z = 24 \end{cases}$$

3. a) $x^2 + x + 1 = 0$ is geen lineaire vergelijking want ze bevat een kwadraat van een variabele.
   b) $x^2 + y^2 = 1$ is geen lineaire vergelijking want ze bevat een kwadraat van een variabele.
   c) $x + y + xy = 0$ is geen lineaire vergelijking want ze bevat een product van variabelen.
   d) $2^x + y = 7$ is geen lineaire vergelijking want ze bevat een onbekende in een exponent.
   e) $\sqrt{x} + \sqrt{y} + \sqrt{z} = 1$ is geen lineaire vergelijking want er worden vierkantswortels genomen van variabelen.

### 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 herstelkosten 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 herstelkosten 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.

### 3.2.2 Oplossingen

1. Noem $b$, $a$ en $i$ het aandeel in de herstelkosten van respectievelijk de bouwheer, aannemer en ir.-architect.

   De totale kost is 300 000 EUR, dus

   $$b + a + i = 300000.$$

   De verantwoordelijkheid van de aannemer is dubbel zo hoog als die van bouwheer:

   $$a = 2b.$$

   De ir.-architect en de aannemer moeten samen 250 000 EUR betalen:

   $$a + i = 250000.$$

   Het stelsel dat we moeten oplossen is

   $$\begin{cases} b + a + i = 300000 \\ a - 2b = 0 \\ a + i = 250000 \end{cases}$$

   Als we de derde vergelijking van de eerste aftrekken dan vinden we onmiddellijk dat $b = 50000$. Uit de tweede vergelijking volgt dan

onmiddellijk dat $a = 100000$. Uit de derde vergelijking volgt dan dat $i = 250000 - 100000 = 150000$. Men controleert eenvoudig (door de gevonden waarden te substitueren) dat dit de juiste oplossing is voor dit stelsel.

2. Het stelsel dat we moeten oplossen is:

$$\begin{cases} m - z = 21 \\ m - 5z = 24 \end{cases}$$

Door de tweede vergelijking van de eerste af te trekken vinden we dat

$$4z = -3 \iff z = -\frac{3}{4},$$

waaruit wegens de eerste vergelijking volgt dat

$$m = 21 + z = 21 - \frac{3}{4} = \frac{81}{4}.$$

De moeder is dus 20 jaar en 3 maanden. De zoon is $-9$ maanden. Als we uitgaan van een natuurlijke bevruchting dan weten we dat de vader op dit moment nogal dicht bij de moeder is.

### 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)

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

### 3.3.4 Oplossingen

1. De LU-decompositie vindt men door Gaussische eliminatie uit te voeren en bij te houden welke veelvouden men gebruikt heeft in de rij-operaties.

   In dit geval maken we nullen onder het eerste pivotelement linksboven door

$$R_2 \leftarrow R_2 - 4R_1 \quad \text{en} \quad R_3 \leftarrow R_3 - 7R_1$$

   uit te voeren. Het resultaat is dan

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 0 & -6 & -12 \end{bmatrix}.$$

   Explicieter betekent dit dat

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 0 & -6 & -12 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ -4 & 1 & 0 \\ -7 & 0 & 1 \end{bmatrix} \mathbf{A}.$$

   In de volgende stap moeten we de volgende rij-operatie uitvoeren:

$$R_3 \leftarrow R_3 - \frac{-6}{-3} R_2 = R_3 - 2R_2$$

   en we vinden

$$\begin{bmatrix} 1 & 2 & 3 \\ 0 & -3 & -6 \\ 0 & 0 & 0 \end{bmatrix}$$

Dit is de bovendriehoeksmatrix **U**. Als we deze laatste rij-operatie expliciet uitschrijven dan vinden we

$$\begin{aligned}\begin{bmatrix}1 & 2 & 3\\ 0 & -3 & -6\\ 0 & 0 & 0\end{bmatrix} &= \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & -2 & 1\end{bmatrix}\begin{bmatrix}1 & 2 & 3\\ 0 & -3 & -6\\ 0 & -6 & -12\end{bmatrix}\\ &= \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & -2 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ -4 & 1 & 0\\ -7 & 0 & 1\end{bmatrix}\mathbf{A}\end{aligned}$$

Hieruit volgt dan (door de twee elementaire matrices één voor één te inverteren) de LU-decompositie van **A**

$$\begin{aligned}\mathbf{A} &= \begin{bmatrix}1 & 0 & 0\\ -4 & 1 & 0\\ -7 & 0 & 1\end{bmatrix}^{-1}\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & -2 & 1\end{bmatrix}^{-1}\begin{bmatrix}1 & 2 & 3\\ 0 & -3 & -6\\ 0 & 0 & 0\end{bmatrix}\\ &= \begin{bmatrix}1 & 0 & 0\\ 4 & 1 & 0\\ 7 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 2 & 1\end{bmatrix}\begin{bmatrix}1 & 2 & 3\\ 0 & -3 & -6\\ 0 & 0 & 0\end{bmatrix}\\ &= \begin{bmatrix}1 & 0 & 0\\ 4 & 1 & 0\\ 7 & 2 & 1\end{bmatrix}\begin{bmatrix}1 & 2 & 3\\ 0 & -3 & -6\\ 0 & 0 & 0\end{bmatrix}.\end{aligned}$$

Merk op dat men de elementen van de matrix **L** onmiddellijk kan invullen vanaf het moment dat men weet welke elementaire rij-operatie

$$R_i \leftarrow R_i - l_{i,j}R_j$$

men uitvoert.

2. a) Vermenigvuldig de twee matrices en verifieer dat men de eenheidsmatrix bekomt.

   b) Dit is een eenvoudige verificatie. Merk op dat dit patroon zich doorzet voor grotere matrices.

3. a) Als we de voorgestelde matrices **L** en **U** met elkaar vermenigvuldigen, dan vinden we:

$$\begin{bmatrix}a & 0\\ al+b & c\end{bmatrix}.$$

Maar: rechtsboven staat de waarde 0, terwijl in **A** rechtsboven een 1 staat. Deze matrix heeft dus geen LU-decompositie. Dit komt omdat het eerste pivotelement onmiddellijk gelijk is aan nul. Dit werkt niet omdat men moet delen door het pivotelement.

b) Als we de voorgestelde matrices **L** en **U** met elkaar vermenigvuldigen, dan vinden we:

$$\begin{bmatrix} a & b & c \\ al & bl+d & cl+e \\ am & bm+dn & cm+en+f \end{bmatrix}.$$

Als we dit nu gelijkstellen aan de matrix **A** dan vinden we voor de eerste rij dat

$$a = 1, \quad b = 1 \quad \text{en} \quad c = 0.$$

Voor de tweede rij volgt dan

$$al = 1 \implies l = 1, \quad bl + d = 1 \implies d = 0, \quad \text{en} \quad cl + e = 2 \implies e = 2.$$

Voor de derde rij volgt

$$am = 2 \implies m = 2$$

maar voor het volgnde element krijgen we

$$bm + dn = 3 \implies 2 + 0 = 3.$$

Dit is uiteraard strijdig en dit betekent dat de gevraagde LU-decompositie niet bestaat.

Opnieuw is de reden een pivotelement dat gelijk is aan nul. Het pivotelement op de tweede rij en tweede kolom is immers nul na het uitvoeren van de rijoperatie $R_2 \leftarrow R_2 - R_1$.

### 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.

### 3.4.5 Oplossingen

1. Zie Figuur 3.1.
2. Zie Figuur 3.2.
3. Zie Figuur 3.3.

### 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?

### 3.5.3 Oplossingen

1. a) Coëfficiëntenmatrix en rechterlid:

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

```python
import numpy as np
import scipy.linalg

def achterwaartse_substitutie(U, b):
  """ Ga er van uit dat U een bovendriehoeksmatrix is.
      Dit controleren we niet.
  """
  x = np.empty(shape=(U.shape[0],))
  n = U.shape[0]
  for r in range(n-1,-1,-1):
    x[r] = b[r]
    for c in range(r+1, n):
      x[r] -= U[r, c] * x[c]
    x[r] /= U[r, r]

  assert np.allclose(U @ x, b), "Fout in achterwaartse substitutie"

  return x

def voorwaartse_substitutie(L, b):
  """ Ga er van uit dat L een benedendriehoeksmatrix is.
      Dit controleren we niet.
  """
  x = np.empty(shape=(L.shape[0],))
  n = L.shape[0]
  for r in range(n):
    x[r] = b[r]
    for c in range(r):
      x[r] -= L[r, c] * x[c]
    x[r] /= L[r, r]

  assert np.allclose(L @ x, b), "Fout in voorwaartse substitutie"

  return x

def los_vierkant_stelsel_op(A, b):
  assert len(A.shape) == 2, "A is geen matrix"
  assert A.shape[0] == A.shape[1], "A s niet vierkant"
  assert len(b.shape) == 1, "b is geen vector"
  assert b.shape[0] == A.shape[0], "A en b niet compatibel"

  P, L, U = scipy.linalg.lu(A)
  # Opmerking: hier geldt A = P @ L @ U of dus P.T @ A = L @ U
  nieuwe_b = P.T @ b

  y = voorwaartse_substitutie(L, nieuwe_b)
  x = achterwaartse_substitutie(U, y)

  return x
```

**Figuur 3.1:** Eenvoudige code voor oplossen vierkant stelsel m.b.v. LU-decompositie

```python
def bereken_inverse_matrix(A):
  assert len(A.shape) == 2, "A is geen matrix"
  assert A.shape[0] == A.shape[1], "A is niet vierkant"

  n = A.shape[0]
  P, L, U = scipy.linalg.lu(A)
  B = np.eye(n)
  nieuwe_B = P.T @ B

  # Dit wordt de inverse
  X = np.empty_like(A)

  # Los n stelsels op met dezelfde L en U
  for c in range(n):
    y = voorwaartse_substitutie(L, nieuwe_B[:, c])
    X[:, c] = achterwaartse_substitutie(U, y)

  return X
```

**Figuur 3.2:** Code voor het berekenen van de inverse matrix.

b)

$$\begin{bmatrix} 2 & 1 & 0 \\ 1 & -1 & 1 \\ 1 & 1 & -1 \end{bmatrix} \underset{\sim}{R_2-2R_1, R_3-2R_1} \begin{bmatrix} 2 & 1 & 0 \\ 0 & -3 & 1 \\ 0 & -1 & -1 \end{bmatrix} \underset{\sim}{R3-\frac{1}{3}R_2} \begin{bmatrix} 2 & 1 & 0 \\ 0 & -3 & 1 \\ 0 & 0 & -\frac{4}{3} \end{bmatrix}$$

De laatste vergelijking zegt dat $0x + 0y = -4/3$. Dit duidt erop dat het stelsel strijdig is.

c) We berekenen $\mathbf{A}^T\mathbf{A}$ en we vinden

$$\mathbf{A}^T\mathbf{A} = \begin{bmatrix} 2 & 1 & 1 \\ 1 & -1 & 1 \end{bmatrix} \begin{bmatrix} 2 & 1 \\ 1 & -1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 6 & 2 \\ 2 & 3 \end{bmatrix}$$

Bovendien is

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

Het stelsel dat we dus oplossen i.p.v. het originele stelsel is

$$\begin{cases} 6x + 2y = 0 \\ 2x + 3y = -2. \end{cases}$$

```python
def bereken_determinant(A):
  assert len(A.shape) == 2, "A is geen matrix"
  assert A.shape[0] == A.shape[1], "A is niet vierkant"

  n = A.shape[0]

  # overschrijf A niet
  U = np.copy(A)

  teken = 1
  for r in range(n - 1):
    # Zoek grootste element in absolute waarde in A[r:, r]
    rij_max = r
    grootste = np.abs(U[r, r])
    for r2 in range(r+1, n):
      if np.abs(U[r2, r]) > grootste:
        grootste = np.abs(U[r2, r])
        rij_max = r2

    # Wissel indien nodig
    if r != rij_max:
      teken *= -1
      U[[r,rij_max]] = U[[rij_max, r]]

    # Doe rij operaties
    for r2 in range(r+1, n):
      U[r2, r:] = U[r2, r:] -  U[r2, r] /U[r,r] *  U[r, r:]

    # Vermenigvuldig alle elementen op de diagonaal
    d = U[0,0]
    for r in range(1, n):
      d *= U[r, r]

    return d * teken
```

**Figuur 3.3:** Code voor bereken determinant, zonder gebruik te maken van methodes in `numpy`.

Toepassen van Gaussische eliminatie levert het equivalente stelsel

$$\begin{cases} 6x + 2y = 0 \\ \quad \frac{7}{3}y = -2. \end{cases}$$

Hieruit volgt dat $y = -6/7$ en $x = -y/3 = 2/7$. (Merk op: je had ook kunnen $\mathbf{A}^T\mathbf{A}$ inverteren.)

Het rechterlid dat we effectief bekomen (i.p.v. $\mathbf{b}$) is

$$\begin{bmatrix} 2 & 1 \\ 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} 2/7 \\ -6/7 \end{bmatrix} = \begin{bmatrix} -2/7 \\ 8/7 \\ -4/7 \end{bmatrix}.$$

HOOFDSTUK 4

# Orthogonale matrices

### 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.)

### 4.1.2 Oplossingen

1. Voor de eerste vector geldt dat

$$\| \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \| = \sqrt{1^2 + 1^2 + 1^2} = \sqrt{3}.$$

Bijgevolg moet je $a = \frac{1}{\sqrt{3}}$ om deze vector te normeren. Op dezelfde manier vind je dat $b = \frac{1}{\sqrt{6}}$. Om de coëfficiënten te vinden van $\mathbf{v}$ zoeken we getallen $\alpha$ en $\beta$ zodat

$$\mathbf{v} = \alpha\mathbf{q_1} + \beta\mathbf{q_2}.$$

Het is eenvoudig om te controleren dat $\mathbf{q_1} \cdot \mathbf{q_2} = 0$.

Uit Eigenschap **??** weten we dat deze coëfficiënten worden gegeven door de inwendige producten van $\mathbf{v}$ en de orthonormale vectoren:

$$\alpha = \mathbf{q_1} \cdot \mathbf{v} = 7/\sqrt{3} \quad \text{en} \quad \beta = \mathbf{q_2} \cdot \mathbf{v} = 2/\sqrt{6}.$$

We kunnen de controle uitvoeren:

$$\frac{7}{\sqrt{3}} \begin{bmatrix} 1/\sqrt{3} \\ 1/\sqrt{3} \\ 1/\sqrt{3} \end{bmatrix} + 2/\sqrt{6} \begin{bmatrix} -1/\sqrt{6} \\ 2/\sqrt{6} \\ 1/\sqrt{6} \end{bmatrix} = \begin{bmatrix} 2 \\ 3 \\ 2 \end{bmatrix}.$$

### 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.

### 4.2.2 Oplossingen

1. We weten dat
$$\mathbf{Q}^T\mathbf{Q} = \mathbf{I}$$
waaruit volgt dat
$$\det(\mathbf{Q}^T\mathbf{Q}) = \det(\mathbf{I}) = 1.$$
Omdat algemeen geldt dat $\det(\mathbf{AB}) = \det(\mathbf{A})\det(\mathbf{B})$ volgt dat
$$\det(\mathbf{Q}^T)\det(\mathbf{Q}) = 1.$$
Tenslotte maken we gebruik van het feit dat in het algemeen geldt dat $\det(\mathbf{A}^T) = \det(\mathbf{A})$ en we bekomen
$$\det(\mathbf{Q})\det(\mathbf{Q}) = \det(\mathbf{Q})^2 = 1.$$
Hieruit volgt nu onmiddellijk dat de determinant van $\mathbf{Q}$ gelijk is aan $+1$ of $-1$.

2. Stel dat $\mathbf{Q_1}$ en $\mathbf{Q_2}$ orthogonale matrices zijn dan vinden we

$$\begin{aligned}(\mathbf{Q_1Q_2})^T(\mathbf{Q_1Q_2}) &= (\mathbf{Q_2}^T\mathbf{Q_1}^T)(\mathbf{Q_1Q_2}) \\ &= \mathbf{Q_2}^T(\mathbf{Q_1}^T\mathbf{Q_1})\mathbf{Q_2} \\ &= \mathbf{Q_2}^T(\mathbf{I})\mathbf{Q_2} \\ &= \mathbf{Q_2}^T\mathbf{Q_2} \\ &= \mathbf{I}.\end{aligned}$$

Dit toont aan dat het product van twee orthogonale matrices opnieuw een orthogonale matrix is.

### 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 willeurige vector $\mathbf{b}$.

### 4.3.3 Oplossingen

1. Definieer $\mathbf{Q}$ als de matrix bestaande uit de gegeven twee vectoren:

$$\mathbf{Q} = \begin{bmatrix}1/\sqrt{3} & -1/\sqrt{6}\\1/\sqrt{3} & 2/\sqrt{6}\\1/\sqrt{3} & -1/\sqrt{6}\end{bmatrix}$$

dan wordt volgens de vereenvoudige formule de projectiematrix gegeven door

$$\mathbf{P} = \mathbf{QQ}^T = \begin{bmatrix}1/\sqrt{3} & -1/\sqrt{6}\\1/\sqrt{3} & 2/\sqrt{6}\\1/\sqrt{3} & -1/\sqrt{6}\end{bmatrix}\begin{bmatrix}1/\sqrt{3} & 1/\sqrt{3} & 1/\sqrt{3}\\-1/\sqrt{6} & 2/\sqrt{6} & -1/\sqrt{6}\end{bmatrix} = \begin{bmatrix}1/2 & 0 & 1/2\\0 & 1 & 0\\1/2 & 0 & 1/2\end{bmatrix}.$$

De projectiematrix is m.a.w.

$$\mathbf{P} = \begin{bmatrix} 1/2 & 0 & 1/2 \\ 0 & 1 & 0 \\ 1/2 & 0 & 1/2 \end{bmatrix}.$$

2.

$$\mathbf{P} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} (x+z)/2 \\ y \\ (x+z)/2 \end{bmatrix}.$$

### 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 tupel bestaande uit (in die volgorde) $\mathbf{Q}$ en $\mathbf{R}$.

### 4.5.2 Oplossingen

1.

$$\mathbf{Q} = \begin{bmatrix} 2/3 & -1/\sqrt{18} \\ 2/3 & -1/\sqrt{18} \\ 1/3 & 4/\sqrt{18} \end{bmatrix} \quad \text{en} \quad \mathbf{R} = \begin{bmatrix} 3 & 3 \\ 0 & \sqrt{18} \end{bmatrix}.$$

2. Zie Figuur 4.1.

```python
import numpy as np

def qr_decompositie(A):
  (n, m) = A.shape
  Q = np.empty(shape=(n, m))

  Q[:, 0] = A[:, 0]
  for c in range(1, m):
    Q[:, c] = A[:, c]
    for i in range(c):
      Q[:, c] -= np.dot(Q[:, i], Q[:,c])/np.dot(Q[:, i], Q[:, i])* Q[:, i]

  # Kolommen normeren
  for c in range(m):
    Q[:, c] /= np.linalg.norm(Q[:,c])

  R = Q.T @ A

  return Q, R


if __name__ == '__main__':
  rng = np.random.default_rng()

  for _ in range(100):
    A = rng.standard_normal(size=(10, 7))*10
    Q, R = qr_decompositie(A)

    assert np.allclose(Q.T @ Q, np.eye(7))
    assert np.allclose(Q @ R, A)
    # Controleer dat R een bovendriehoeksmatrix is
    assert np.allclose(R, np.triu(R))

  print("Einde")
```

**Figuur 4.1:** Code voor het berekenen van de QR-decompositie.

HOOFDSTUK 5

# De singuliere waarden ontbinding

## 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

   $$\text{Tr}(\mathbf{X}^T\mathbf{X}) = \text{Tr}(\hat{\mathbf{\Sigma}}^2)$$

   lettend op het feit dat we hebben aangetoond dat

   $$\mathbf{X}^T\mathbf{X} = \mathbf{V}\hat{\mathbf{\Sigma}}^2\mathbf{V}^T?$$

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 **??**.

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()`.

## 5.6 Oplossingen

1. Met `numpy` kan je verifiëren dat de eerste twee eigenschappen voldaan zijn, maar de derde eigenschap niet. (Zie Figuur 5.1 voor mogelijke code.) De tweede eigenschap toont bijna onmiddellijk aan dat

$$\mathrm{Tr}(\mathbf{X}^T\mathbf{X}) = \mathrm{Tr}(\mathbf{V}\hat{\mathbf{\Sigma}}^2\mathbf{V}^T) = \mathrm{Tr}(\hat{\mathbf{\Sigma}}^2\mathbf{V}^T\mathbf{V}) = \mathrm{Tr}(\hat{\mathbf{\Sigma}}^2)$$

waarbij in de laatste stap werd gebruikmaakt van het feit dat $\mathbf{V}$ een orhtogonale matrix is.

2. a) De datamatrix $\mathbf{X}$ is gelijk aan

$$\mathbf{X} = \begin{bmatrix} 2 & 0 \\ 0 & -2 \\ 3 & -3 \\ -1 & 1 \end{bmatrix}$$

Het gemiddeld van de kolommen is 1 en $-1$ respectievelijk. De gecentreerde dataset wordt m.a.w. gegeven door

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

b) TO DO

```python
import numpy as np
import math

if __name__ == "__main__":
  rng = np.random.default_rng()

  # Eigenschap 1
  for _ in range(100):
    A = rng.standard_normal(size=(7,7)) * 10
    B = rng.standard_normal(size=(7,7)) * 10
    assert math.isclose(np.trace(A @ B), np.trace(B @ A))

  print("Eigenschap 1 geldig")

  # Eigenschap 2
  for _ in range(100):
    A = rng.standard_normal(size=(7,7)) * 10
    B = rng.standard_normal(size=(7,7)) * 10
    C = rng.standard_normal(size=(7,7)) * 10

    assert math.isclose(np.trace(A @ B @ C), np.trace(B @ C @ A))

  print("Eigenschap 2 geldig")

  # Eigenschap 3 (niet geldig)
  A = rng.standard_normal(size=(7,7)) * 10
  B = rng.standard_normal(size=(7,7)) * 10
  C = rng.standard_normal(size=(7,7)) * 10
  if not math.isclose(np.trace(A @ B @ C), np.trace(B @ A @ C)):
    print("Tegenvoorbeeld voor eigenschap 3")

  print("Einde")
```

**Figuur 5.1:** Code om eigenschappen m.b.t. het spoor te verifiëren.

c) Python geeft ons de numerieke SVD. We kunnen hierin echter "mooie getallen" herkennen.

$$\hat{\mathbf{U}} = \begin{bmatrix} 0 & \sqrt{2}/2 \\ 0 & -\sqrt{2}/2 \\ -\sqrt{2}/2 & 0 \\ \sqrt{2}/2 & 0 \end{bmatrix}, \quad \hat{\mathbf{\Sigma}} = \begin{bmatrix} 4 & 0 \\ 0 & 2 \end{bmatrix}, \quad \mathbf{V} = \begin{bmatrix} -\sqrt{2}/2 & \sqrt{2}/2 \\ \sqrt{2}/2 & -\sqrt{2}/2 \end{bmatrix}$$

d) De (richting) va eerste principale component is gelijk aan de tweede bissectrice, en wijst in dit geval naar het noordwesten. Twee punten hebben coördinaat nul. De andere twee liggen ongeveer 1.4 van de oorsprong.

e) Bereken `X_centered @ V[:, 0]`, de inwendige producten van de gecentreerde datapunten met de eerste principale component.

f) Verklaard percentage variabiliteit:

$$\frac{\sigma_1^2}{\sigma_1^2 + \sigma_2^2} = \frac{16}{16+4} = 80\%.$$

3. a) M.b.v. `numpy` vinden we dat

$$\mathbf{U} = \begin{bmatrix} -1/\sqrt{5} & 0 & 2/\sqrt{5} \\ 0 & -1 & 0 \\ -2/\sqrt{5} & 0 & -1/\sqrt{5} \end{bmatrix}, \quad \mathbf{\Sigma} = \begin{bmatrix} 5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad \mathbf{V}^T = \mathbf{U}^T.$$

b) De pseudoinverse wordt bijgevolg gegeven door

$$\mathbf{A}^+ = \mathbf{V} \begin{bmatrix} 1/5 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \mathbf{U}^T = \begin{bmatrix} 4/100 & 0 & 8/100 \\ 0 & 1 & 0 \\ 8/100 & 0 & 16/100 \end{bmatrix}$$

c) We vinden eenvoudig dat

$$\mathbf{A}^+\mathbf{A} = \begin{bmatrix} 2/10 & 0 & 4/10 \\ 0 & 1 & 0 \\ 4/10 & 0 & 8/10 \end{bmatrix}$$

Passen we dit nu toe op een rij van $\mathbf{A}$, bv. op de eerste rij

$$\mathbf{A}^+\mathbf{A} \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 2/10 & 0 & 4/10 \\ 0 & 1 & 0 \\ 4/10 & 0 & 8/10 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix}.$$

Dit lukt ook voor de andere rijen.

d) Dit hebben we eigenlijk net bewezen aangezien zowel $\mathbf{A}$ en $\mathbf{A}^+$ symmetrische matrices zijn.

4. a) Als er een rechte $y = ax + b$ zou passen door de vier punten dan zouden de coëffiënten $a$ en $b$ de oplossing zijn van het volgend stelsel lineaire vergelijkingen:

$$\mathbf{Ax} = \mathbf{b} \iff \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 3 \\ 1 & 4 \end{bmatrix} \begin{bmatrix} b \\ a \end{bmatrix} = \begin{bmatrix} 0 \\ 8 \\ 8 \\ 20 \end{bmatrix}.$$

Dit stelsel is echter strijdig omdat zo'n rechte niet bestaat, maar we kunnen trachten de "beste" oplossing te vinden m.b.v. de pseudoinverse.

De pseudoinvere van $\mathbf{A}$ wordt volgens `numpy` gegeven door:

$$\mathbf{A}^+ = \begin{bmatrix} 0.65 & 0.45 & 0.05 & -0.15 \\ -0.2 & -0.1 & 0.1 & 0.2 \end{bmatrix}.$$

Als we deze matrix vermenigvuldigen met het rechterlid

$$\begin{bmatrix} 0 \\ 8 \\ 8 \\ 20 \end{bmatrix}$$

dan vinden we

$$\begin{bmatrix} 1 \\ 4 \end{bmatrix}$$

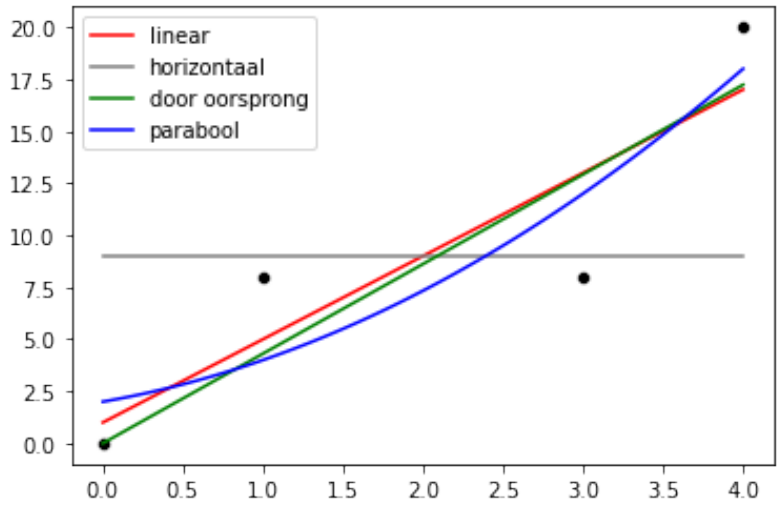
De best passende rechte wordt m.a.w. gegeven door $y = 1 + 4x$.

b) Om de best passende horizontale rechte te vinden moeten we het stelsel

$$\begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \begin{bmatrix} a \end{bmatrix} = \begin{bmatrix} 0 \\ 8 \\ 8 \\ 20 \end{bmatrix}$$

oplossen. De pseudoinverse van de coëfficiëntenmatrix is gelijk aan

$$\begin{bmatrix} 1/4 & 1/4 & 1/4 & 1/4 \end{bmatrix}$$




**Figuur 5.2:** Vier datapunten en verschillende best passende curves.

en de waarde voor $a$ is gelijk aan 9. (Dit is, uiteraard, het gemiddelde van de gegeven $y$-waarden. Dit zullen we later nog eens bewijzen m.b.v. afgeleiden.)

c) In dit geval vinden we een best passende rechte door de oorsprong met als vergelijking $y = 4.308x$ (afgerond om drie cijfers na de komma).

d) Om de best passende parabool $y = ax^2 + bx + c$ te vinden moeten we volgend stelsels oplossen:

$$\begin{bmatrix} 1 & 0 & 0 \\ 1 & 1 & 1 \\ 1 & 3 & 9 \\ 1 & 4 & 16 \end{bmatrix} \begin{bmatrix} c \\ b \\ a \end{bmatrix} = \begin{bmatrix} 0 \\ 8 \\ 8 \\ 20 \end{bmatrix}.$$

Bereken van de pseudoinverse en vermenigvuldigen met het rechterlid levert dat

$$y = \frac{2}{3}x^2 + \frac{4}{3}x + 2$$

de best passende parabool is.

In Figuur 5.2 worden de grafieken van alle best passende functies getoond, samen met de originele datapunten.

5. Zie code in Figuur 5.3.

```python
import numpy as np
import math

# 'Eenvoudige'/manuele methode
def pseudoinverse(A):
  U, S, Vt = np.linalg.svd(A, full_matrices=True)

  Splus = np.zeros(shape=A.T.shape)

  for i in range(min(A.shape)):
    Splus[i, i] = 0.0 if math.isclose(S[i], 0.0, abs_tol=10**-9) else 1/S[i]

  return Vt.T @ Splus @ U.T

# Gebruikmakend van numpy methodes
def pseudoinverse2(A):
  U, S, Vt = np.linalg.svd(A, full_matrices=True)

  # Hier kan je een RuntimeWarning krijgen maar het resultaat is correct
  Splus = np.where(np.isclose(S, 0.0), 0.0, 1/S)

  SplusFull = np.zeros_like(A.T)

  np.fill_diagonal(SplusFull, Splus)

  return Vt.T @  SplusFull @ U.T

# Gebruikmakend van numpy methodes en zuinige SVD
def pseudoinverse3(A):
  U, S, Vt = np.linalg.svd(A, full_matrices=False)

  # Hier kan je een RuntimeWarning krijgen maar het resultaat is correct
  Splus = np.where(np.isclose(S, 0.0), 0.0, 1/S)

  # Zorg dat SplusFull tussen Vt.T en U.T past !
  SplusFull = np.zeros(shape=(Vt.T.shape[1], U.T.shape[0]))

  np.fill_diagonal(SplusFull, Splus)

  return Vt.T @  SplusFull @ U.T

def test_pseudoinverse(A):
  Aplus = pseudoinverse(A)

  assert np.allclose(Aplus @ A @ A.T, A.T), "Fout bij rijen"
  assert np.allclose(A @ Aplus @ A, A), "Fout bij kolommen "

  return True
```

**Figuur 5.3:** Code voor het berekenen van de pseudoinverse.

# Deel II

# Analyse

HOOFDSTUK 6

# Reële functies in één veranderlijke

### 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:

$$f: \mathbb{R} \to \mathbb{R}: x \mapsto f(x) = \lfloor x \rfloor = \text{grootste } z \in \mathbb{Z} \text{ waarvoor } z \leq x.$$

   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) $\lim_{x \to 0} \frac{\sin(x)}{x}$

   b) $\lim_{x \to 0} \frac{\tan(x)}{x}$

   c) $\lim_{x \to \infty} \frac{3x^2 + x + 1}{x^2 - 2}$

   d) $\lim_{x \to 1} \frac{3x^2 + x + 1}{x - 1}$

   e) $\lim_{x \to 1} \frac{x^2 - 3x + 2}{x - 1}$

### 6.1.4 Oplossingen

1. a) Het domein van deze functie is $\mathbb{R}$, het beeld is $\mathbb{Z}$. Je kan immers elk reëel getal afronden naar beneden en het resultaat is steeds een geheel getal.

   b) De functie is continu in punten die geen geheel getal zijn. In de punten $z$ die een geheel getal zijn is de functie niet continue omdat de linkerlimiet gelijk is aan $z-1$, terwijl de rechterlimiet gelijk is aan $z$.

2. a) $\lim_{x\to 0} \frac{\sin(x)}{x} = 1$

   b) $\lim_{x\to 0} \frac{\tan(x)}{x} = 1$

   c) $\lim_{x\to \infty} \frac{3x^2+x+1}{x^2-2} = 3$

   d) $\lim_{x\to 1} \frac{3x^2+x+1}{x-1}$ bestaat niet. De linkerlimiet is $-\infty$ terwijl de rechterlimiet $+\infty$ is.

   e) $\lim_{x\to 1} \frac{x^2-3x+2}{x-1} = -1$

### 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

$$\sin(x) - \sin(y) = 2\cos\left(\frac{x+y}{2}\right)\sin\left(\frac{x-y}{2}\right)$$
$$\cos(x) - \cos(y) = -2\sin\left(\frac{x+y}{2}\right)\sin\left(\frac{x-y}{2}\right).$$

   In sectie 6.1.3 heb je ook een interessante limiet gezien die kan helpen bij het beantwoorden van deze vraag.

### 6.2.2 Oplossingen

1. De limiet die we moeten berekenen is:

$$\begin{aligned}\lim_{h\to 0}\frac{\sqrt{x+h}-\sqrt{x}}{h} &= \lim_{h\to 0}\frac{(\sqrt{x+h}-\sqrt{x})(\sqrt{x+h}+\sqrt{x})}{h(\sqrt{x+h}+\sqrt{x})}\\ &= \lim_{h\to 0}\frac{x+h-x}{h(\sqrt{x+h}+\sqrt{x})}\\ &= \lim_{h\to 0}\frac{h}{h(\sqrt{x+h}+\sqrt{x})}\\ &= \lim_{h\to 0}\frac{1}{\sqrt{x+h}+\sqrt{x}}\\ &= \frac{1}{2\sqrt{x}}\end{aligned}$$

   De afgeleide functie van $\sqrt{x}$ is m.a.w. $\frac{1}{2\sqrt{x}}$ wanneer $x > 0$.

2. Om de afgeleide van de sinus te berekenen moeten we de volgende limiet bepalen:

$$\begin{aligned}\lim_{h\to 0}\frac{\sin(x+h)-\sin(x)}{h} &= \lim_{h\to 0}\frac{2\cos(x+h/2)\sin(h/2)}{h}\\ &= \lim_{h\to 0}\cos(x+h/2)\lim_{h\to 0}\frac{\sin(h/2)}{h/2}\\ &= \cos(x)\times 1\\ &= \cos(x).\end{aligned}$$

   We hebben m.a.w. gevonden dat

$$\sin'(x) = \cos(x).$$

   Op gelijkaardige manier vind je dat

$$\cos'(x) = -\sin(x).$$

### 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) $f(x) = \dfrac{x}{x^2+1}$
e) $f(x) = \dfrac{\sin(x)}{\cos(x)}$ (of $f(x) = \tan(x)$).

### 6.4.2 Oplossingen

1. 
   - $f'(x) = 10(x+7)^9$
   - $f'(x) = (\sin(x) + \cos(x)) + x(\cos(x) - \sin(x))$
   - $f'(x) = 1.$
   - $f(x) = x(x^2+1)^{-1}$ en dus is
   $$f'(x) = (x^2+1)^{-1} - x(x^2+1)^{-2}(2x) = \frac{x^2+1-2x^2}{(x^2+1)^2} = \frac{-x^2+1}{(x^2+1)^2}.$$
   - $f(x) = \sin(x)\cos^{-1}(x)$ en bijgevolg is
   $$\begin{aligned} f'(x) &= \cos(x)\cos^{-1}(x) - \sin(x)\cos^2(x)(-\sin(x)) \\ &= \frac{\cos^2(x) + \sin^2(x)}{\cos^2(x)} \\ &= \frac{1}{\cos^2(x)}. \end{aligned}$$

### 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 **??** werd gezien. Wat hebben we gevonden?

2. Gegeven een dataset bestaande uit $m$ koppels

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

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 cylindervormig 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 $V$).

### 6.5.2 Oplossingen

1. Voor de eenvoud minimaliseren we

$$\|p\mathbf{a} - \mathbf{b}\|^2$$

i.p.v.

$$\|p\mathbf{a} - \mathbf{b}\|.$$

Dit mag omdat kwadrateren een stijgende functie is in $\mathbb{R}^+$, m.a.w. het minimum verandert niet (en de bijhorende minimumwaarde wordt gekwadrateerd.)

Stel

$$f(p) = \|p\mathbf{a} - \mathbf{b}\|^2 = \sum_{i=1}^{n} (pa_i - b_i)^2.$$

We berekenen de afgeleide functie naar $p$:

$$f'(p) = 2 \sum_{i=1}^{n} (pa_i - b_i) a_i.$$

Als we deze afgeleide gelijk aan nul stellen dan vinden we:

$$\begin{aligned} f'(p) = 0 &\iff 2\sum_{i=1}^{n}(pa_i - b_i)a_i = 0 \\ &\iff \sum_{i=1}^{n}(pa_i - b_i)a_i = 0 \\ &\iff p\mathbf{a}\cdot\mathbf{a} - \mathbf{a}\cdot\mathbf{b} = 0. \end{aligned}$$

Hieruit volgt dat

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

Dit is precies de formule voor de projectie op (de deelruimte opgespannen door) een vector.

Opmerking: we hebben nog niet aangetoond dat dit effectief een minimum is. Hiertoe kunnen we de tweede afgeleide berekenen:

$$f''(p) = 2\sum_{i=1}^{n} a_i^2.$$

Deze is steeds groter dan nul, en dus is het gevonden punt minimum.

2. Stel

$$f(c) = \frac{1}{m}\sum_{i=1}^{m}(c - y^{(i)})^2,$$

dan is

$$f'(c) = \frac{2}{m}\sum_{i=1}^{m}(c - y^{(i)}).$$

Als we deze afgeleide gelijk stellen aan nul dan vinden we

$$\begin{aligned} f'(c) = 0 &\iff \frac{2}{m}\sum_{i=1}^{m}(c - y^{(i)}) = 0 \\ &\iff \sum_{i=1}^{m}(c - y^{(i)}) = 0 \\ &\iff mc - \sum_{i=1}^{m} y^{(i)} = 0 \\ &\iff c = \frac{1}{m}\sum_{i=1}^{m} y^{(i)}. \end{aligned}$$

M.a.w. de beste $c$-waarde is het gemiddelde van alle $y$-waarden.

Opmerking: opnieuw moeten we in principe nog controleren dat we een minimum hebben gevonden.

3. Veronderstel dat het grondvlak van het blikje straal $r$ heeft en dat de hoogte van het blikje gelijk is aan $h$. Dan weten we dat

$$V = \pi r^2 h \tag{6.1}$$

Verder weten we dat de oppervlakte van het blikje gelijk is aan

$$\begin{aligned} S &= 2 \times \text{oppervlakte grondvlak} + \text{oppervlakte mantel} \\ &= 2\pi r^2 + 2\pi r h. \end{aligned}$$

Uit de vergelijking voor het volume leiden we af dat

$$h = \frac{V}{\pi r^2}. \tag{6.2}$$

Als we dit substitueren in $S$ krijgen we een formule voor de oppervlakte enkel in functie van de straal van het grondvlak $r$:

$$S(r) = 2\pi r^2 + \frac{2V}{r}.$$

Het is deze functie die we moeten minimaliseren. We berekenen de afgeleide en stellen deze gelijk aan nul:

$$\begin{aligned} S'(r) = 0 &\iff 4\pi r - 2\frac{V}{r^2} = 0 \\ &\iff V = 2\pi r^3. \end{aligned}$$

Hieruit volgt dat $r = \sqrt[3]{V/(2\pi)}$.

Om $h$ te bepalen zou je deze waarde voor $r$ kunnen invullen in vergelijking (6.2), maar we kunnen ook formule (6.1) gebruiken en deze gelijkstellen aan $V = 2\pi r^3$. Dan vinden we

$$\pi r^2 h = 2\pi r^3 \iff h = 2r.$$

Voor het ideale blikje is de hoogte van het blikje dus gelijk aan de diameter van het grondvlak.

Opmerking: om zeker te zijn dat het gaat om een minimum zouden we de tweede afgeleide moeten bepalen en verifiëren dat deze positief is in het gevonden punt. Dit is inderdaad het geval.

### 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[1].

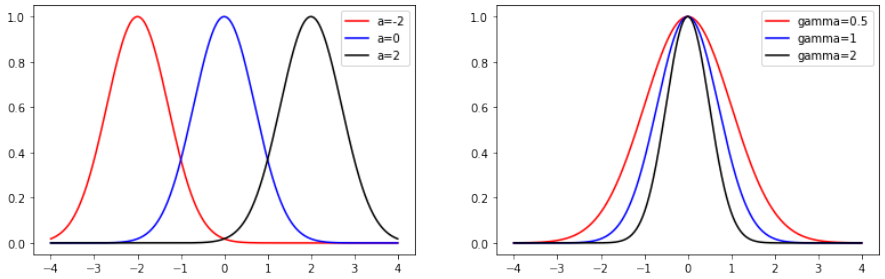
3. Beschouw functies van de vorm

$$f(x) = \exp\big(-\gamma(x-a)^2\big),$$

   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*[2].

   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?

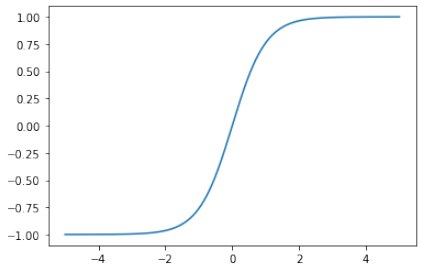
[1] Dit zal je zien in het opleidingsonderdeel Machine Learning
[2] Een methode van machine learning die je later zal bespreken.




**Figuur 6.1:** Invloed van parameters $a$ en $\gamma$ op de vorm van de functie $f(x) = \exp(-\gamma(x-a)^2)$.

### 6.6.7 Oplossingen

1. a) De tanh functie ziet er ongeveer zo uit



   b) Het domein van tanh is $\mathbb{R}$. Het beeld is het open interval $(-1, 1)$.

   c) De afgeleide functie van tanh berekenen we als volgt:

$$\begin{aligned}\tanh'(x) &= \frac{e^x + e^{-x}}{e^x + e^{-x}} - (e^x - e^{-x})\frac{e^x - e^{-x}}{(e^x + e^{-x})^2} \\ &= 1 - \frac{(e^x - e^{-x})^2}{(e^x + e^{-x})^2} \\ &= 1 - \tanh^2(x).\end{aligned}$$

2. TODO

3. a) Het domein van de functie is $\mathbb{R}$. Het beeld is het halfopen interval $(0, 1]$.

   b) Zie Figuur 6.1.

   c) We berekenen de afgeleide functie:

$$f'(x) = -2\gamma(x-a)\exp(-\gamma(x-a)^2).$$

De afgeleide is nul wanneer $x = a$. Uit de plot zien we duidelijk dat dit een maximum is.

d) Hoe groter $\gamma$ is, hoe smaller en meer gepiekt de belvorm is.

e) Om te weten waar de functie het snelst stijgt/daalt bepalen we de kritische punten van de afgeleide functie. Hiertoe berekenen we de afgeleide van de afgeleide:

$$f''(x) = \left(-2\gamma + 4\gamma^2(x-a)^2\right)\exp(-\gamma(x-a)^2)$$

en bijgevolg is

$$\begin{aligned} f''(x) = 0 &\iff -2\gamma + 4\gamma^2(x-a)^2 = 0 \\ &\iff (x-a)^2 = \frac{1}{2\gamma} \\ &\iff (x-a) = \pm\sqrt{\frac{1}{2\gamma}} \\ &\iff x = a \pm \sqrt{\frac{1}{2\gamma}}. \end{aligned}$$

M.a.w. als $\gamma$ groter is dan worden de meest extreme richtingscoëfficiënten dichter bij $a$ bereikt.

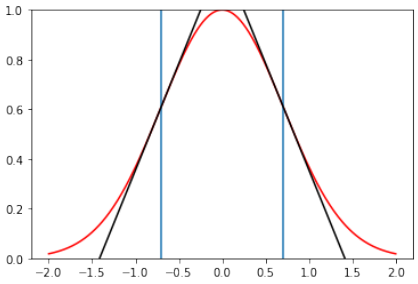
De waarde van de richtingscoëfficiënt vinden we door de gevonden $x$-waarden te substitueren in de afgeleide functie:

$$f'(a - \sqrt{\frac{1}{2\gamma}}) = \sqrt{2\gamma}\exp(-1/2)$$

en

$$f'(a + \sqrt{\frac{1}{2\gamma}}) = -\sqrt{2\gamma}\exp(-1/2)$$

We zien dat grotere waarden voor $\gamma$ leiden tot grotere (absolute) waarden van deze richtingscoëfficiënten. Tenslotte nog een plot om te verifiëren of de waarden correct (kunnen) zijn. Op deze plot tonen we de functie $f$ met $\gamma = 1$ en $a = 0$. We tonen de twee raaklijnen met maximale (absolute) richtingscoëfficiënt in het zwart en de $x$-coördinaten van de punten waar deze raaklijnen raken aan de functie worden aangeduid met twee blauwe verticale lijnen.




### 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-Rhapson 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.

### 6.7.2 Oplossingen

1. Zie de code in Figuur 6.2.

2. a) De volgende code maakt een plot waarmee je duidelijk kunt vaststellen dat er een nulpunt is tussen 3 en 4:

```python
import math
import numpy as np

def bereken_vierkantswortel(a, aantal_decimalen):
  epsilon = 10**(-aantal_decimalen)

  # Eerst iteratie vooraf om gemakkelijker te kunnen vergelijken
  x_prev = a
  x = (x_prev * x_prev + a) / (2*x_prev)
  while abs(x - x_prev) > epsilon:
    x_prev = x
    x = (x_prev * x_prev + a) / (2*x_prev)

  return x

if __name__ == "__main__":
  rng = np.random.default_rng()

  # 4 decimalen
  for _ in range(1000):
    a = rng.uniform(low=0.0, high=1000)
    v = bereken_vierkantswortel(a, 4)
    assert int(v * 10 ** 4) == int(math.sqrt(a) * 10 ** 4)
```

**Figuur 6.2:** Code voor vierkantswortel samen met een deel van de testen.

```python
import numpy as np
import matplotlib.pyplot as plt

f = lambda x : 2*x**2 + 5 - np.exp(x)
xs = np.linspace(0,4, 100)
ys = f(xs)
plt.plot(xs, ys)
plt.hlines(y=0, xmin=0, xmax=4, color='red')
plt.vlines(x=3, ymin=-5, ymax=5, color='black')
plt.vlines(x=4, ymin=-5, ymax=5, color='black');
\end{enumerate}
\end{enumerate}
```

b) Met de volgende code vind je dat het nulpunt gelijk is aan 3.275601.

```
x_prev = 3
x = 4
while abs(x_prev -x) > 10**(-6):
  x_prev = x
  x = x_prev - (2*x**2 + 5 - np.exp(x))/(4*x - np.exp(x))

print(f"Benadering voor het nulpunt is {x:.6f}")
```

3. a) De plot kan je op een gelijkaardige manier maken als in de vorige opgave.

   b) Uit de plot kan je afleiden dat er nulpunten liggen in de buurt van $-4$, 0 en 4. De nulpunten zelf zijn ongeveer $-3.442146$, 0.066392 en 4.375754.

HOOFDSTUK 7

# Reële functies in meerdere veranderlijken

### 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) $f(x,y,z) = \exp(-z)(x^2 y + 2)$

   e) $f(x,y,z) = \dfrac{\exp(z)}{\exp(x) + \exp(y) + \exp(z)}$.

### 7.2.2 Oplossingen

1. a)

$$\frac{\partial f}{\partial x}(x,y) = -2x \sin(x^2 + 2y)$$

en

$$\frac{\partial f}{\partial y}(x,y) = -2 \sin(x^2 + 2y)$$

b)

$$\frac{\partial f}{\partial x}(x,y) = 2x \exp(x^2 + y^2)$$

en

$$\frac{\partial f}{\partial y}(x,y) = 2y\exp(x^2+y^2)$$

c)

$$\frac{\partial f}{\partial s}(s,t,v) = \frac{t^2}{s+2t} - 3s^2\ln(3v),$$

$$\frac{\partial f}{\partial t}(s,t,v) = 2t\ln(s+2t) + 2\frac{t^2}{s+2t} - 2t\ln(3v),$$

en

$$\frac{\partial f}{\partial v}(s,t,v) = -\frac{1}{v}(s^3+t^2-4v) + 4\ln(3v).$$

d)

$$\frac{\partial f}{\partial x}(x,y,z) = \exp(-z)(2xy)$$

$$\frac{\partial f}{\partial y}(x,y,z) = \exp(-z)(x^2)$$

en

$$\frac{\partial f}{\partial z}(x,y,z) = -\exp(-z)(x^2y+2)$$

e)

$$\frac{\partial f}{\partial x}(x,y,z) = -\frac{\exp(z+x)}{(\exp(x)+\exp(y)+\exp(z))^2}$$

$$\frac{\partial f}{\partial y}(x,y,z) = -\frac{\exp(z+y)}{(\exp(x)+\exp(y)+\exp(z))^2}$$

en

$$\begin{aligned}\frac{\partial f}{\partial z}(x,y,z) &= \frac{\exp(z)}{\exp(x)+\exp(y)+\exp(z)} - \frac{\exp(2z)}{(\exp(x)+\exp(y)+\exp(z))^2} \\ &= \frac{\exp(z+x)+\exp(z+y)}{(\exp(x)+\exp(y)+\exp(z))^2}\end{aligned}$$

### 7.4.2 Oefeningen

1. Bekijk opnieuw de functie uit Voorbeeld **??**.

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 $\alpha$?

### 7.4.3 Oplossingen

1. a) De code in Figuur 7.1 geeft als resultaat

```
Update 1 geeft [0.99998 0.99996]
Update 2 geeft [0.99996 0.99992]
Update 10 geeft [0.99980002 0.99960007]
Update 100 geeft [0.99800198 0.99600791]
Update 1000 geeft [0.98019848 0.96078867]
Update 10000 geeft [0.81872912 0.67031468]
Laatste punt [0.81872912 0.67031468]
```

```python
import numpy as np

# De gradiënt
g = lambda x : np.array([2*x[0], 4*x[1]])


start = np.array([1,1])
alpha = 0.00001
x = start
for i in range(10000):
        x = x - alpha * g(x)
        if i in (0,1,9,99,999,9999):
                print(f"Update {i+1} geeft {x}")
                print(f"Laatste punt {x}")
```

**Figuur 7.1:** Voorbeeldcode gradient descent.

b) Dezelfde code maar met $\alpha = 1$ geeft als uitvoer:

```
Update 1 geeft [-1 -3]
Update 2 geeft [1 9]
Update 10 geeft [    1 59049]
Update 100 geeft [         1 -818408495]
Update 1000 geeft [          1 -742892767]
Update 10000 geeft [           1 -1391131839]
Laatste punt [           1 -1391131839]
```

samen met een waarschuwing voor overflow.

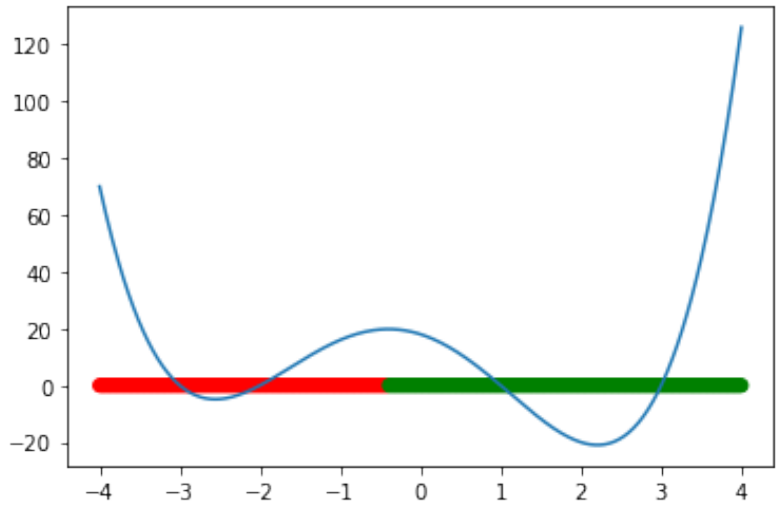
2. a) Telkens één factor afleiden en de andere laten staan leidt tot:

$$f'(x) = (x-1)(x+2)(x+3) + (x-3)(x+2)(x+3) \\ + (x-3)(x-1)(x+3) + (x-3)(x-1)(x+2).$$

b) De grafiek van de functie ziet er ongeveer uit als in Figuur 7.2 De functie heeft twee minima: een globaal minimum iets groter dan 2 en een lokaal minimum tussen $-3$ en $-2$.

c) Voor $\alpha = 0.1$ zie je in Figuur 7.2 welke $x$-waarden convergeren naar het globale minimum (groen) en welke naar het lokale minimum (rood).

De stapgrootte kan niet al te groot worden gekozen. Bv. als je start in $x = 3$ dan lijkt $\alpha = 0.5$ al niet te werken: de waarden springen




**Figuur 7.2:** Grafiek van de functie $f(x) = (x-3)(x-1)(x+2)(x+3)$. Voor $\alpha = 0.1$ convergeren de groene punten naar het globale minimum, de rode punten convergeren naar het lokale minimum.

weg en weer tussen ongeveer 2.5 en 1.4. Voor $\alpha = 1.0$ blazen de waarden op wanneer men start in $x = 3$.

### 7.5.1 Oefeningen

1. Veronderstel dat $g\colon \mathbb{R} \to \mathbb{R}^n$ en $h\colon \mathbb{R}^n \to \mathbb{R}$ functies zijn, en dat

$$f\colon \mathbb{R} \to \mathbb{R} : t \mapsto f(t) = h(g(t))$$

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.

### 7.5.2 Oplossingen

1. a) $-15$
   b) 18
2. a) $x'(t) = 2$ en $y'(t) = 1$.
   b) $\frac{\partial f}{\partial x} = 2xy$ en $\frac{\partial f}{\partial y} = x^2$.
   c)
   $$\frac{df}{dt} = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt} = 2xy \times 2 + x^2 = 2(2t)t \times 2 + 4t^2 = 12t^2.$$
   Controle door eerst $f(t)$ expliciet te berekenen:
   $$f(t) = 4t^2 \times t = 4t^3$$
   en inderdaad
   $$f'(t) = 12t^2.$$
3. a) $x'(t) = -\sin(t)$ en $y'(t) = \cos(t)$.
   b)
   $$\frac{\partial f}{\partial x} = \frac{1}{xy}y = \frac{1}{x}$$
   en
   $$\frac{\partial f}{\partial y} = \frac{1}{xy}x = \frac{1}{y}$$

c)

$$\frac{df}{dt} = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt} = \frac{1}{x}(-\sin(t)) + \frac{1}{y}\cos(t) = -\frac{\sin(t)}{\cos(t)} + \frac{\cos(t)}{\sin(t)}.$$

### 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:

```python
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 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. Ver-

geet 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 opgekeerde 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

$$\frac{\mathrm{d}\,b}{\mathrm{d}\,a},$$

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__.

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 **??**.

```
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 (**??**).

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

$$\tanh'(x) = 1 - \big(\tanh(x)\big)^2$$

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.

### 7.6.2 Oplossingen

1. a) We hebben de volgende knopen in de berekeningsgraaf:
   - $n_1 = x$
   - $n_2 = y$
   - $n_3 = n_1 n_2$ met $\frac{\partial n_3}{\partial n_1} = n_2$ en $\frac{\partial n_3}{\partial n_2} = n_1$
   - $n_4 = \exp(n_3)$ met $\frac{\partial n_4}{\partial n_3} = \exp(n_3)$
   - $n_5 = n_3 + n_4$ met $\frac{\partial n_5}{\partial n_3} = 1$ en $\frac{\partial n_5}{\partial n_4} = 1$.

   b) De voorwaartste berekening levert de volgende tabel:

| knoop | waarde | bewaarde afgeleiden |
|---|---|---|
| $n_1$ | $-1$ | |
| $n_2$ | $2$ | |
| $n_3$ | $n_3 = -2$ | $\frac{\partial n_3}{\partial n_1} = 2$ en $\frac{\partial n_3}{\partial n_2} = -1$ |
| $n_4$ | $n_4 = \exp(-2)$ | $\frac{\partial n_4}{\partial n_3} = \exp(-2)$ |
| $n_5$ | $n_5 = -2 + \exp(-2)$ | $\frac{\partial n_5}{\partial n_3} = 1$ en $\frac{\partial n_5}{\partial n_4} = 1$ |

   De laatste rij geeft de functiewaarde in de kolom "waarde".

   c) De achterwaartse berekening levert de volgende tabel.

| knoop | berekeningen en boodschappen |
|---|---|
| $n_5$ | $\frac{\partial n_5}{\partial n_3} = 1$ naar $n_3$ en $\frac{\partial n_5}{\partial n_4} = 1$ naar $n_4$. |
| $n_4$ | $\frac{\partial n_5}{\partial n_3} = \frac{\partial n_5}{\partial n_4}\frac{\partial n_4}{\partial n_3} = 1 \times \exp(-2)$ naar $n_3$. |
| $n_3$ | Bepaal $\frac{\partial n_5}{\partial n_3} = 1 + \exp(-2)$ (som van boodschappen)<br>$\frac{\partial n_5}{\partial n_1} = \frac{\partial n_5}{\partial n_3}\frac{\partial n_3}{\partial n_1} = (1 + \exp(-2)) \times 2$ naar $n_1$<br>$\frac{\partial n_5}{\partial n_2} = \frac{\partial n_5}{\partial n_3}\frac{\partial n_3}{\partial n_2} = (1 + \exp(-2)) \times (-1)$ naar $n_2$ |
| $n_2$ | $\frac{\partial n_5}{\partial n_2} = (1 + \exp(-2)) \times (-1)$ |
| $n_1$ | $\frac{\partial n_5}{\partial n_1} = (1 + \exp(-2)) \times 2$ |

We vinden dus finaal dus

$$\nabla f(-1,2) = \begin{bmatrix} (1+\exp(-2)) \times 2 \\ (1+\exp(-2)) \times (-1) \end{bmatrix}.$$

In dit geval is de functie die werd berekend zeer eenvoudig en kunnen de gradiënt volledig analytisch bepalen:

$$\nabla f(x,y) = \begin{bmatrix} \frac{\partial f}{\partial x}(x,y) \\ \frac{\partial f}{\partial y}(x,y) \end{bmatrix} = \begin{bmatrix} y + \exp(xy)y \\ x + \exp(xy)x \end{bmatrix}$$

Als we deze gradiënt evalueren in het punt $(-1,2)$ dan krijgen we uitdrukking terug die werd gevonden m.b.v. de berekeningsgraaf.

2. Hieronder vind je uitgewerkte Python-klasse `Value` voor deze oefening.

```python
class Value:

    def __init__(self, data, children=(), op='', label=''):
        self.data = data
        self.grad = 0.0
        # default: does nothing (e.g. leaf node)
        self._backward = lambda: None
        self._prev = children
        self._op = op
        self.label = label

    def __repr__(self):
        return f"Value(data={self.data})"

    def __add__(self, other):
        other = other if isinstance(other, Value)
                      else Value(other)
        out = Value(self.data + other.data,
                    children=(self, other), op='+')

        def _backward():
            self.grad  += 1.0 * out.grad
            other.grad += 1.0 * out.grad

        out._backward = _backward
        return out

    def __neg__(self):
        return self * (-1)

    def __sub__(self, other):
        return self + (-other)

    def __mul__(self, other):
        other = other if isinstance(other, Value)
                      else Value(other)

        out = Value(self.data * other.data,
                    children=(self, other), op='*')

        def _backward():
            self.grad  += other.data * out.grad
            other.grad += self.data * out.grad

        out._backward = _backward
        return out
```

```
# Vervolgcode. Nog steeds in Value

    def __rmul__(self, other):
        return self * other

    """
    def __truediv__(self, other):
        other = other if isinstance(other, Value)
                      else Value(other)

        out = Value(self.data / other.data,
                    children=(self, other), op='/')

        def _backward():
            self.grad  += 1/other.data * out.grad
            other.grad += (-self.data / other.data**2) * out.grad

        out._backward = _backward
        return out
    """


    # In the video Andrej does it like this
    def __truediv__(self, other):
        return self * other**(-1)


    def __pow__(self, other):
        assert isinstance(other, (int, float)),
               "only supporting int/float powers"

        out = Value(self.data ** other,
                    children=(self, ), op=f"**{other}")

        def _backward():
            self.grad += other * self.data ** (other -1) * out.grad

        out._backward = _backward

        return out
```

```
    def tanh(self):
        x = self.data
        t = (math.exp(2*x) - 1)/(math.exp(2*x) + 1)
        out = Value(t, children=(self, ), op='tanh')

        def _backward():
            self.grad += (1-t**2) * out.grad

        out._backward = _backward

        return out

    def exp(self):
        e = math.exp(self.data)
        out = Value(e, children=(self,), op='exp')

        def _backward():
            self.grad += e * out.grad

        out._backward = _backward
        return out

    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)

        build_topo(self)

        self.grad = 1.0
        for node in reversed(topo):
            node._backward()
```