Programmeringsoppgaver MAT1060

Richard Pedersen Patrono og Vasin Phumimas

Sommer 2020

Introduksjon

Dette dokumentet er et sett med programmeringsoppgaver utviklet av Richard Pedersen Patrono og Vasin Phumimas sommeren 2020, med tilegg av Simen Kvaal. Alle oppgavene tar utgangspunkt i Python som verktøy, og NumPy er hovedbiblioteket for løsning av oppgavene. Biblioteket SymPy kan alternativt brukes i noen sammenhenger. Fordelen med SymPy er at den tillater studenten å bruke symbolske beregninger, mens NumPy er hovedsakelig et numerisk bibliotek.

En har forsøkt å sette vanskelighetsgrad på alle oppgavene, angitt av antall stjerner ($\star$).

Kapittel 1

Elementær matriseregning

Oppgave 0: oppvarming ($\star$)

I NumPy kan vi sette opp en vektorer og matriser med `numpy.array`. Matrise-matrise- og matrise-vektorprodukt kan utføres med `@`-operatoren, eller alternativt `numpy.dot`. Sistnevnte kan også regne ut indreprodukter. Her er eksempler:

```
import numpy as np

# Definer en 3-vektor
x = np.array([0.2, -0.2, 5])

# Definer en 3x3-matrise
A = np.array([[1.0, 3.0, 0.0], [0.0, 1.0, -2.0], [0.0, -1.0, 0.0]])

# Regn ut matrise-vektorprodukt
y1 = A @ x
y2 = np.dot(A,x)
indre = np.dot(x,y1) # indreprodukt
```

0a) La $\mathbf{x}$ og $\mathbf{y}$ være gitt ved

$$\mathbf{x} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}.$$

Regn ut for hånd hva vinkelen mellom de to vektorene er. Skriv en funksjon som bruker NumPy-funksjonen `numpy.dot` og `math.acos` til å regne ut vinkelen mellom to *vilkårlige* vektorer $\mathbf{x}$ og $\mathbf{y}$ av samme lengde n. Test funksjonen på de gitte vektorene.

0b) Skriv en ny funksjon for beregning av vinkler, men *uten* NumPy-funksjoner.

Oppgave 1 ($\star$)

Vi skal se på hvordan vi kan bruke Python til å regne ut matriseprodukt for oss. Vi starter med matrisene:

$$M = \begin{pmatrix} 1 & 2 & 1 \\ 6 & -1 & 0 \\ -1 & -2 & -1 \end{pmatrix} \text{ og } P = \begin{pmatrix} -1 & -2 & -1 \\ -6 & -3 & 2 \\ 13 & 2 & 1 \end{pmatrix},$$

og vi ønsker å finne en matrise D slik at vi har

$$M = PDP^{-1}. \tag{1.1}$$

Lag et program i Python som finner matrisen D. (HINT: Skriv om likning (1.1). I NumPy er funksjonen `linalg.inv` nyttig her.)

Oppgave 2: Kryssprodukt ($\star$)

I denne oppgaven skal vi se hvordan vi kan bruke Python til å regne kryssprodukt for oss:

2a) Gitt vektorene

$$\mathbf{x} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \text{ og } \mathbf{y} = \begin{pmatrix} -1 \\ 1 \\ 0 \end{pmatrix}$$

Skriv en funksjon som tar vektorene $\mathbf{x}$ og $\mathbf{y}$ som input og returnerer kryssproduktet.

La $\mathbf{a}$, $\mathbf{b}$ og $\mathbf{c}$ være tre vektorer i $\mathbb{R}^3$. Vi definerer et parallellpiped som den romlige figuren utspent av de tre vektorene,

$$P = \{a\mathbf{a} + b\mathbf{b} + c\mathbf{c} \mid a, b, c \in [0, 1]\}.$$

Volumet til parallellpipedet P er da gitt ved formelen:

$$V = |(\mathbf{a} \times \mathbf{b}) \cdot \mathbf{c}|.$$

2b) Lag en funksjon i Python som tar inn tre vektorer og returnerer volumet til parallellpipedet vektorene utspenner. Gjenbruk funksjonen fra oppgave **a)**.

2c) Det finnes en annen formel for å regne ut volumet til et parallellpiped som er utspent av tre vektorer

$$\mathbf{a} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix} \text{ og } \mathbf{c} = \begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix},$$

som er gitt ved formelen:

$$V = |\begin{vmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{vmatrix}|$$

Skriv en Python-funksjon som regner ut volumet V med denne alternative formelen.

2d) Bruk funksjonene du lagde i obbgave **b)** og **c)** til å regne ut volumet til parallellpipedet utspent av vektorene:

$$\begin{pmatrix}1\\0\\2\end{pmatrix}, \begin{pmatrix}3\\1\\1\end{pmatrix} \text{ og } \begin{pmatrix}1\\2\\1\end{pmatrix}.$$

Kontroller at du får samme svar med de to funksjonene.

Oppgave 3 ($\star$)
La A være en 4×4 matrise gitt ved

$$A = \begin{pmatrix}96 & 69 & 420 & 1\\44 & 96 & 69 & 420\\420 & 44 & 96 & 69\\1 & 420 & 44 & 96\end{pmatrix} \tag{1.2}$$

3a) Bruk NumPy-funksjonen `linalg.eig` til å finne egenverdier og egenvektorer til A (likning (1.2))

3b) Finn rangen til A ved hjelp av `linalg.matrix_rank` eller `rref` (som finnes i modulen sympy).

Gitt en vektor i $\mathbb{R}^4$

$$\mathbf{b} = \begin{pmatrix}1\\1\\2\\3\end{pmatrix}$$

3c) Skriv $\mathbf{b}$ som en lineærkombinasjon av egenvektorene til A (**Hint:** `linalg.solve` kan være nyttig).

Oppgave 4 ($\star$)
Gitt matrisene

$$A = \begin{pmatrix}1 & 1\\2 & 2\\0 & 0\end{pmatrix}, U = \frac{1}{\sqrt{20}}\begin{pmatrix}2 & -4 & 0\\4 & 2 & 0\\0 & 0 & \sqrt{20}\end{pmatrix} \text{ og } V = \frac{1}{\sqrt{2}}\begin{pmatrix}1 & -1\\1 & 1\end{pmatrix},$$

der U og V er ortogonale matriser.

4a) Regn ut U^{-1}, U^T, V^{-1} og V^T.

4b) Anta at matrisen A kan skrives som $A = U\Sigma V^T$. Finn Σ.

Det vi gjorde nå i oppgave 5 kalles for *singulærverdidekomposisjon*, og matrisen Σ som vi finner er en matrise som består av kvadratroten til egenverdiene til matrisen $A^T A$. Singulær verdi dekomposisjon har flere anvendelsesområder, som bildebehandling og lineær regresjon.

Oppgave 5: von Neumanns ekspansjon ($\star\star$)

La $A \in \mathbb{R}^{n\times n}$ være en kvadratisk matrise, og la I være identitetsmatrisen. Vi skal studere en rekkeutvikling for $(I+A)^{-1}$ oppkalt etter von Neumann.

5a) Anta at $I + A$ er invertibel. Vis at

$$(I+A)^{-1} = I - A(I+A)^{-1}.$$

5b) Utled en formell potensekke for $(I + A)^{-1}$ på formen

$$(I+A)^{-1} = I - A + A^2 - A^3 + \cdots .$$

Dette er *von Neumanns rekke*. Hva kan vi si om denne rekka i tilfellet $n = 1$? Det er et faktum at denne rekka er konvergent for $n > 1$ dersom matriseelementene til A er tilstrekkelig små.

5c) Finn et eksplisitt utrykk for $(I+A)^{-1}$ under antakelsen at A er *nullpotent*, dvs. $A^k = 0$ for et heltall k. Trenger vi nå å anta at matriseelementene til A er tilstrekkelig små?

5d) Bruk forrige resultat til å regne ut inversen til $\begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix}$ analytisk.

5e) Skriv en Python-funksjon som beregner von Neumanns rekke med m termer gitt en kvadratisk matrise A. Funksjonen skal også printe en sammenlikning med inversen beregnet med `numpy.linalg.inv`.

5f) La så A være en 10×10-matrise gitt ved

$$A_{ij} = \sin(|i-j|)/|i-j|,$$

dersom $i \neq j$ og $A_{ii} = 0$. Anvend funksjonen fra forrige oppgave på αA der $\alpha = 0.2$ og bruk $m = 100$ termer i rekken. Ser rekken ut til å konvergere? La så $\alpha = 1.0$. Ser rekken ut til å konvergere nå?

5g) La $n > 1$ være en vilkårlig dimensjon. La A være en kvadratisk matrise gitt ved $A_{ij} = 0$ dersom $i \geq j$, og $A_{ij} = \exp(j - i^2)$ ellers. A er en øvre triangulær matrise. Forklar at A er nullpotent.

5h) La A være som i forrige oppgave. Skriv et Python-program som regner ut den numerisk eksakte inversen til $I + A$ med von Neumanns rekke. Sammenlikne med resultatet `numpy.linalg.inv` produserer.

Oppgave 6: hyperplan ($\star\star$)

Et hyperplan H av dimensjon $n-1$ i $\mathbb{R}^n$ er gitt ved

$$H = \{\mathbf{x} \in \mathbb{R}^n \mid \mathbf{x}\cdot\mathbf{n} - a = 0\},$$

der $\mathbf{n} \in \mathbb{R}^n$, $\mathbf{n} \neq 0$ er *normal* til H og der $a \in \mathbb{R}$. I planet $\mathbb{R}^2$ er et hyperplan en linje, og kan alternativt til $\mathbf{n}$ og a parameteriseres med en *tangent* $\mathbf{t} \in \mathbb{R}^2$ og et *skift* $\mathbf{b} \in \mathbb{R}^2$,

$$H = \{s\mathbf{t} + \mathbf{b} \mid s \in \mathbb{R},\, \mathbf{b}, \mathbf{t} \in \mathbb{R}^2,\, \mathbf{t} \neq 0\}.$$

I planet kan rotasjon med en vinkel θ (målt i radianer) uttrykkes ved $\mathbf{y} = R(\theta)\mathbf{x}$, der

$$R(\theta) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix}.$$

6a) Gitt m hyperplan H_i, $i = 1, 2, \ldots m$, med normaler $\mathbf{n}_i$ og konstanter a_i. Forklar at mengden av punkter $\mathbf{x} \in \mathbb{R}^n$ som er element i *alle* hyperplanene er løsninger av et lineært likningssystem

$$N\mathbf{x} = \mathbf{a},$$

og spesifisér koeffisientmatrisen N og vektoren $\mathbf{a}$ som funksjoner av normalene og skiftene.

6b) Vi arbeider nå i $\mathbb{R}^2$. Gitt to linjer parameterisert med tangenter og skift,

$$\mathbf{t}_1 = (0.2, 0.5)^T, \quad \mathbf{b}_1 = (0, 0)^T,$$

$$\mathbf{t}_2 = (0.6, -0.1)^T, \quad \mathbf{b}_2 = (0, 1.2)^T.$$

Bruk `matplotlib` til å tegne de to linjene i samme aksesystem.

6c) Gitt en linje parameterisert med tangent $\mathbf{t}$ og skift $\mathbf{b}$, skriv en Python-funksjon som regner ut en normal $\mathbf{n}$ og en konstant a som representerer den samme linja. (HINT: Normalen kan beregnes ved å rotere tangenten.)

6d) Skriv en Python-funksjon som tar to normaler $\mathbf{n}_1$ og $\mathbf{n}_2$ samt to konstanter a_1 og a_2 og regner ut punktet i planet der de to linjene gitt ved disse parametrene krysser hverandre. Anvend funksjonen sammen med funksjonen fra forrige oppgave på linjene gitt i **b)**. Stemmer resultatet med plottet?