Fachhochschule Graubünden
University of Applied Sciences

Introduction to Data Science HS 2021

Aufgabenblatt 4: Lineare Algebra-Wiederholung und Tensoren

Die Bearbeitung der Aufgaben ist freiwillig; es erfolgt keine Bewertung.

Aufgabe 1

Gegeben sind folgende Vektoren

$$\vec{\mathbf{u}} = \begin{pmatrix} 3 \\ 4 \\ 0 \\ 1 \end{pmatrix}, \quad \vec{\mathbf{v}} = \begin{pmatrix} 1 \\ 0 \\ -1 \\ 2 \end{pmatrix}, \quad \vec{\mathbf{w}} = \begin{pmatrix} 4 \\ -3 \\ 4 \\ 0 \end{pmatrix}.$$

Zeigen Sie rechnerisch und mittels Python, welche Paarungen obiger Vektoren orthogonal sind, d.h. senkrecht aufeinander stehen.

– Lösung –

Anmerkung: Zwei Vektoren sind orthogonal, wenn ihr Skalarprodukt null ist.

$$\vec{\mathbf{u}}^T \cdot \vec{\mathbf{v}} = \begin{pmatrix} 3 & 4 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ -1 \\ 2 \end{pmatrix} = 3 \cdot 1 + 4 \cdot 0 + 0 \cdot (-1) + 1 \cdot 2 = 5$$

$$\vec{\mathbf{u}}^T \cdot \vec{\mathbf{w}} = \begin{pmatrix} 3 & 4 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ -3 \\ 4 \\ 0 \end{pmatrix} = 3 \cdot 4 + 4 \cdot (-3) + 0 \cdot 4 + 1 \cdot 0 = 0$$

$$\vec{\mathbf{v}}^T \cdot \vec{\mathbf{w}} = \begin{pmatrix} 1 & 0 & -1 & 2 \end{pmatrix} \cdot \begin{pmatrix} 4 \\ -3 \\ 4 \\ 0 \end{pmatrix} = 1 \cdot 4 + 0 \cdot (-3) + (-1) \cdot 4 + 2 \cdot 0 = 0$$

Demnach stehen die Vektorenpaare $\vec{\mathbf{u}}$, $\vec{\mathbf{w}}$ und $\vec{\mathbf{v}}$, $\vec{\mathbf{w}}$ aufeinander senkrecht.

In Python (mithilfe Paket numpy):

```
>>> u = [3, 4, 0, 1]
>>> v = [1, 0, -1, 2]
>>> w = [4, -3, 4, 0]
```

```
>>> np.dot(u,v)
5
>>> np.dot(u,w)
0
>>> np.dot(v,w)
0
```

Aufgabe 2

Gegeben ist folgende Matrix

$$\mathbf{A} = \begin{pmatrix} 3 & 0 & 4 & 1 \\ 2 & 1 & 3 & 2 \end{pmatrix}.$$

a) Bestimmen Sie die Transponierte $\mathbf{A}^T$ und prüfen Sie Ihr Ergebnis mittels Python.

– Lösung –

$$\mathbf{A}^T = \begin{pmatrix} 3 & 2 \\ 0 & 1 \\ 4 & 3 \\ 1 & 2 \end{pmatrix}$$

In Python (mithilfe Paket numpy):

```
>>> A = [[3, 0, 4, 1], [2, 1, 3, 2]]
>>> np.transpose(A)
[[3, 2],
 [0, 1],
 [4, 3],
 [1, 2]]
```

b) Welche Dimension benötigt ein Vektor $\vec{\mathbf{v}}$ für die Matrix-Vektor-Multiplikation $\mathbf{A} \cdot \vec{\mathbf{v}}$ und $\mathbf{A}^T \cdot \vec{\mathbf{v}}$ für vorbezeichnete Matrix $\mathbf{A}$? In welchem der beiden Fälle erfolgt eine Dimensionsreduktion bzw. eine Dimensionserhöhung?

– Lösung –

- Matrix $\mathbf{A}$ hat die Grösse 2×4, d.h. 2 Zeilen und 4 Spalten $\rightarrow$ demnach muss ein Vektor $\vec{\mathbf{v}}$ vier Einträge haben, d.h. $\vec{\mathbf{v}}$ ist 4-dimensional

  Im Fall der Matrix-Vektor-Multiplikation $\mathbf{A} \cdot \vec{\mathbf{v}} = \vec{\mathbf{w}}$ erfolgt damit eine Dimensionsreduktion, d.h. Vektor $\vec{\mathbf{w}}$ ist 2-dimensional.

- Matrix $\mathbf{A}^T$ hat die Grösse 4×2, d.h. 4 Zeilen und 2 Spalten → demnach muss ein Vektor $\vec{\mathbf{v}}$ zwei Einträge haben, d.h. $\vec{\mathbf{v}}$ ist 2-dimensional

  Im Fall der Matrix-Vektor-Multiplikation $\mathbf{A}^T \cdot \vec{\mathbf{v}} = \vec{\mathbf{w}}$ erfolgt damit eine Dimensionserhöhung, d.h. Vektor $\vec{\mathbf{w}}$ ist 4-dimensional.

Merke: Der zu multiplizierende Vektor $\vec{\mathbf{v}}$ benötigt genauso viele Einträge wie die Matrix Spalten hat, der Ergebnisvektor $\vec{\mathbf{w}}$ hat genauso viele Einträge wie die Matrix Zeilen hat.

c) Gegeben sind folgende Vektoren

$$\vec{\mathbf{u}} = \begin{pmatrix} 2 \\ 0 \\ 4 \\ 1 \end{pmatrix}, \quad \vec{\mathbf{v}} = \begin{pmatrix} 1 \\ 2 \end{pmatrix}.$$

Berechnen Sie mittels Python die beiden Matrix-Vektor-Multiplikationen $\mathbf{A} \cdot \vec{\mathbf{u}}$ und $\mathbf{A}^T \cdot \vec{\mathbf{v}}$.

[Zur Übung können Sie die beiden Matrix-Vektor-Multiplikationen auch selbst berechnen.]

– Lösung (mithilfe Paket numpy) –

```
>>> u = [2, 0, 4, 1]
>>> v = [1, 2]
>>> np.dot(A,u)
[23, 18]

>>> np.dot(np.transpose(A),v)
[7, 2, 10, 5]
```

Aufgabe 3

Gegeben sind folgende Vektoren

$$\vec{\mathbf{u}} = \begin{pmatrix} 2 \\ 4 \\ 1 \end{pmatrix}, \quad \vec{\mathbf{v}} = \begin{pmatrix} 3 \\ 1 \\ 3 \\ 2 \end{pmatrix}, \quad \vec{\mathbf{w}} = \begin{pmatrix} 5 \\ 2 \end{pmatrix}.$$

a) Berechnen Sie das dyadische Produkt $\vec{\mathbf{u}} \cdot \vec{\mathbf{v}}^T$ und kontrollieren Sie Ihr Ergebnis in Python auf <u>zwei unterschiedliche</u> Arten.

– Lösung –

$$\vec{\mathbf{u}} \cdot \vec{\mathbf{v}}^T = \begin{pmatrix} 2 \\ 4 \\ 1 \end{pmatrix} \cdot \begin{pmatrix} 3 & 1 & 3 & 2 \end{pmatrix} = \begin{pmatrix} 2\cdot 3 & 2\cdot 1 & 2\cdot 3 & 2\cdot 2 \\ 4\cdot 3 & 4\cdot 1 & 4\cdot 3 & 4\cdot 2 \\ 1\cdot 3 & 1\cdot 1 & 1\cdot 3 & 1\cdot 2 \end{pmatrix} = \begin{pmatrix} 6 & 2 & 6 & 4 \\ 12 & 4 & 12 & 8 \\ 3 & 1 & 3 & 2 \end{pmatrix}$$

In Python (mithilfe Paket numpy):

```
>>> u = [2, 4, 1]
>>> v = [3, 1, 3, 2]
>>> w = [5, 2]
>>> np.outer(u,v)
[[ 6,  2,  6,  4],
 [12,  4, 12,  8],
 [ 3,  1,  3,  2]]

>>> np.einsum('i,j->ij',u,v)
[[ 6,  2,  6,  4],
 [12,  4, 12,  8],
 [ 3,  1,  3,  2]]
```

b) Berechnen Sie den Tensor $\mathcal{X}$ in Python als äusseres Produkt der drei Vektoren $\vec{w} \circ \vec{u} \circ \vec{v}$.

– Lösung (mithilfe Paket numpy) –

```
>>> X = np.einsum('i,j,k->ijk',w,u,v)
>>> X
[[[30, 10, 30, 20],
  [60, 20, 60, 40],
  [15,  5, 15, 10]],

 [[12,  4, 12,  8],
  [24,  8, 24, 16],
  [ 6,  2,  6,  4]]]
```

Aufgabe 4

Gegeben sind folgende Matrizen

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix}.$$

Berechnen Sie in Python das Kronecker-Produkt $\mathbf{A} \otimes \mathbf{B}$, das Khatri-Rao-Produkt $\mathbf{A} \odot \mathbf{B}$ und das Hadamard-Produkt $\mathbf{A} * \mathbf{B}$. Nutzen Sie das Modul `tensorly` für Ihre Berechnung.

Tipp: Erzeugen Sie sich die Matrizen als Reihung mittels `np.arange( )` und ändern Sie deren Gestalt mittels `reshape((#zeilen,#spalten))`.

– Lösung –

Anmerkung: **`np.arange(n)`** erzeugt eine Reihung [0, 1, ..., n–1] mit den Zahlen 0 bis n–1. Um eine Reihung [1, 2, ..., n] zu erhalten, muss der Wert 1 auf die gesamte Reihung addiert werden → **`np.arange(n)+1`**

```
>>> import tensorly as tl
>>> A = np.arange(6).reshape((2,3))+1
>>> B = np.arange(6).reshape((2,3))+7
>>> tl.tenalg.kronecker((A,B))
[[ 7,  8,  9, 14, 16, 18, 21, 24, 27],
 [10, 11, 12, 20, 22, 24, 30, 33, 36],
 [28, 32, 36, 35, 40, 45, 42, 48, 54],
 [40, 44, 48, 50, 55, 60, 60, 66, 72]]

>>> tl.tenalg.khatri_rao((A,B))
[[ 7, 16, 27],
 [10, 22, 36],
 [28, 40, 54],
 [40, 55, 72]]

>>> A*B
[[ 7, 16, 27],
 [40, 55, 72]]
```