Fachhochschule Graubünden
University of Applied Sciences

Methods & Algorithms FS 2023

Aufgabenblatt 4: Lineare Algebra & lineare Diskriminanzanalyse

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 <u>und</u> 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 (mittels Paket `numpy`):

```
u = [3, 4, 0, 1]

v = [1, 0, -1, 2]

w = [4, -3, 4, 0]
```

```
np.dot(u,v)
np.dot(u,w)
np.dot(v,w)
```

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 (mittels Paket `numpy`):

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

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 → 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 Dimensions-reduktion, 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 (mittels Paket `numpy`) –

```
u = [2, 0, 4, 1]

v = [1, 2]

np.dot(A,u)

np.dot(np.transpose(A),v)
```

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 zwei unterschiedliche 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 (mittels Paket `numpy`):

```
u = [2, 4, 1]

v = [3, 1, 3, 2]

w = [5, 2]

np.outer(u,v)

np.einsum('i,j->ij',u,v)
```

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

– Lösung (mittels Paket **numpy**) –

```
X = np.einsum('i,j,k->ijk',w,u,v)
print(X)
```

Aufgabe 4

Der Brustkrebs-Datensatz aus dem Paket **sklearn.datasets** beinhaltet 569 Untersuchungsergebnisse, die nach 30 unterschiedlichen Merkmalen entweder als bösartig (M) oder gutartig (B) klassifiziert wurden. Laden Sie zunächst den Datensatz **cancer** wie folgt

```
from sklearn import datasets
cancer = datasets.load_breast_cancer()
```

der u. a. die Teile **cancer.data** (Grösse 569×30) mit den Merkmalswerten sowie **cancer.target** mit den korrekten Klassifikationen (0 oder 1) und **cancer.target_names** mit den korrekten Namen (malignant oder benign) enthält.

a) Teilen Sie den Datensatz im Verhältnis 90:10 in Trainings- und Testdaten auf und führen Sie anschliessend mittels linearer Diskriminanzanalyse (LDA) eine Reduktion des Merkmalraumes auf eine Dimension (d.h. $n\_components = 1$) durch.

– Lösung –

```
from sklearn.model_selection import train_test_split as tts
from sklearn.discriminant_analysis import
    LinearDiscriminantAnalysis as LDA

x, x_test, y_true, y_test = tts(cancer.data, cancer.target,
    test_size=0.1, random_state=42)
model = LDA(n_components=1)
model.fit(x, y_true)
x_proj = model.transform(x)
x_test_proj = model.transform(x_test)
```

b) Trainieren Sie einen k-Nächste-Nachbarn-Algorithmus für $k \in [1, 10]$ anhand der projizierten Trainingsdaten und führen Sie dann mittels der projizierten Testdaten eine Vorhersage (Klassifikation) durch. Für welche k-Werte lässt sich die beste Treffsicherheit erzielen?

– Lösung –

```
from sklearn.neighbors import KNeighborsClassifier as knn
from sklearn.metrics import accuracy_score

hits = [0]
for k in range(1,11):
  model = knn(k)
  model.fit(x_proj, y_true)
  y_pred = model.predict(x_test_proj)
  hits.append(accuracy_score(y_test, y_pred))
```

c) Visualisieren Sie Ihre Ergebnisse aus b) als Treppenkurve (`plt.step`) in einem Diagramm inklusive Beschriftung und Legende. Zusatzfrage: Wie viele bösartigen Fälle wurden als gutartig und umgekehrt klassifiziert?

– Lösung –

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

x_grid = np.linspace(0, 10, 11)
plt.step(x_grid, hits, where='mid', color='red', linewidth=2)
plt.show()
```

Zusatzfrage: Wie wurde falsch klassifiziert?

```
for i in range(len(y_test)):
  if y_test[i] != y_pred[i]:
    print(cancer.target_names[y_test[i]], '=>',
      cancer.target_names[y_pred[i]])
```

Für $k = 5$ wurden insgesamt vier Fälle falsch klassifiziert, davon drei gutartige Fälle als bösartig (benign $\Rightarrow$ malignant) und ein bösartiger Fall als gutartig (malignant $\Rightarrow$ benign).