Fachhochschule Graubünden
University of Applied Sciences

Methods & Algorithms FS 2023

Aufgabenblatt 5: Tensorfaktorisierung

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

Aufgabe 1

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 mittels **`np.arange( )`** und ändern Sie deren Gestalt mittels **`reshape((#zeilen,#spalten))`**.

– Lösung –

```
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))
tl.tenalg.khatri_rao((A,B))
A*B
```

Aufgabe 2

Erzeugen Sie sich einen 3-Mode-Tensor $\mathcal{X}$ der Grösse $5 \times 2 \times 5$ mit den Zahlen von 1 bis 50.

```
X = np.arange(50).reshape((5,2,5))
X = X + 1
```

Entfalten Sie den Tensor mithilfe des Pakets **`tensorly`** nach allen drei Moden.

– Lösung –

```
import tensorly as tl
```

```
tl.unfold(X, 0)
tl.unfold(X, 1)
tl.unfold(X, 2)
```

Aufgabe 3

Erzeugen Sie sich einen 3-Mode-Tensor $\mathcal{X}$ der Grösse $30 \times 30 \times 30$ bestehend aus Zufallszahlen.

```
X = np.random.rand(30,30,30)
```

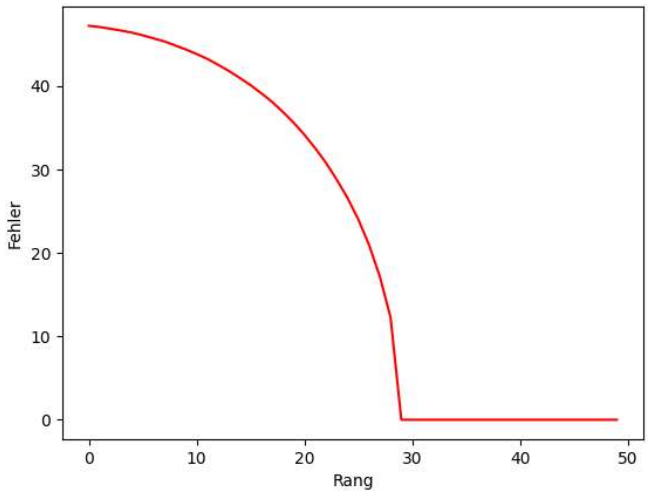
Führen Sie sukzessive eine Tucker-Zerlegung von $\mathcal{X}$ mit Rang $r \in [1, 50]$ gemäss

```
tucker(X, (r, r, r))
```

aus dem Paket **`tensorly`** durch, bestimmen Sie den jeweiligen Fehler des rekonstruierten Tensors und zeichnen Sie die Fehlerkurve. Was fällt auf?

– Lösung –

```
from tensorly.decomposition import tucker
err = []
for r in range(1,51):
  G, fac = tucker(X, (r, r, r))
  X_rec = tl.tucker_to_tensor((G, fac))
  err.append(tl.norm(X - X_rec))
plt.plot(err, 'r-')
plt.show()
```



Es fällt auf, dass der Fehler stetig abnimmt und etwa ab Rang 28 [Anmerkung: Da es sich um einen Zufallstensor handelt, kann Ihre Kurve leicht anders aussehen] bei Null liegt. Das heisst, etwa ab Rang 28 lässt sich keine sichtbare Genauigkeitssteigerung mehr erzielen, die Faktorisierung ist damit nahezu perfekt.

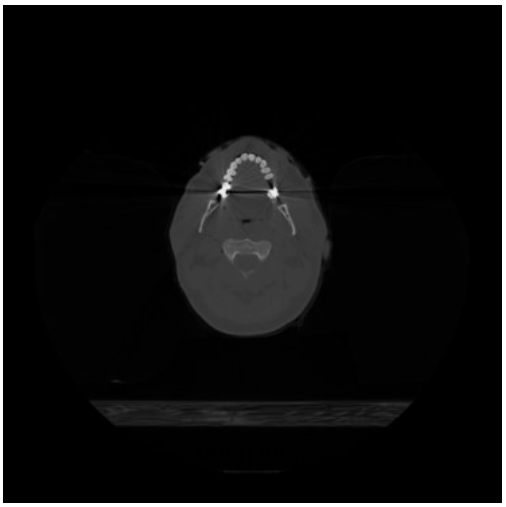
Aufgabe 4

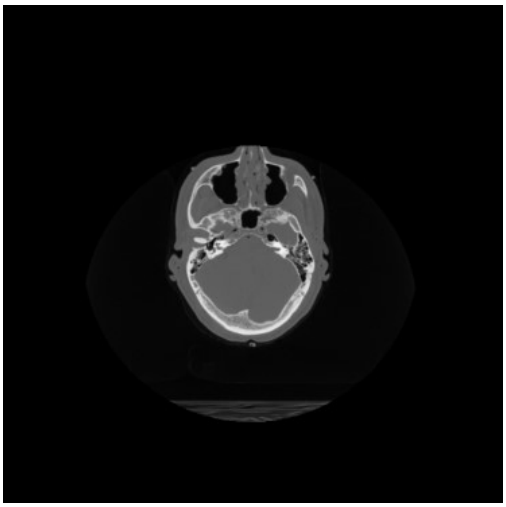
Die Datei **`MRI.npy`** enthält einen Tensor der Grösse $245 \times 512 \times 512$ mit 245 MRI-Schnittbildern (Grösse jeweils 512×512 Pixel) eines menschlichen Kopfes. Laden und entpacken Sie zunächst die Datei **`MRI.zip`** aus Moodle, anschliessend können Sie den Tensor in Python folgendermassen einlesen.

```
import numpy as np
X = np.load('MRI.npy')
```

Um einzelne Schnittbilder $i \in [0, 244]$ zu betrachten, können Sie diese mittels **`imshow`** aus dem Paket **`matplotlib.pyplot`** darstellen.

```
plt.imshow(X[i], cmap='gray')
plt.show()
```

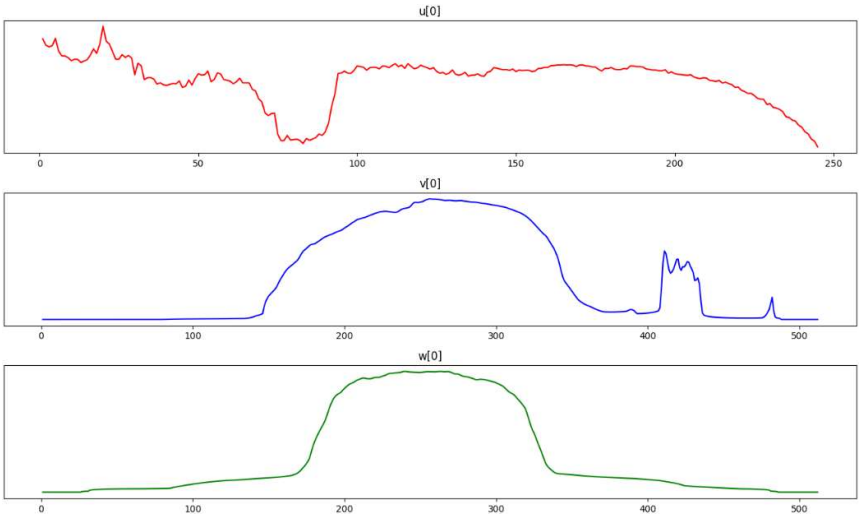




a) Führen Sie eine Rang-1-Zerlegung mittels **`parafac`** aus dem Paket **`tensorly`** durch und stellen Sie das Ergebnis (analog zur Vorlesung) mit der Funktion **`plot_uvw_one_feature`** aus dem Paket **`mytensor.py`** dar. Welche Aussagen lassen sich über den Tensor treffen?

– Lösung –

```
from tensorly.decomposition import parafac
from mytensor import plot_uvw_one_feature as plot1f
w_, fac = parafac(X, 1)
plot1f(fac)
```



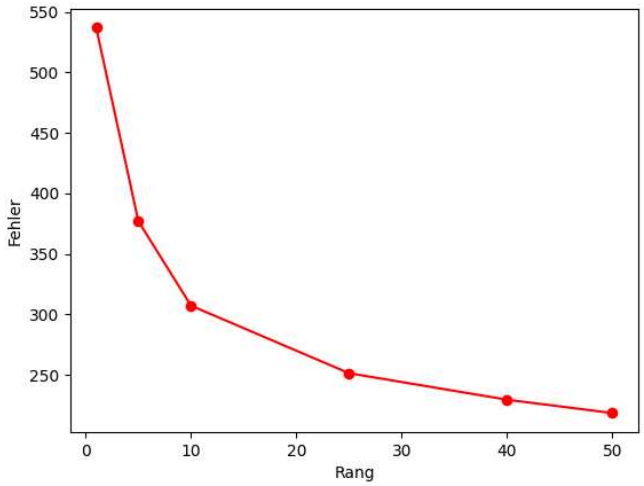
Anmerkung: Aufgrund der zufälligen Initialisierung der Tensor-Zerlegung können Ihre Kurven anders aussehen.

Vektor `u[0]` zeigt den Verlauf über alle 245 Schnittbilder; eine generelle Aussage ist daraus nicht ableitbar. Die Vektoren `v[0]` und `w[0]` zeigen den gemittelten Verlauf in x- und y-Richtung über alle Schnittbilder. Ausser, dass die relevante Information im Mittelpunkt der jeweiligen Bilder liegen muss, lässt sich auch hier keine generelle Aussage über den Tensor bzw. dessen Inhalt ableiten.

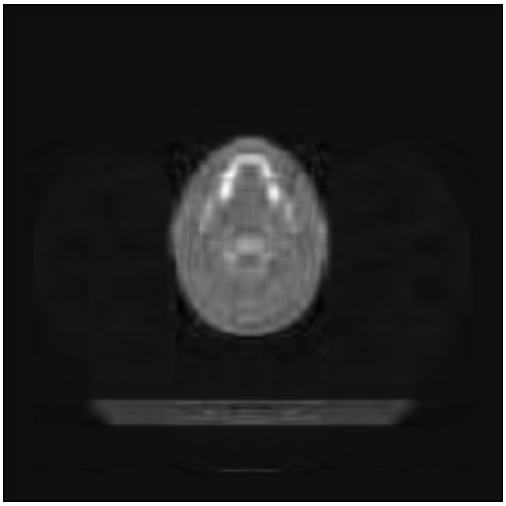
b) Führen Sie eine Tensor-Zerlegung mittels **`parafac`** mit Rang $r \in \{1, 5, 10, 25, 40, 50\}$ durch, bestimmen Sie den jeweiligen Fehler des rekonstruierten Tensors und zeichnen Sie die Fehlerkurve. Was fällt auf? [**Achtung**: Die Zerlegungen sind sehr aufwändig und können insgesamt bis zu einer Stunde oder länger dauern.]

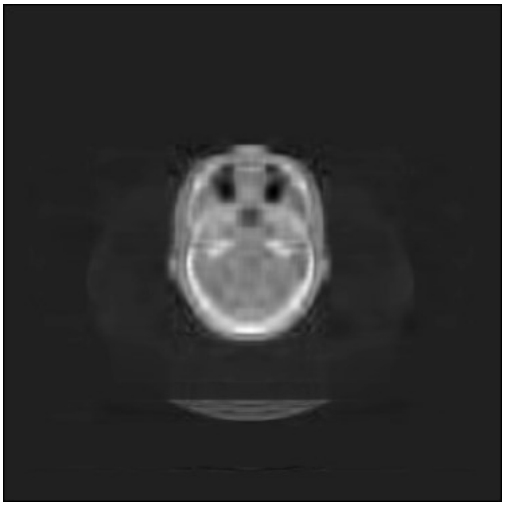
– Lösung –

```
import tensorly as tl
err = []
rank = [1, 5, 10, 25, 40, 50]
for r in rank:
  fac = parafac(X, r)
  X_rec = tl.kruskal_to_tensor(fac)
  err.append(tl.norm(X - X_rec))
plt.plot(err, 'r-')
plt.show()
```



Es fällt auf, dass der Fehler zwar stetig abnimmt, aber selbst bei Rang 50 noch immer sehr hoch ist. Dies lässt auf eine mindere Genauigkeit der rekonstruierten Schnittbilder schliessen. Eine visuelle Überprüfung mittels **`imshow`** (siehe oben) zeigt, dass es den Bildern an Detailtreue fehlt, d.h. hochfrequente Anteile wurden geglättet.





c) Wie hoch ist die Kompressionsrate des Tensors bei einer Rang-50-Zerlegung gegenüber den Originaldaten?

– Lösung –

Der Originaltensor hat eine Grösse von $245 \times 512 \times 512$ Elementen, d.h. insgesamt müssen $nt = 245*512*512 = 64.225.280$ Elemente gespeichert werden. Die drei Faktormatrizen haben Grössen von 245×50, 512×50 und 512×50, d.h. insgesamt müssen $nf = 245*50 + 512*50 + 512*50 = 63.450$ Elemente gespeichert werden. Damit ergibt sich eine Kompressionsrate von $nt / nf = 64.225.280 / 63.450 \approx 1012$, d.h. die Daten lassen sich um einen Faktor 1000 komprimieren.

Zur Veranschaulichung: jedes Element benötigt ein Byte $\Rightarrow$ 64 MByte vs. 64 KByte.