

NNW-Übung 1

Bemerkungen zu Python

Empfehlung für Python auf Ihrem eigenen Rechner: Python-Distribution *Anaconda*¹. Anaconda enthält die meisten benötigten Pakete und zudem die Entwicklungsumgebungen *Spyder* (Desktop-Applikation) und *Jupyter* (läuft im Webbrowser, starten mit `jupyter notebook`). Python ist ein Interpreter und erlaubt eine *interaktive Befehlseingabe*, was ein schnelles Ausprobieren erleichtert, z.B. bei Spyder in der „IPython-Shell“ rechts unten.

Auf der Moodle-Seite finden Sie Links zu Einführungen in Python. Ein auffälliger Unterschied zu Java/C ist, dass Blöcke *nicht* durch geschweifte Klammern, sondern durch *Einrückung* (indentation) gekennzeichnet werden. Empfohlen werden 4 Leerzeichen je Einrückungstiefe (Achtung: *nicht* mit Tabulatoren mischen). Semikolons werden normalerweise nicht gebraucht – nur bei mehr als einer Anweisung pro Zeile.

Hilfe zu Funktionen kann in Spyder durch Drücken von **Strg-I** aufgerufen werden (das funktioniert in der „IPython-Shell“ rechts unten teilweise besser als im Editorfenster) oder in der „IPython-Shell“ ein `?` hinter die Anweisung eintippen, z.B. `print?`.

1 Arbeiten mit NumPy

NumPy ist eine Erweiterung von Python für das Rechnen mit Arrays (Matrizen).²

Probieren Sie die folgenden Befehle *nacheinander* (!) aus und vollziehen Sie die Ergebnisse nach:

```
import numpy as np # numpy heißt jetzt np ... spart Tipparbeit!
```

```
W=np.array([[1.,2,3],[3,4,5]])
```

```
W.shape #Zeilen, Spalten
```

```
W.shape[0]
```

```
W.shape[1]
```

```
W.T
```

```
W.T.shape # shape ist keine Funktion, sondern eine Eigenschaft
```

```
type(W)
```

```
type(W.shape)
```

```
# Aufruf der NumPy-Funktion arange:
```

```
r=np.arange(10)
```

```
print(r)
```

```
r.shape
```

```
M=np.arange(12).reshape(3,4)
```

```
M
```

```
M.shape
```

```
M[2,0]
```

```
M[1,2]
```

¹www.anaconda.com/products/individual

²Falls Sie Erfahrungen mit MATLAB haben: Einige Konstrukte sind ähnlich, es gibt aber auch einige deutliche Unterschiede. Auf der Moodle-Seite ist ein Link zu einer Seite mit einer Gegenüberstellung von MATLAB und NumPy.

```

M[1,:]
M[1]
M[:,1] # ergibt Zeile statt Spalte
M[:,[1]] # ergibt eine Spalte
M[:,[3,0,1,1]]
M[:,2:4] # Spalte 2 bis exklusive Spalte 4
M[-2,:] # 2te Zeile von hinten
M[-2:,:] # 2te Zeile von hinten bis Ende
M[:,2]=2 # ändern ist möglich

M>4
M[M>4]
M[M>4]=-17
M

```

Beachten Sie bei dem nun folgenden Code:

- Die Einrückung (indentation) der Zeilen zur Definition von Blöcken (siehe oben)!
- Benannte Parameter: Optionale Parameter *können* durch Angabe ihres Bezeichners gesetzt werden (statt über die Position des Parameters), im Beispiel bei `print` z.B. `sep=''`.
Was diese Parameter bedeuten, verrät Spyder Ihnen, wenn Sie **Strg-I** drücken.

```

s=0
for i in range(M.shape[0]):
    for j in range(M.shape[1]):
        s+=M[i,j]
        print(s, ',', sep='', end='') # Parameter sep und end über Namen setzen
print(s)

def func(x):
    print(x*x)
    return x

func(3)

```

2 Laden und Plotten des Iris-Blüten Datensatzes

Der Iris-Blüten Datensatz ist ein bekannter Datensatz zum Einstieg in maschinelle Lernverfahren.³ Er enthält für drei Arten von Iris-Blüten jeweils mehrere Beobachtungen (samples) von vier Merkmalen (features): jeweils die Breite und Länge der zwei unterschiedlichen Blütenblättern (also $2 \times 2 = 4$).

In der Vorlesung wird (soweit nicht anders angegeben) folgende Konvention verwendet:

- Die Merkmale $x_1, x_2, \dots$ eines Objekts werden in *Spalten*vektoren $\vec{x}$ gespeichert. Die Merkmale stehen also in den *Zeilen*.
- Haben wir viele Objekte, z.B. N Trainingsbeispiele, so speichern wir diese Daten in einer Matrix X : je *Trainingsbeispiel* eine *Spalte*! Das ist nichts anderes, als das Nebeneinanderschreiben der Merkmalsvektoren (Spalten) der einzelnen Objekte!

³Der Iris-Blüten Datensatz wird am Ende von Kapitel 2.1 kurz vorgestellt, weitere Informationen z.B. auf Wikipedia: en.wikipedia.org/wiki/Iris_flower_data_set

- Die Zielwerte bei supervised learning sollen entsprechend in einer Matrix T (für target) stehen.

Auf der Moodle-Seite finden Sie die Iris-Daten als „comma-separated-values“ (`.csv`).

- Schauen Sie sich den Inhalt der Datei an, zum Beispiel mit einem Text-Editor (wie `nano` (Unix) oder Notepad++ (Windows)). Gespeichert sind die vier Merkmale und die Iris-Art (kodiert als Zahl von 0 bis 2).

Wie viele Daten gibt es je Iris-Art?

- Laden Sie die Daten in Python mit Hilfe der NumPy-Funktion `loadtxt`. Die Beschreibung der Parameter der Funktion und des Rückgabewerts finden Sie im Internet oder durch Drücken von **Strg-I** in Spyder.

Tipp: Sie brauchen (auch) den Parameter `delimiter`.

- Zerlegen Sie die eingelesenen Daten in die Matrizen X und T .
- Stehen die Daten je Blüte in den Zeilen oder Spalten? Passen Sie die Daten ggf. der Konvention (siehe oben) an.
- Plotten Sie die *ersten beiden Merkmale* der Daten mit der Funktion `scatter` aus dem Modul `matplotlib.pyplot`, das Sie daher zuerst importieren müssen:

```
import matplotlib.pyplot as plt
```

- Färben Sie die drei Iris Arten durch Übergabe der folgenden zusätzlichen Parameter von `scatter` ein: `c=T`, `cmap=plt.cm.prism` (unter der Annahme, dass `matplotlib.pyplot` als `plt` importiert wurde).

Alternativ können Sie auch selbst eine Color-Map definieren:

```
cmap=colors.ListedColormap(['red', 'green', 'blue'], wobei Sie vorher colors importieren müssen: from matplotlib import colors
```

3 Interaktive 3D Scatter Plots in Python

Möchte man drei Merkmale visuell darstellen, können 3D Scatter Plots nützlich sein. Da der Bildschirm nur 2D ist, ist es hilfreich, wenn diese interaktiv gedreht werden können, damit man die 3D Verteilung der Punkte von verschiedenen Seiten betrachten kann.

Eine Möglichkeit interaktive 3D Scatter Plots in Python zu erzeugen ist das Paket `plotly` (plotly.com/python/).⁴

- Die Erstellung von 3D Scatter Plots mit `plotly` wird auf der folgenden Webseite erklärt – anhand des Iris-Blüten Datensatzes: plotly.com/python/3d-scatter-plots/.
- Probieren Sie `plotly` in Ihrem eigenen Programm aus der vorigen Aufgabe aus: Lassen Sie die ersten drei Merkmale anzeigen und je nach Iris-Art einfärben.

Achtung: Je nach Entwicklungsumgebung kann es sein, das erstmal nichts angezeigt wird. Abhilfe schaffen kann das Folgende *vor* dem ersten Aufruf von `fig.show()` (oder Sie gehen zur nächsten Teilaufgabe über):

```
import plotly.io as pio
pio.renderers.default='browser'
```

⁴`plotly` gibt es für verschiedene Sprachen, die Python-Variante wird manchmal auch `plotly.py` genannt.

- c) Sie können das Plot auch in einer `html`-Datei abspeichern und die gespeicherte Datei mit einem Browser öffnen. Statt `fig.show()` verwenden Sie dafür `fig.write_html('myfigure.html')`. Mit dem zusätzlichen Parameter `auto_open` sollte automatisch ein Browser mit der Datei geöffnet werden – das funktioniert aber nicht bei jeder Installation.
- d) Lassen Sie Merkmale zwei bis vier (bzw. ab 0 gezählt 1 bis 3) anzeigen. Falls Sie die Variante mit dem Speichern als `html` verwenden, müssen Sie im Browser auf „neu laden“ klicken, damit sich das Diagramm aktualisiert.

4 Der Thyroid Disease Datensatz – ein etwas größerer Datensatz

Auf der Moodle-Seite befindet sich zudem eine Datei namens `thyroid-ann-train.data` mit Daten zur Schilddrüse bei verschiedenen Patientengruppen (letzte Spalte).⁵

- a) Wie viele Merkmale enthält der Datensatz?
- b) Wie viele der Merkmale sind binär und wie viele kontinuierlich?
- c) Wie viele verschiedene Patientengruppen gibt es? Eine nützliche Python-Funktion dafür ist `numpy.unique`.
- d) Wie häufig kommen die verschiedenen Patientengruppen vor?
Auch dies können Sie mit `numpy.unique` zählen lassen, schauen Sie sich die optionalen Parameter an.
- e) Plotten Sie mit `plotly` die folgenden Merkmalskombinationen in verschiedene Diagramme, jeweils eingefärbt nach Patientengruppe:
 - i) Merkmal 0, 16, 17
 - ii) Merkmal 0, 17, 18
 - iii) Merkmal 18, 19, 20
 - iv) Merkmal 16, 18, 20

Die Punkte sind in der Standardeinstellung für die vielen Daten zu groß. Dies können Sie zum Beispiel ändern mit⁶: `fig.update_traces(marker={'size': 3})`

- f) Welche der Kombinationen aus drei Merkmalen ist vermutlich besser und welche schlechter zur Erkennung (Klassifikation) der Patientengruppe geeignet?
- g) Würden Sie vermuten, dass bei einer oder mehreren der Kombinationen eines der drei Merkmale überflüssig ist, also – vermutlich – keinen weiteren Vorteil für die Klassifikation haben wird?

⁵Quelle: archive.ics.uci.edu/dataset/102/thyroid+disease. Die Original-Datei heißt `ann-train.data`. Damit das Einlesen problemlos funktioniert, wurden die zwei Leerzeichen, die sich am Ende jeder Zeile der Original-Datei befinden, gelöscht.

⁶In Python wird mit `{key : value}` ein `dict` (für dictionary) angelegt.