KNN

June 22, 2021

1 Künstliche Neuronale Netze

2 mit Tensorflow / Keras

2.1 Beispiel 1: XOR-Verknüpfung

Mit Hilfe eines KNNs soll die XOR-Verknüpfung nachgebildet werden:

X1	X2	y
0	0	0
0	1	1
1	0	1
1	1	0

[1]:
```
import numpy as np
from tensorflow.keras.optimizers import *
from tensorflow.keras.models import *
from tensorflow.keras.layers import *
from tensorflow.keras.losses import *
from tensorflow.keras.utils import *
```

[2]:
```
# Daten
X = [[0,0],[0,1],[1,0],[1,1] ]
y = [[0],  [1],  [1],  [0]]
```

[3]:
```
model = Sequential()
model.add(Dense(8, input_dim=2)) # Input-Layer
model.add(Activation("relu"))

model.add(Dense(10)) # Hidden-Layer
model.add(Activation("tanh"))

model.add(Dense(10)) # Hidden-Layer
model.add(Activation("tanh"))

model.add(Dense(1)) # Output-Layer
model.add(Activation("sigmoid"))
```

```python
sgd = SGD(lr=0.1) # Stochastic Gradient Descent
model.compile(loss="binary_crossentropy", optimizer=sgd, metrics="accuracy")

# Für die Ausgabe der Trainingsinformationen verbose auf 2 setzen
model.fit(X,y, epochs=100, verbose=0)
```

[3]: <tensorflow.python.keras.callbacks.History at 0x1c5f3cc51f0>

[4]:
```python
(model.predict(X)>0.5).astype(int)
```

[4]:
```
array([[0],
       [1],
       [1],
       [0]])
```

[5]:
```python
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 8)                 24
_________________________________________________________________
activation (Activation)      (None, 8)                 0
_________________________________________________________________
dense_1 (Dense)              (None, 10)                90
_________________________________________________________________
activation_1 (Activation)    (None, 10)                0
_________________________________________________________________
dense_2 (Dense)              (None, 10)                110
_________________________________________________________________
activation_2 (Activation)    (None, 10)                0
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 11
_________________________________________________________________
activation_3 (Activation)    (None, 1)                 0
=================================================================
Total params: 235
Trainable params: 235
Non-trainable params: 0
_________________________________________________________________
```

2.2 Beispiel 2: Vorhersage Brustkrebs

[6]:
```python
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split

data = load_breast_cancer()
print(data.DESCR)
```

```
.. _breast_cancer_dataset:

Breast cancer wisconsin (diagnostic) dataset
--------------------------------------------

**Data Set Characteristics:**

    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the class

    :Attribute Information:
        - radius (mean of distances from center to points on the perimeter)
        - texture (standard deviation of gray-scale values)
        - perimeter
        - area
        - smoothness (local variation in radius lengths)
        - compactness (perimeter^2 / area - 1.0)
        - concavity (severity of concave portions of the contour)
        - concave points (number of concave portions of the contour)
        - symmetry
        - fractal dimension ("coastline approximation" - 1)

        The mean, standard error, and "worst" or largest (mean of the three
        worst/largest values) of these features were computed for each image,
        resulting in 30 features.  For instance, field 0 is Mean Radius, field
        10 is Radius SE, field 20 is Worst Radius.

        - class:
                - WDBC-Malignant
                - WDBC-Benign

    :Summary Statistics:

    ===================================== ====== ======
                                           Min    Max
    ===================================== ====== ======
    radius (mean):                        6.981  28.11
    texture (mean):                       9.71   39.28
    perimeter (mean):                     43.79  188.5
```

area (mean):	143.5	2501.0
smoothness (mean):	0.053	0.163
compactness (mean):	0.019	0.345
concavity (mean):	0.0	0.427
concave points (mean):	0.0	0.201
symmetry (mean):	0.106	0.304
fractal dimension (mean):	0.05	0.097
radius (standard error):	0.112	2.873
texture (standard error):	0.36	4.885
perimeter (standard error):	0.757	21.98
area (standard error):	6.802	542.2
smoothness (standard error):	0.002	0.031
compactness (standard error):	0.002	0.135
concavity (standard error):	0.0	0.396
concave points (standard error):	0.0	0.053
symmetry (standard error):	0.008	0.079
fractal dimension (standard error):	0.001	0.03
radius (worst):	7.93	36.04
texture (worst):	12.02	49.54
perimeter (worst):	50.41	251.2
area (worst):	185.2	4254.0
smoothness (worst):	0.071	0.223
compactness (worst):	0.027	1.058
concavity (worst):	0.0	1.252
concave points (worst):	0.0	0.291
symmetry (worst):	0.156	0.664
fractal dimension (worst):	0.055	0.208

:Missing Attribute Values: None

:Class Distribution: 212 - Malignant, 357 - Benign

:Creator: Dr. William H. Wolberg, W. Nick Street, Olvi L. Mangasarian

:Donor: Nick Street

:Date: November, 1995

This is a copy of UCI ML Breast Cancer Wisconsin (Diagnostic) datasets.
https://goo.gl/U2Uwz2

Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

Separating plane described above was obtained using Multisurface Method-Tree (MSM-T) [K. P. Bennett, "Decision Tree

```
Construction Via Linear Programming." Proceedings of the 4th
Midwest Artificial Intelligence and Cognitive Science Society,
pp. 97-101, 1992], a classification method which uses linear
programming to construct a decision tree.  Relevant features
were selected using an exhaustive search in the space of 1-4
features and 1-3 separating planes.

The actual linear program used to obtain the separating plane
in the 3-dimensional space is that described in:
[K. P. Bennett and O. L. Mangasarian: "Robust Linear
Programming Discrimination of Two Linearly Inseparable Sets",
Optimization Methods and Software 1, 1992, 23-34].

This database is also available through the UW CS ftp server:

ftp ftp.cs.wisc.edu
cd math-prog/cpo-dataset/machine-learn/WDBC/

.. topic:: References

   - W.N. Street, W.H. Wolberg and O.L. Mangasarian. Nuclear feature extraction
     for breast tumor diagnosis. IS&T/SPIE 1993 International Symposium on
     Electronic Imaging: Science and Technology, volume 1905, pages 861-870,
     San Jose, CA, 1993.
   - O.L. Mangasarian, W.N. Street and W.H. Wolberg. Breast cancer diagnosis and
     prognosis via linear programming. Operations Research, 43(4), pages
570-577,
     July-August 1995.
   - W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Machine learning
techniques
     to diagnose breast cancer from fine-needle aspirates. Cancer Letters 77
(1994)
     163-171.
```

[7]:
```python
# Aufteilen in Trainings- und Testdaten
X = data.data
y = data.target

X_train, X_test, y_train, y_test = train_test_split(X,y,shuffle=True,
 ↪test_size=0.3)
```

[8]:
```python
# KNN erstellen und trainieren
# Wir verwenden hier "ADAM" als Optimizer (Adaptive moment estimation)
model2 = Sequential()
model2.add(Dense(30, input_dim=X_train.shape[1])) # Input-Layer
model2.add(Activation("relu"))
```

```python
model2.add(Dense(50)) # Hidden-Layer
model2.add(Activation("relu"))

model2.add(Dense(50)) # Hidden-Layer
model2.add(Activation("relu"))

model2.add(Dense(50)) # Hidden-Layer
model2.add(Activation("relu"))


model2.add(Dense(1)) # Output-Layer
model2.add(Activation("sigmoid"))

model2.compile(loss="binary_crossentropy", optimizer=Adam(lr=0.001),␣
 ↪metrics="accuracy")
model2.fit(x=X_train,y=y_train, epochs=100, validation_data=(X_test, y_test),␣
 ↪verbose=0)
```

[8]: <tensorflow.python.keras.callbacks.History at 0x1c5f6239760>

[9]:
```python
from sklearn.metrics import confusion_matrix, accuracy_score
pred = (model2.predict(X_test)>0.5).astype(int)
print(confusion_matrix(y_test, pred))
print(accuracy_score(y_test, pred))
```

```
[[ 61   7]
 [  0 103]]
0.9590643274853801
```