KuenstlichesNeuronalesNetz

May 12, 2023

1 KünstlicheNeuronaleNetze

In diesem Notebook wollen wir ein Künstliches Neuronales Netzwerk erstellen.

Im Gegensatz zu den bisher verwendeten Jupyter Notebooks kommt dieses Notebook nicht mehr mit den in Anaconda bereits installierten Packages aus. Wir müssen, damit dieses Notebook funktioniert, das Package *Keras* installieren. Dazu öffnet man unter Windows das Programm *Anaconda Powershell Prompt* (unter Linux oder Mac ein Terminalfenster öffnen) und gibt nacheinander folgenden Befehl ein:

- conda create -n tensorflow_env tensorflow
- conda activate tensorflow_env
- conda install -c anaconda keras

Die Ausführung jeweils nach einigen Sekunden mit einem **y** bestätigen, dann wird das Package installiert.

Als Datensatz verwenden wir wieder *diabetes.csv* und versuchen, eine Diabetes-Erkrankung vorherzusagen.

Feature	Bedeutung
age	age in years
sex	sex (1 = male, 0 = female)
cp	chest pain type
trestbps	resting blood pressure im mm Hg
chol	serum cholesteral in mg/dl
fbs	fasting blood sugar > 120 mg/dl (1=true, 0=false)
thalach	maximum heart rate achieved
restecg	resting electrocardiographic results
exang	exercise induced angina (0=no, 1=yes)
oldpeak	ST depression induced by exercise relative to rest
slope	the slope of the peak exercise ST segment
ca	number of major vessels (0-3) colored by flouroscopy
thal	3=normal, 6=fixed defect, 7=reversable defect
target	1 or 0

Wir teilen den Datensatz wieder in Trainings- und Testdaten auf, erstellen ein Objekt der Klasse *Sequential* und fügen mit der *add*-Methode mehrere Layer hinzu. Als Aktivierungsfunktion verwenden wir für die Hidden Layer die *relu*-, für die Output-Schicht die *Sigmoid*-Funktion. Für

die Qualitätsbeurteilung geben wir an, dass wir die Accuracy verwenden wollen. Das Training des Modells wird mit jeweils 10 Beobachtungen durchgeführt mit einer festgelegten Gesamtanzahl an Wiederholungen, den *epoches*, womit jeweils die Accuracy (bezogen auf die Trainingsdaten) ermittelt wird.

[12]:
```
import pandas as pd

url = "https://raw.githubusercontent.com/troescherw/datasets/master/diabetes.
 ↪csv"
df = pd.read_csv(url)
df.head()
```

[12]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

Da *Sequential* die Daten in Form eines Arrays benötigt, wandeln wir das DataFrame in ein Pandas Array um:

[13]:
```
import numpy as np
df = df.values
```

Wie üblich teilen wir den Datensatz in einen Test- und Trainingsdatensatz auf.

[14]:
```
from sklearn.model_selection import train_test_split
X = df[:,:13]
y = df[:, -1]

X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,␣
 ↪shuffle=True, random_state=42)
```

Nun erstellen wir das KNN, in dem wir ein Objekt der Klasse *Sequential* erstellen und die Layer hinzufügen. Außerdem definieren wir die Aktivierungsfunktionen. Außerdem entfernen wir auch Kanten aus dem Netz, sodass ggf. zu starke Gewichtungen wieder herauszunehmen, was u.U. zu Overfitting führt. Dies erreichen wir mit einem *Dropout*-Objekt.

[15]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
```

```
model = Sequential()
model.add(Dense(12, input_dim=13, activation='relu'))
model.add(Dense(500, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(13, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss="binary_crossentropy", optimizer="adam",␣
 ↪metrics=["accuracy"])
```

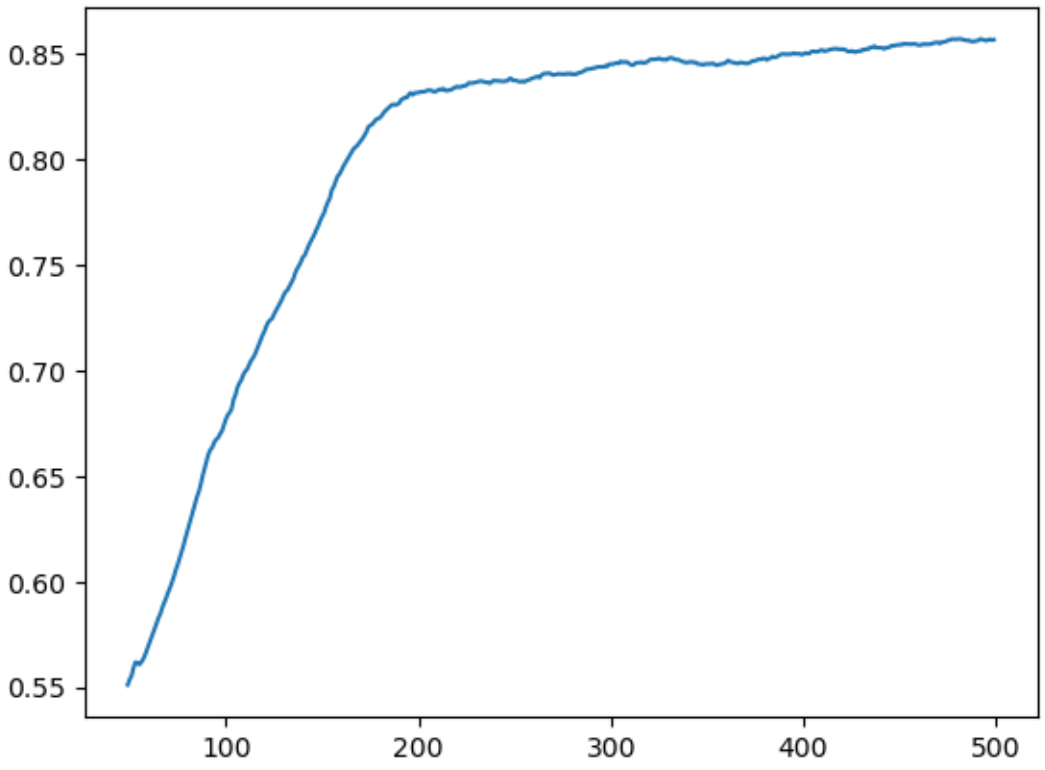
Nun trainieren wir das Netz, was allerdings etwas dauert...

[16]:
```
history = model.fit(X_train, y_train, epochs=500, batch_size=10, verbose=0)
```

Wir können die Accuracy, die im Laufe des Trainings ermittelt wurde, als Plot ausgeben.

[17]:
```
import matplotlib.pyplot as plt
import pandas as pd
plt.plot(pd.DataFrame(history.history["accuracy"]).rolling(50).mean())
```

[17]:
```
[<matplotlib.lines.Line2D at 0x1fe6e3c2820>]
```



Schließlich geben wir noch einen Report aus.

[26]:
```python
from sklearn.metrics import classification_report
pred=model.predict(X_test)
classes=np.argmax(pred,axis=1)
print(classification_report(y_test, classes))
```

```
3/3 [==============================] - 0s 2ms/step
              precision    recall  f1-score   support

         0.0       0.45      1.00      0.62        41
         1.0       0.00      0.00      0.00        50

    accuracy                           0.45        91
   macro avg       0.23      0.50      0.31        91
weighted avg       0.20      0.45      0.28        91


C:\Users\dea40349\Anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\dea40349\Anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\dea40349\Anaconda3\lib\site-
packages\sklearn\metrics\_classification.py:1344: UndefinedMetricWarning:
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

[ ]:
```python

```