Woche 10: Programmierung – Neuronale Netzwerke in Python (Teil 1)

Skript

Erarbeitet von
Ludmila Himmelspach

Lernziele

- Erklären können, was One-Hot-Codierung ist.
- Die Merkmalswerte eines Datensatzes auf einen Bereich zwischen 0 und 1 normalisieren können.
- Ein einfaches künstliches neuronales Netzwerk mithilfe des Keras-Moduls in Python programmieren können.
- Die Anzahl der Modellparameter berechnen können.

Inhalt

Einstieg

Künstliche neuronale Netzwerke werden wie die klassischen Machine Learning Verfahren unter anderem für Klassifikationsaufgaben eingesetzt. Diese werden dann auf gelabelten Beobachtungen trainiert und zur Vorhersage der Klassenzugehörigkeit neuer

Projekt „KI für alle"
HeiCAD – Heine Center for Artificial Intelligence and Data Science

Beobachtungen benutzt. In diesem Video lernst du, wie man einen Datensatz für das Trainieren eines künstlichen neuronalen Netzes vorbereitet und wie man ein einfaches Feed-Forward-Netz in Python programmiert.

Der Titanic-Datensatz

Zuerst importieren wir alle Module, die wir in unserem Programm brauchen werden.

```
# Importiere die nötigen Module
import seaborn as sns
import pandas as pd
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
import numpy as np
import keras
```

In diesem Video arbeiten wir mit dem *Titanic* Datensatz, der Informationen inklusive Überlebensstatus über 891 Passagiere der Titanic enthält. Dieser Datensatz stamm aus einer *Kaggle* Challenge zur Vorhersage des Überlebens der Passagiere.

Quelle [1]

Also besteht unsere Klassifikationsaufgabe darin, anhand der Informationen über die Passagiere ihre Überlebenschance zu bestimmen. Übrigens, die Hauptquelle, aus der die meisten Informationen über die Passagiere der Titanic entnommen wurden, ist die *Encyclopedia Titanica*. Auf ihrer Seite kannst du detaillierte Biographien und Fotos der Passagiere und Besatzungsmitglieder, sowie viele Fakten und Nachrichtenartikel über Titanic und ihren Untergang finden.

Quelle [2]

Wir laden aber zuerst den *Titanic*-Datensatz und speichern ihn in der Variable *titanic*. Dieser Datensatz ist im Modul *seaborn* enthalten.

Quelle [3]

```
# Lade den Titanic-Datensatz
titanic = sns.load_dataset("titanic")
```

Der Datensatz wird wie alle Datensätze aus diesem Modul in der Datenstruktur *DataFrame* gespeichert.

Quelle [4]

Mit der Methode *info()* können wir eine kurze Zusammenfassung des DateFrames einschließlich der Merkmalsbezeichnungen ausgeben lassen.

```
# Gebe eine kurze Zusammenfassung des DataFrames aus
print(titanic.info())
```

```
RangeIndex: 891 entries, 0 to 890
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype   
---  ------       --------------  -----   
 0   survived     891 non-null    int64   
 1   pclass       891 non-null    int64   
 2   sex          891 non-null    object  
 3   age          714 non-null    float64 
 4   sibsp        891 non-null    int64   
 5   parch        891 non-null    int64   
 6   fare         891 non-null    float64 
 7   embarked     889 non-null    object  
 8   class        891 non-null    category
 9   who          891 non-null    object  
 10  adult_male   891 non-null    bool    
 11  deck         203 non-null    category
 12  embark_town  889 non-null    object  
 13  alive        891 non-null    object  
 14  alone        891 non-null    bool    
```

In der Ausgabe sehen wir, dass dieser Datensatz 891 Einträge und 15 Spalten enthält. In unserem Fall handelt es sich um 15 Merkmale der 891 Passagiere auf der Titanic. Neben der Bezeichnung einzelner Merkmale bekommt man zusätzlich die Information darüber, in wie vielen Einträgen des Merkmals tatsächlich Werte vorhanden sind. Diese Angaben können uns bei der Auswahl der Merkmale für die weitere Analyse helfen. So sehen wir zum Beispiel, dass das Alter von nur 714 der 891 Passagiere bekannt ist. Obwohl dieses Merkmal für unsere Fragestellung sich als wichtig erweisen könnte, können wir es wegen vieler fehlender Einträgen zum Trainieren des Klassifikators nicht benutzen. Das Gleiche gilt für das Merkmal *deck*, in dem nur 203 Einträge vorhanden sind. In den Merkmalen *embarked* und *embark_town* fehlen zwar jeweils nur für zwei Beobachtungen die Einträge, da es sich aber um den Einschiffungshafen der Passagiere handelt, tragen diese Merkmale zu der Lösung unseres Klassifikationsproblems wahrscheinlich wenig bei. Deswegen lassen wir diese Merkmale bei der weiteren Analyse außer Betracht.

Da einige Merkmale ähnliche Informationen über die Passagiere zu enthalten scheinen, lassen wir uns am besten mit der Funktion *head()* die ersten fünf Zeilen des Datensatzes anzeigen.

```
# Gebe die ersten fünf Zeilen des Datensatzes aus
print(titanic.head())
```

```
   survived  pclass     sex   age  sibsp  parch     fare embarked   class  \
0         0       3    male  22.0      1      0   7.2500        S   Third   
1         1       1  female  38.0      1      0  71.2833        C   First   
2         1       3  female  26.0      0      0   7.9250        S   Third   
3         1       1  female  35.0      1      0  53.1000        S   First   
4         0       3    male  35.0      0      0   8.0500        S   Third   

     who  adult_male deck  embark_town alive  alone  
0    man        True  NaN  Southampton    no  False  
1  woman       False    C    Cherbourg   yes  False  
2  woman       False  NaN  Southampton   yes   True  
3  woman       False    C  Southampton   yes  False  
4    man        True  NaN  Southampton    no   True  
```

Und tatsächlich sehen wir in der Ausgabe, dass die Merkmale *survived* und *alive* die gleiche Information über den Überlebensstatus der Passagiere enthalten. Im Merkmal *embarked* sind die Einschiffungshäfen (*embark_town*) in der abgekürzten Form gespeichert. In den beiden Merkmalen *pclass* und *class* ist die Passagierklasse einmal als numerischer und einmal als kategorischer Attributwert erfasst und soll über den sozioökonomischen Status der Passagiere Aufschluss geben.

Die Bedeutung der Merkmale *sibsp* und *parch* lässt sich nicht so einfach aus ihrer Bezeichnung ableiten, kann aber auf der Internetseite der *Kaggle* Challenge nachgeschaut werden. Dort erfahren wir, dass im Merkmal *sibsp* die Anzahl der Geschwister bzw. der Ehepartner*innen an Bord erfasst ist. Im Merkmal *parch* ist die Anzahl der Eltern bzw. die Anzahl der Kinder an Bord der Titanic des jeweiligen Passagiers gespeichert. Auf der Seite der *Encyclopedia Titanica* kann man außerdem nachlesen, dass einige Kinder nur in Begleitung ihres Kindermädchens gereist sind. In diesem Fall wurde im Merkmal *parch* 0 als Wert eingetragen.

Nachdem wir die Bedeutung der Merkmale *sibsp* und *parch* geklärt haben, können wir das Merkmal *alone* „verwerfen“, weil dieses sich aus dem Zusammenschluss der Merkmale *sibsp* und *parch* ableiten lässt. Denn wer keine nahen Verwandten an Bord der Titanic hatte, galt als Alleinreisender.

Auf der Internetseite der *Kaggle* Challenge steht leider nichts über die Bedeutung der Merkmale *who* und *adult_male*. Während wir die Bedeutung des Merkmals *adult_male* aus seiner Bezeichnung ableiten können, können uns nur die Einträge des Merkmals *who* etwas über seine Bedeutung verraten. Mit der Funktion *unique()* können wir eindeutige Werte des Merkmals *who* ausgeben lassen.

```
# Gebe eindeutige Werte des Merkmals 'who' aus
print(titanic['who'].unique())
```

```
['man' 'woman' 'child']
```

Laut Ausgabe werden mit Hilfe dieses Merkmals die Passagiere in Kategorien „Mann“, „Frau“ und „Kind“ unterteilt. Da wir das Merkmal „Alter“ wegen fehlender Werte nicht benutzen können, können die Einträge des Merkmals *who* uns zumindest den Aufschluss über die grobe Altersgruppe der Passagiere geben.

Nachdem wir die Bedeutung aller Merkmale geklärt haben, wählen wir für die weitere Analyse nur die Merkmale *Passagierklasse*, *Geschlecht*, *Anzahl der Geschwister bzw. Ehepartner*innen an Bord*, *Anzahl der Eltern bzw. Kinder an Bord*, *Passagiertarif*, die *Altersgruppe* bzw. das Merkmal *who* aus und speichern ihre Werte im DataFrame *X* ab. Zur Überprüfung geben wir danach die ersten fünf Zeilen des neuen DataFrames aus.

```
# Speichere das zweite (Passagierklasse), das dritte (Geschlecht),
# das fünfte (Anzahl der Geschwister/Ehepartner an Bord), das sechste (Anzahl der
Eltern/Kinder an Bord),
# das siebte (Passagiertarif (Britisches Pfund)),
# und das zehnte (Mann/Frau/Kind) Merkmal in die Datenmatrix
X = titanic[['pclass', 'sex', 'sibsp', 'parch', 'fare', 'who']]
print(X.head())
```

```
   pclass     sex  sibsp  parch     fare    who
0       3    male      1      0   7.2500    man
1       1  female      1      0  71.2833  woman
2       3  female      0      0   7.9250  woman
3       1  female      1      0  53.1000  woman
4       3    male      0      0   8.0500    man
```

Da wir mit unserem Modell die Überlebenschance der Passagiere vorhersagen wollen, ist unser Zielmerkmal der Überlebensstatus. Deswegen speichern wir die Werte des Merkmals *survived* im DataFrame *y* ab.

```
# Speichere das erste Merkmal (überlebt) als Zielmerkmal
y = titanic[['survived']]
print(y.head())
```

```
   survived
0         0
1         1
2         1
3         1
4         0
```

One-Hot-Codierung

Jetzt müssen wir uns um die Vorverarbeitung kategorischer Merkmale in unserem Datensatz kümmern, weil neuronale Netzwerke nur numerische Werte direkt verarbeiten können. Wir fangen mit dem Merkmal *who* an. Eigentlich brauchen wir daraus nur die Information darüber, ob es sich bei einem Passagier um ein Kind gehandelt hat, denn die Geschlechtszugehörigkeit der erwachsenen Passagiere ist im Merkmal *Geschlecht* bereits enthalten. Deswegen ersetzen wir alle Einträge im Merkmal *who*, wo früher *child* stand, mit 1 und alle anderen mit 0. Dabei hilft uns die Funktion *where()* des NumPy-Moduls. In der Bedingung der *where()*-Funktion vergleichen wir zuerst die Werte des Merkmals *who* mit der Zeichenkette *child*. Kurz zur Erinnerung: In Python fungiert das doppelte Gleichheitszeichen als Vergleichsoperator. Nach einem Komma können wir einen Wert angeben, mit dem jeder Eintrag ersetzt wird, für den die Bedingung erfüllt ist. Also geben wir hier 1 an. Nach dem zweiten Komma geben wir 0 als den Wert an, mit dem alle anderen Einträge ersetzt werden sollen. Auf Grund der Änderung der Einträge macht es Sinn, das Merkmal *who* umzubenennen. Das erreichen wir mit der Funktion *rename()* des Pandas-Moduls. In der Klammer der Funktion spezifizieren wir, dass wir die Spalte bzw. das Merkmal *who* in *is_child* umbenennen.

Um Warnungen zu unterdrücken, die beim Benutzen einiger Funktionen in der Regel angezeigt werden, wird auf der Pandas-Seite empfohlen den folgenden Befehl auszuführen.

```
# Damit unterdrücken wir einige Warnungen
pd.options.mode.chained_assignment = None



# Speichere im Merkmal 'who' nur die Information darüber, ob ein Passagier
# ein Kind war, und benenne es in 'is_child' um
X['who'] = np.where(X['who'] == 'child', 1, 0)
X = X.rename(columns={'who': 'is_child'})
print(X.head())
```

```
   pclass     sex  sibsp  parch     fare embarked  is_child
0       3    male      1      0   7.2500        S         0
1       1  female      1      0  71.2833        C         0
2       3  female      0      0   7.9250        S         0
3       1  female      1      0  53.1000        S         0
4       3    male      0      0   8.0500        S         0
```

Wie schon vorhin erwähnt, können neuronale Netzwerke nur numerischen Werte direkt verarbeiten. In unserem Datensatz haben wir aber noch das Merkmale „Geschlecht“, das kategorische Werte enthält. Es ergibt wenig Sinn, die Ausprägungen dieses Merkmals einfach durch numerische Werte zu ersetzen, da ein künstliches neuronales Netz Ordnung zwischen diesen annehmen wird. Die Werte des Merkmals „Geschlecht“ lassen sich aber wie zum Beispiel die Werte des Merkmals „Passagiertarif“ nicht miteinander vergleichen. Eine gängige Vorgehensweise kategorische Merkmale in numerische Merkmale umzuwandeln, nennt sich *One-Hot-Codierung*. Bei der One-Hot-Codierung wird für jede Ausprägung eines kategorischen Merkmals ein eigenes Merkmal erstellt, dessen Werte 1 für die Beobachtungen annehmen, für die die Ausprägung des kategorischen Merkmals zutrifft. Für alle anderen Beobachtungen wird 0 im neuen Merkmal gespeichert.

Quelle [5]

In Python lassen sich die kategorischen Merkmale nach der One-Hot-Methode zum Beispiel mit Hilfe der Funktion *get_dummies()* des Pandas-Moduls umwandeln. In der Funktion müssen wir dem Parameter *columns* die Liste der Merkmale zuweisen, die umcodiert werden sollen. Das ist bei uns das Merkmal „Geschlecht“. Außerdem können wir mit Hilfe des Parameters *dtype* den Datentyp der Einträge der neuen Merkmale festlegen. Wir speichern die Werte als Fließkommazahlen. In der Ausgabe sehen wir, dass wir im DataFrame *X* anstatt des alten Merkmals „Geschlecht“ zwei neue Merkmale *sex_female* und *sex_male* mit den entsprechenden Einträgen haben.

```
X = pd.get_dummies(X, columns=['sex'], dtype=float)
print(X.head())
```

```
   pclass  sibsp  parch     fare  is_child  sex_female  sex_male
0       3      1      0   7.2500         0         0.0       1.0
1       1      1      0  71.2833         0         1.0       0.0
2       3      0      0   7.9250         0         1.0       0.0
3       1      1      0  53.1000         0         1.0       0.0
4       3      0      0   8.0500         0         0.0       1.0
```

Neben dem Merkmal „Geschlecht“ müssen wir noch unser Zielmerkmal *survived* gemäß der One-Hot-Codierung umwandeln, weil das künstliche neuronale Netz für jede Klasse *überlebt* bzw. *nicht überlebt* jeweils eine Vorhersage berechnen wird. Dafür nutzen wir wieder die Funktion *get_dummies()*.

```
y = pd.get_dummies(y, columns=['survived'], dtype=float)
print(y.head())
```

```
   survived_0  survived_1
0         1.0         0.0
1         0.0         1.0
2         0.0         1.0
3         0.0         1.0
4         1.0         0.0
```

Quellen

Quelle [1] Titanic - Machine Learning from Disaster. https://www.kaggle.com/competitions/titanic/overview (Abgerufen am 01.02.2025)

Quelle [2] *Encyclopedia Titanica*. www.encyclopedia-titanica.org (Abgerufen am 01.02.2025)

Quelle [3] Waskom, M. L. (2021). seaborn: statistical data visualization. Journal of Open Source Software, 6(60), 3021. https://doi.org/10.21105/joss.03021

Quelle [4] NumFOCUS, Inc. (2022). DataFrame. In *pandas API Reference, Release 1.5.2* https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.html

Quelle [5] Géron, A. (2023). *Praxiseinstieg Machine Learning mit Scikit-Learn, Keras und TensorFlow: Konzepte, Tools und Techniken für intelligente Systeme* (aktualisierte und erweiterte Auflage.). O'Reilly Verlag.

Weiterführendes Material

https://www.tensorflow.org/tutorials/keras/classification

Disclaimer

Transkript zu dem Video „Woche 10: Programmierung – Neuronale Netzwerke in Python (Teil 1)“, Ludmila Himmelspach.