Lineare_Regression_Moore_Penrose

April 5, 2021

1 Lineare Regression mit Moore-Penrose-Pseudo-Inversen

Wir können die Koeffizienten einer Linearen Regressionsgleichung auch mit Hilfe der Linearen Algebra ermitteln. Allgmeine benötigt man bei k Koeffizienten (Unbekannten) k linear unabhängige Gleichungen, um eine Lösung zu ermitteln. Ist Y der Ergebnisvektor und X die Koeffizientenmatrix, so kann man schreiben:

$Y = X \cdot \beta$

Gesucht sind die Koeffizienten β. Multipliziert man beide Seiten der Gleichung mit der inversen Matrix von X, also X^-1

$X^-1 \cdot Y = X^-1 \cdot X \cdot \beta$

$X^-1 \cdot X$ ergibt die Einheitsmatrix, sodass sich ergibt:

$\beta = X^-1 \cdot Y$

Hier haben wir allerdings wesentlich mehr Gleichungen (entspricht Stichprobengröße) und erhalten somit ein überbestimmtes Gleichungssystem. Hier hilft uns die Moore-Penrose-Pseudo-Inverse X^+, die wie folgt definiert ist:

$X^+ = (X^T X)^{-1} X^T$

Eingesetzt in die obige Gleichung ergibt sich:

$\beta = X^+ Y \cdot Y = (X^T X)^{-1} X^T \cdot Y$

Diese Gleichung setzen wir jetzt mit Hilfe von *numpy*-Funktionen um. Wir benötigen:

- X^T: Wir müssen also die Matrix X transponieren (aus Zeilen werden Spalten, aus Spalten werden Zeilen. Dies geschieht ganz einfach durch ".T", also zum Beispiel x.T, wenn x die Matrix ist.
- Multiplikation von Matrizen: Dies können wir durch den @-Operator durchführen. Sind a und b Matrizen (2-dimensionale Numpy-Arrays), so können wir mit a@b eine Matrixmultiplikation durchführen.
- Schließlich müssen wir noch die Inverse einer Matrix ermitteln. Wollen wir von der Matrix a die Inverse berechnen, also a^{-1}, so erledigt dies die Funktion *np.linalg.inv*.

Mit diesen Informationen können wir nun die Koeffizienten des überbestimmten Gleichungssystems lösen. Wir wollen wieder den Verbrauch der Autos in Abhängigkeit von Gewicht und Leistung ermitteln.

Laden wir zuerste den Datensatz:

[1]:
```
import pandas as pd
url = "https://raw.githubusercontent.com/troescherw/datasets/master/autos.csv"
autos = pd.read_csv(url)
autos.head()
```

[1]:

	Verbrauch	Leistung	Gewicht
0	11.20	82	1310
1	11.20	82	1437
2	10.32	69	1160
3	10.99	82	1607
4	12.58	130	1720

In der Spalte *Verbrauch* stehen die vorherzusagenden Werte, also der Ergebnisvektor y:

[4]:
```
y = autos.Verbrauch
```

[ ]:
```
Die Spalten *Leistung* und *Gewicht* bilden die Koeffizienten-Matrix X:
```

[7]:
```
x = autos[["Leistung", "Gewicht"]]
```

Allerdings benötigen wir noch einen Koeffizienten für die Konstante β_0. Da es sich eben um eine Konstante handelt, fügen wir in die Koeffizientenmatrix noch eine Spalte mit Einsen ein:

[8]:
```
x.insert(0, "Beta_0", 1)
x.head()
```

[8]:

	Beta_0	Leistung	Gewicht
0	1	82	1310
1	1	82	1437
2	1	69	1160
3	1	82	1607
4	1	130	1720

Nun wenden wir die Formel von oben an:

[10]:
```
import numpy as np
betas = np.linalg.inv(x.T @ x) @ x.T @ y
betas
```

[10]:
```
0    1.493910
1    0.023576
2    0.005399
dtype: float64
```

Auf das gleiche Ergebnis kommt auch die Funktion *ols*:

[14]:
```
import statsmodels.formula.api as sm
model = sm.ols("Verbrauch~Leistung+Gewicht", data=autos).fit()
model.params
```

[14]:
```
Intercept    1.493910
Leistung     0.023576
Gewicht      0.005399
dtype: float64
```

Wir können die Pseudo-Inverse auch direkt mit Hilfe der Funktion *np.linalg.pinv* berechnen lassen:

[16]:
```
np.linalg.pinv(x)@y
```

[16]:
```
array([1.49391033, 0.0235758 , 0.005399  ])
```

Weiter oben ist aufgeführt, dass es sich bei der Koeffizienten-Matrix um *linear unabhängige* Gleichungen handeln muss. Dies erklärt nun auch, weshalb man bei der Dummy-Kodierung nur k-1 Spalten einfügen darf. Würde man dies missachten, und zum Beispiel bei der Kategorie *Lage* im Beispiel mit den Mietpreisen folgende Matrix erstellen:

Quadratmeter	Innenstadt	Umland	Aussenbezirk
41	0	0	1
79	0	1	0
180	1	0	0
…	…	…	…

Wo wären die Spalten *Innenstadt*, *Umland*, und *Aussenbezirk* linear abhängig: Man könnte auf den Inhalt einer Spalte schließen, in dem wir die anderen beiden Spaltenwerte kennen. Wäre zum Beispiel $Innenstadt = 0$ und $Umland = 0$, so muss in *Aussenbezirk* eine 1 stehen!

Ganz konkret wird es zum Problem, wenn wir die Koeffizienten mit dieser **falschen** Dummy-Kodierung durchführen wollen. Es erscheint nämlich eine Fehlermeldung!

[36]:
```
url = "https://raw.githubusercontent.com/troescherw/datasets/master/wohnungen.
↪csv"
wohnungen = pd.read_csv(url, delimiter=";")
wohnungen_dummies = pd.get_dummies(wohnungen)
wohnungen_dummies
```

[36]:

	Mietpreis	Quadratmeter	Lage_Aussenbezirk	Lage_Innenstadt	Lage_Umland
0	1100	87	0	0	1
1	588	42	0	0	1
2	850	54	0	0	1
3	500	33	0	0	1
4	1900	104	0	0	1
..	…	…	…	…	…
95	1800	101	1	0	0
96	2958	174	1	0	0
97	950	50	1	0	0
98	2656	166	1	0	0
99	2400	160	1	0	0

```
[100 rows x 5 columns]
```

Berechnen wir mit der *ols*-Funktion die Koeffizienten, so erhalten wir auch eine Warnung, dass der kleinste "Eigenwert" sehr gering ist, was auf eine Multikollinearität hinweist. Dies bedeutet, dass die berechneten Koeffizienten ggf. nicht korrekt sind bzw. das erstellte Modell sehr instabil ist!

[38]:
```
model = sm.
↪ols("Mietpreis~Quadratmeter+Lage_Aussenbezirk+Lage_Innenstadt+Lage_Umland",
              data=wohnungen_dummies).fit()
model.summary()
```

[38]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:              Mietpreis   R-squared:                       0.943
Model:                            OLS   Adj. R-squared:                  0.941
Method:                 Least Squares   F-statistic:                     527.5
Date:                Mon, 05 Apr 2021   Prob (F-statistic):           1.73e-59
Time:                        10:51:25   Log-Likelihood:                -682.44
No. Observations:                 100   AIC:                             1373.
Df Residuals:                      96   BIC:                             1383.
Df Model:                           3
Covariance Type:            nonrobust
================================================================================
=====
                        coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------------------
-----
Intercept           151.1396     39.428      3.833      0.000      72.876
229.403
Quadratmeter         17.4891      0.489     35.755      0.000      16.518
18.460
Lage_Aussenbezirk  -116.5511     31.641     -3.684      0.000    -179.358
-53.745
Lage_Innenstadt     536.6284     37.544     14.293      0.000     462.103
611.153
Lage_Umland        -268.9377     41.078     -6.547      0.000    -350.476
-187.399
==============================================================================
Omnibus:                        0.150   Durbin-Watson:                   1.592
Prob(Omnibus):                  0.928   Jarque-Bera (JB):                0.328
Skew:                          -0.036   Prob(JB):                        0.849
Kurtosis:                       2.729   Cond. No.                     7.13e+17
==============================================================================

Notes:
```

```
[1] Standard Errors assume that the covariance matrix of the errors is correctly
specified.
[2] The smallest eigenvalue is 2.25e-30. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
"""
```