MOwNiT

March 13, 2024

1 Laboratorium 02

1.1 Metoda najmniejszych kwadratów

Iga Antonik, Helena Szczepanowska

2 Zadanie 1.

Celem zadania jest zastosowanie metody najmniejszych kwadratów do predykcji, czy nowotwór jest złośliwy (ang. malignant) czy łagodny (ang. benign). Nowotwory złośliwe i łagodne mają różne charakterystyki wzrostu. Istotne cechy to m. in. promień i tekstura. Charakterystyki te wyznaczane są poprzez diagnostykę obrazową i biopsje. Do rozwiązania problemu wykorzystamy bibliotekę pandas, typ DataFrame oraz dwa zbiory danych:

- breast-cancer-train.dat
- breast-cancer-validate.dat.

Nazwy kolumn znajdują się w pliku breast-cancer.labels. Pierwsza kolumna to identyfikator pacjenta patient ID. Dla każdego pacjenta wartość w kolumnie Malignant/Benign wskazuje klasę, tj. czy jego nowotwór jest złośliwy czy łagodny. Pozostałe 30 kolumn zawiera cechy, tj. charakterystyki nowotworu.

- (a) Otwórz zbiory breast-cancer-train.dat i breast-cancer-validate.dat używając funkcji pd.io.parsers.read csv z biblioteki pandas.
- (b) Stwórz histogram i wykres wybranej kolumny danych przy pomocy funkcji hist oraz plot. Pamiętaj o podpisaniu osi i wykresów.
- (c) Stwórz reprezentacje danych zawartych w obu zbiorach dla liniowej i kwadratowej metody najmniejszych kwadratów (łącznie 4 macierze). Dla reprezentacji kwadratowej użyj tylko podzbioru dostępnych danych, tj. danych z kolumnradius (mean), perimeter (mean), area (mean), symmetry (mean).
- (d) Stwórz wektor b dla obu zbiorów (tablicę numpy 1D-array o rozmiarze identycznym jak rozmiar kolumny Malignant/Benign odpowiedniego zbioru danych). Elementy wektora b to 1 jeśli nowotwór jest złośliwy, -1 w przeciwnym wypadku. Funkcja np.where umożliwi zwięzłe zakodowanie wektora b.
- (e) Znajdź wagi dla liniowej oraz kwadratowej reprezentacji najmniejszych kwadratów przy pomocy macierzy A zbudowanych na podstawie zbioru breast-cancer-train.dat. Potrzebny będzie także wektor b zbudowany na podstawie zbioru breast-cancer-train.dat.

Uwaga. Problem najmniejszych kwadratów należy rozwiązać stosując równanie normalne (tj. nie używając funkcji scipy.linalg.lstsq). Rozwiązując równanie normalne należy użyć funkcji solve, unikając obliczania odwrotności macierzy funkcją scipy.linalg.pinv.

- (f) Oblicz współczynniki uwarunkowania macierzy, cond(AT A), dla liniowej i kwadratowej metody najmniejszych kwadratów.
- (g) Sprawdź jak dobrze otrzymane wagi przewidują typ nowotworu (łagodny czy złośliwy). W tym celu pomnóż liniową reprezentację zbioru breast-cancer-validate.dat oraz wyliczony wektor wag dla reprezentacji liniowej. Następnie powtórz odpowiednie mnożenie dla reprezentacji kwadratowej. Zarówno dla reprezentacji liniowej jak i kwadratowej otrzymamy wektor p. Zakładamy, że jeśli p[i] $>$ 0, to i-ta osoba (prawdopodobnie) ma nowotwór złośliwy. Jeśli p[i] 0 to i-ta osoba (prawdopodobnie) ma nowotwór łagodny.

Porównaj wektory p dla reprezentacji liniowej i kwadratowej z wektorem b (użyj reguł p[i] $>$ 0 oraz p[i] 0).

Oblicz liczbę fałszywie dodatnich (ang. false-positives) oraz fałszywie ujemnych (ang. false-negatives) przypadków dla obu reprezentacji. Przypadek fałszywie dodatni zachodzi, kiedy model przewiduje nowotwór złośliwy, gdy w rzeczywistości nowotwór był łagodny. Przypadek fałszywie ujemny za- chodzi, kiedy model przewiduje nowotwór łagodny, gdy w rzeczywistości nowotwór był złośliwy.

2.1 Rozwiązanie

2.1.1 Biblioteki

Korzystam z biblioteki NumPy ze względu na jej zalety w pracy z wielowymiarowymi tablicami danych oraz operacjami numerycznymi. Używam także biblioteki Pandas, która umożliwia efektywną pracę z danymi w postaci tabelarycznej. Do rysowania wykresów wykorzystuję bibliotekę matplotlib.

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from numpy.linalg import solve, cond
```

Wczytanie nazw kolumn z pliku

[ ]:
```
labels_path = './breast-cancer.labels'
with open(labels_path, 'r') as file:
    column_names = file.read().splitlines()
```

Wczytanie zbiorów danych

[ ]:
```
train_data_path = './breast-cancer-train.dat'
validate_data_path = './breast-cancer-validate.dat'
train_data = pd.io.parsers.read_csv(train_data_path, header=None,␣
 ↪names=column_names)
validate_data = pd.io.parsers.read_csv(validate_data_path, header=None,␣
 ↪names=column_names)
```

```
train_data.head()
validate_data.head()
```

[ ]:

	patient ID	Malignant/Benign	radius (mean)	texture (mean)	\
0	892438	M	19.53	18.90	
1	892604	B	12.46	19.89	
2	89263202	M	20.09	23.86	
3	892657	B	10.49	18.61	
4	89296	B	11.46	18.16	

	perimeter (mean)	area (mean)	smoothness (mean)	compactness (mean)	\
0	129.50	1217.0	0.11500	0.16420	
1	80.43	471.3	0.08451	0.10140	
2	134.70	1247.0	0.10800	0.18380	
3	66.86	334.3	0.10680	0.06678	
4	73.59	403.1	0.08853	0.07694	

	concavity (mean)	concave points (mean)	…	radius (worst)	\
0	0.21970	0.10620	…	25.93	
1	0.06830	0.03099	…	13.46	
2	0.22830	0.12800	…	23.68	
3	0.02297	0.01780	…	11.06	
4	0.03344	0.01502	…	12.68	

	texture (worst)	perimeter (worst)	area (worst)	smoothness (worst)	\
0	26.24	171.10	2053.0	0.1495	
1	23.07	88.13	551.3	0.1050	
2	29.43	158.80	1696.0	0.1347	
3	24.54	70.76	375.4	0.1413	
4	21.61	82.69	489.8	0.1144	

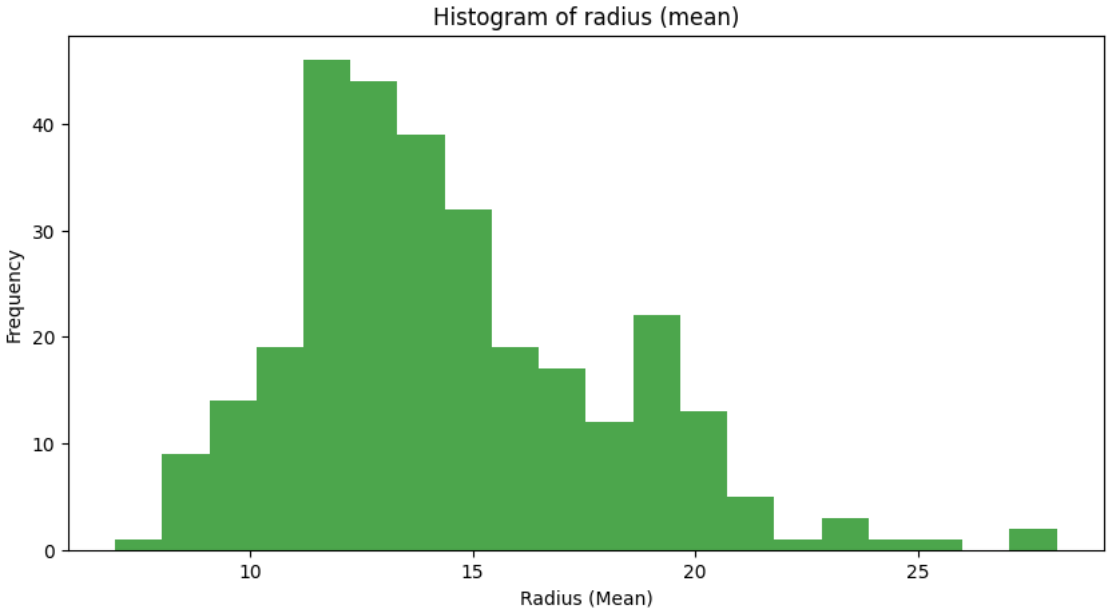
	compactness (worst)	concavity (worst)	concave points (worst)	\
0	0.4116	0.61210	0.19800	
1	0.2158	0.19040	0.07625	
2	0.3391	0.49320	0.19230	
3	0.1044	0.08423	0.06528	
4	0.1789	0.12260	0.05509	

	symmetry (worst)	fractal dimension (worst)
0	0.2968	0.09929
1	0.2685	0.07764
2	0.3294	0.09469
3	0.2213	0.07842
4	0.2208	0.07638

[5 rows x 32 columns]

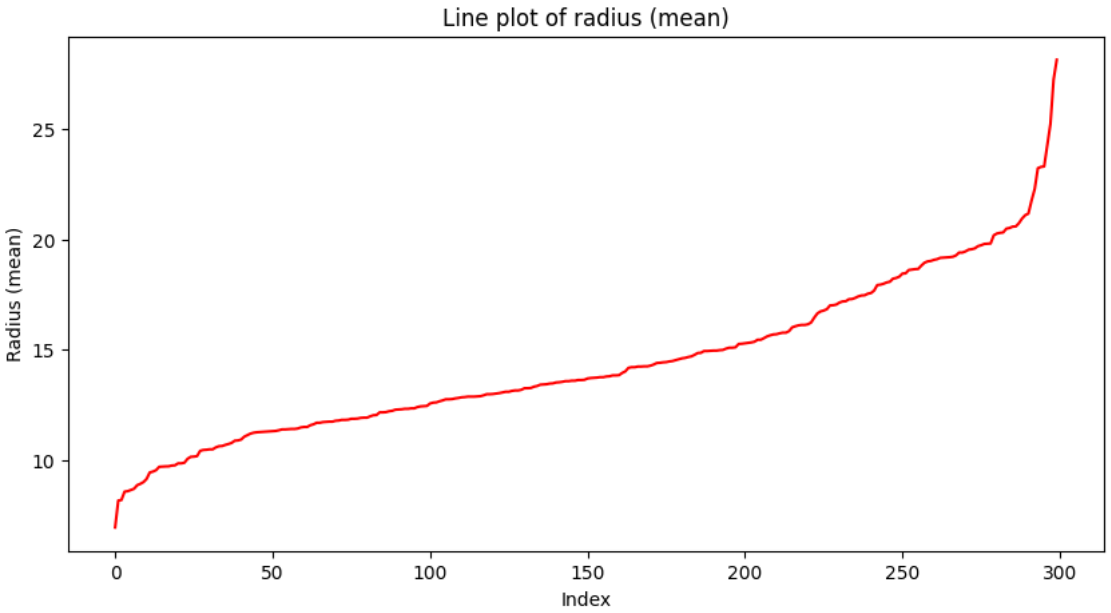
2.1.2 Histogram dla kolumny ‘radius (mean)’

```
[ ]: plt.figure(figsize=(10, 5))
plt.hist(train_data['radius (mean)'], bins=20, color='green', alpha=0.7)
plt.title('Histogram of radius (mean)')
plt.xlabel('Radius (Mean)')
plt.ylabel('Frequency')
plt.show()
```



2.1.3 Wykres liniowy dla kolumny ‘radius (mean)’

```
[ ]: plt.figure(figsize=(10, 5))
plt.plot(train_data['radius (mean)'].sort_values().reset_index(drop=True),␣
  ↪color='red')
plt.title('Line plot of radius (mean)')
plt.xlabel('Index')
plt.ylabel('Radius (mean)')
plt.show()
```



Na wykresach mozemy zaobserwowac, ze zdecydowana wiekszosc pacjentów posiada guza, którego średnica jest między 10 a 20. Wykres liniowy został narysowany dla rosnących wartości średnicy aby lepiej mozna było zaobserwować tendencję wielkości.

2.1.4 Przygotowanie macierzy dla metod liniowej i kwadratowej

Dla metody liniowej macierz wygląda następująco:

$$A_{\text{lin}} = \begin{bmatrix} f_{1,1} & f_{1,2} & f_{1,3} & f_{1,4} \\ f_{2,1} & f_{2,2} & f_{2,3} & f_{2,4} \\ \vdots & \vdots & \vdots & \vdots \\ f_{n,1} & f_{n,2} & f_{n,3} & f_{n,4} \end{bmatrix}$$

A dla kwadratowej:

$$A_{\text{quad}} = \begin{bmatrix} f_{1,1}, f_{1,2}, f_{1,3}, f_{1,4}, f_{1,1}^2, f_{1,2}^2, f_{1,3}^2, f_{1,4}^2, f_{1,1}f_{1,2}, f_{1,1}f_{1,3}, f_{1,1}f_{1,4}, f_{1,2}f_{1,3}, f_{1,2}f_{1,4}, f_{1,3}f_{1,4} \\ \vdots \\ f_{n,0}, f_{n,1}, f_{n,2}, f_{n,3}, f_{n,0}^2, f_{n,1}^2, f_{n,3}^2, f_{n,4}^2, f_{n,1}f_{n,2}, f_{n,1}f_{n,3}, f_{n,1}f_{n,4}, f_{n,2}f_{n,3}, f_{n,2}f_{n,4}, f_{n,3}f_{n,4} \end{bmatrix}$$

[ ]:
```
columns_quadratic = ['radius (mean)', 'perimeter (mean)', 'area (mean)',␣
↪'symmetry (mean)']
cq = columns_quadratic
```

```
A_train_linear = np.matrix(train_data.iloc[:, 2:])
A_validate_linear = np.matrix(validate_data.iloc[:, 2:])

train_quadratic = train_data[columns_quadratic]
validate_quadratic = validate_data[columns_quadratic]

quadratic_square =  np.square(train_quadratic)
quadratic_product = pd.concat([train_quadratic[cq[0]]*train_quadratic[cq[1]],␣
 ↪train_quadratic[cq[0]]*train_quadratic[cq[2]],
                               train_quadratic[cq[0]]*train_quadratic[cq[3]],␣
 ↪train_quadratic[cq[1]]*train_quadratic[cq[2]],
                               train_quadratic[cq[1]]*train_quadratic[cq[3]],␣
 ↪train_quadratic[cq[2]]*train_quadratic[cq[3]]], axis = 1)

A_train_quadratic = np.concatenate((train_quadratic, quadratic_square,␣
 ↪quadratic_product), axis = 1)

validate_square =  np.square(validate_quadratic)
validate_product = pd.
 ↪concat([validate_quadratic[cq[0]]*validate_quadratic[cq[1]],␣
 ↪validate_quadratic[cq[0]]*validate_quadratic[cq[2]],
                               ␣
 ↪validate_quadratic[cq[0]]*validate_quadratic[cq[3]],␣
 ↪validate_quadratic[cq[1]]*validate_quadratic[cq[2]],
                               ␣
 ↪validate_quadratic[cq[1]]*validate_quadratic[cq[3]],␣
 ↪validate_quadratic[cq[2]]*validate_quadratic[cq[3]]], axis = 1)

A_validate_quadratic = np.concatenate((validate_quadratic, validate_square,␣
 ↪validate_product), axis = 1)

print("A_train_linear")
print(A_train_linear, " \n")
print("A_validate_linear")
print(A_validate_linear ,"\n")
print("A_train_quadratic")
print(A_train_quadratic, " \n")
print("A_validate_quadratic")
print(A_validate_quadratic, "\n")
```

```
A_train_linear
[[1.799e+01 1.038e+01 1.228e+02 … 2.654e-01 4.601e-01 1.189e-01]
 [2.057e+01 1.777e+01 1.329e+02 … 1.860e-01 2.750e-01 8.902e-02]
 [1.969e+01 2.125e+01 1.300e+02 … 2.430e-01 3.613e-01 8.758e-02]
 …
 [1.176e+01 1.814e+01 7.500e+01 … 7.160e-02 1.978e-01 6.915e-02]
```

```
 [1.426e+01 1.817e+01 9.122e+01 … 7.530e-02 2.636e-01 7.676e-02]
 [1.051e+01 2.309e+01 6.685e+01 … 3.125e-02 2.227e-01 6.777e-02]]

A_validate_linear
[[1.953e+01 1.890e+01 1.295e+02 … 1.980e-01 2.968e-01 9.929e-02]
 [1.246e+01 1.989e+01 8.043e+01 … 7.625e-02 2.685e-01 7.764e-02]
 [2.009e+01 2.386e+01 1.347e+02 … 1.923e-01 3.294e-01 9.469e-02]
 …
 [9.423e+00 2.788e+01 5.926e+01 … 0.000e+00 2.475e-01 6.969e-02]
 [1.459e+01 2.268e+01 9.639e+01 … 1.105e-01 2.258e-01 8.004e-02]
 [1.151e+01 2.393e+01 7.452e+01 … 9.653e-02 2.112e-01 8.732e-02]]

A_train_quadratic
[[1.7990000e+01 1.2280000e+02 1.0010000e+03 … 1.2292280e+05
  2.9705320e+01 2.4214190e+02]
 [2.0570000e+01 1.3290000e+02 1.3260000e+03 … 1.7622540e+05
  2.4081480e+01 2.4027120e+02]
 [1.9690000e+01 1.3000000e+02 1.2030000e+03 … 1.5639000e+05
  2.6897000e+01 2.4890070e+02]
 …
 [1.1760000e+01 7.5000000e+01 4.3110000e+02 … 3.2332500e+04
  1.2142500e+01 6.9795090e+01]
 [1.4260000e+01 9.1220000e+01 6.3310000e+02 … 5.7751382e+04
  1.4914470e+01 1.0351185e+02]
 [1.0510000e+01 6.6850000e+01 3.3420000e+02 … 2.2341270e+04
  1.1331075e+01 5.6646900e+01]]

A_validate_quadratic
[[1.9530000e+01 1.2950000e+02 1.2170000e+03 … 1.5760150e+05
  2.3206400e+01 2.1808640e+02]
 [1.2460000e+01 8.0430000e+01 4.7130000e+02 … 3.7906659e+04
  1.4324583e+01 8.3938530e+01]
 [2.0090000e+01 1.3470000e+02 1.2470000e+03 … 1.6797090e+05
  3.0294030e+01 2.8045030e+02]
 …
 [9.4230000e+00 5.9260000e+01 2.7130000e+02 … 1.6077238e+04
  1.0323092e+01 4.7260460e+01]
 [1.4590000e+01 9.6390000e+01 6.5710000e+02 … 6.3337869e+04
  1.4015106e+01 9.5542340e+01]
 [1.1510000e+01 7.4520000e+01 4.0350000e+02 … 3.0068820e+04
  1.0343376e+01 5.6005800e+01]]
```

2.1.5 Stworzenie wektorów b

[ ]:
```
b_train = np.where(train_data['Malignant/Benign'] == 'M', 1, -1)
b_validate = np.where(validate_data['Malignant/Benign'] == 'M', 1, -1)
```

2.1.6 Znalezienie wag metodą najmniejszych kwadratów

$$w = (A^T A)^{-1} A^T b$$

[ ]:
```
weights_linear = solve(A_train_linear.T @ A_train_linear, A_train_linear.T @␣
 ↪b_train[:, np.newaxis]).flat
weights_quadratic = solve(A_train_quadratic.T @ A_train_quadratic,␣
 ↪A_train_quadratic.T @ b_train[:, np.newaxis]).flat
```

2.1.7 Obliczenie współczynników uwarunkowania macierzy

[ ]:
```
cond_linear = cond(A_train_linear.T @ A_train_linear)
cond_quadratic = cond(A_train_quadratic.T @ A_train_quadratic)

print("cond_linear: ",cond_linear, "\n cond_quadratic: ",cond_quadratic)
```

```
cond_linear:  1809248222566.8225
 cond_quadratic:  9.056816948763561e+17
```

2.1.8 Obliczenie fałszywie dodatnich oraz fałszywie ujemnych przypadków

[ ]:
```
p_linear = A_validate_linear @ weights_linear
p_quadratic = A_validate_quadratic @ weights_quadratic

predictions_linear = np.where(p_linear > 0, 1, -1)
predictions_quadratic = np.where(p_quadratic > 0, 1, -1)

fp_linear = np.sum((predictions_linear == 1) & (b_validate == -1))
fn_linear = np.sum((predictions_linear == -1) & (b_validate == 1))
fp_quadratic = np.sum((predictions_quadratic == 1) & (b_validate == -1))
fn_quadratic = np.sum((predictions_quadratic == -1) & (b_validate == 1))

print("Linear \n False Positive:", fp_linear, "\n False Negative:", fn_linear)
print("Quadratic \n False Positive:", fp_quadratic, "\n False Negative:",␣
 ↪fn_quadratic)
```

```
Linear
 False Positive: 6
 False Negative: 2
Quadratic
 False Positive: 15
 False Negative: 5
```

2.2 Wnioski

Jak mozna zauwazyc waga oraz macierz otrzymane przy pomocy liniowej metody najmniejszych kwadratów dały mniej zarówno fałszywie pozytywnych jak i fałszywie negatywnych wyników niz metoda kwadratowa. Moze to wynikac z faktu, ze do macierzy tworzonej dla metody kwadratowej przekazalismy mniej danych.

Współczynnik uwarunkowania macierzy dla liniowej reprezentacji jest równiez o wiele mniejszy od współczynnika wyliczonego dla reprezentacji kwadratowej. Oznacza to, ze w większym stopniu błąd reprezentacji numerycznej danych wejściowych wpływa na błąd wyniku w reprezentacji kwadratowej.

Podsumowując, przeprowadzona analiza pokazuje, ze ilość przekazanych do modelu danych oraz wybór metody jaką będziemy te dane analizać ma kluczowy wpływ na wyniki jakie otrzymamy. Zadna z uzytych metod nie dała idealnych wyników, ale metoda liniowa najmniejszych kwadratów sprawdziła się lepiej.

2.3 Bibliografia

http://heath.cs.illinois.edu/scicomp/notes/cs450_chapt03.pdf

https://numpy.org/doc/stable/reference/generated/numpy.linalg.solve.html

prezentacja Least squares metod Marcin Kuta