SPRAWOZDANIE

Zajęcia: Matematyka konkretna

Prowadzący: prof. dr hab. Inż. Vasyl Martsenyuk

|  |  |
| --- | --- |
| Laboratorium Nr 3  Data 23.10.2025 **Temat:** „Macierz Pseudoodwrotna. Najmniejsze  kwadraty. Regresja”  Wariant 1 | Marika Daniszewska  Informatyka  II stopień, stacjonarne,  2 semestr, gr. 1a |

1. Polecenie: wariant 1 zadania

Celem ćwiczenia było wykonanie **wieloliniowej regresji** dla zależności:

z wykorzystaniem **macierzy** pseudoodwrotnej **(**SVD, Moore–Penrose**)**.  
Na podstawie danych z pliku war1.csv należało obliczyć współczynniki i , porównać wyniki uzyskane przy pomocy rozkładu SVD oraz funkcji np.linalg.pinv, a także przeanalizować jakość dopasowania (RMSE, R², liczba warunkowa).  
Wyniki należało przedstawić w postaci notatnika Jupyter .ipynb oraz opisać w sprawozdaniu PDF.

1. Opis programu opracowanego (kody źródłowe, zrzuty ekranu)

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from pathlib import Path

#0 Ścieżka do pliku

FILENAME = "war1.csv"

WAR1\_PATH = None

def resolve\_path():

    if isinstance(WAR1\_PATH, (str, Path)):

        p = Path(WAR1\_PATH)

        if p.exists():

            return p.resolve()

        raise FileNotFoundError(f"Nie ma pliku pod WAR1\_PATH: {p}")

    p = Path(FILENAME)

    if p.exists():

        return p.resolve()

    # prosta próba znalezienia w ./data lub ..

    for cand in [Path("data")/FILENAME, Path("..")/FILENAME, Path("..")/"data"/FILENAME]:

        if cand.exists():

            return cand.resolve()

    raise FileNotFoundError(

        f"Nie znalazłem '{FILENAME}'. Umieść plik obok notatnika albo ustaw WAR1\_PATH."

    )

path = resolve\_path()

print(f"Używam pliku: {path}")

#1 Wczytanie + konwersja polskich przecinków dziesiętnych

df = pd.read\_csv(path, sep=None, engine="python")

for col in df.columns:

    if not pd.api.types.is\_numeric\_dtype(df[col]):

        try:

            df[col] = (

                df[col].astype(str)

                        .str.replace(",", ".", regex=False)

                        .str.replace(" ", "", regex=False)

            )

            df[col] = pd.to\_numeric(df[col], errors="raise")

        except Exception:

            pass

#2 Wybór kolumn x1, x2, y

num\_cols = df.select\_dtypes(include=[np.number]).columns.tolist()

def pick\_col(cands, pool):

    for c in cands:

        if c in pool:

            return c

    return None

x1\_col = "x1" if "x1" in num\_cols else pick\_col(["x1","X1"], num\_cols)

x2\_col = "x2" if "x2" in num\_cols else pick\_col(["x2","X2"], num\_cols)

y\_col  = pick\_col(["y","Y","y2","target","output"], num\_cols)

if y\_col is None:

    rem = [c for c in num\_cols if c not in {x1\_col, x2\_col}]

    y\_col = rem[0] if rem else None

assert x1\_col and x2\_col and y\_col, f"Brak trzech kolumn numerycznych x1,x2,y. Numeryczne: {num\_cols}"

work = df[[x1\_col, x2\_col, y\_col]].copy()

work.columns = ["x1","x2","y"]

print("\nPodgląd danych (pierwsze 5 wierszy):")

display(work.head())

#3 Regresja LS bez interceptu (SVD i pinv)

A = work[["x1","x2"]].to\_numpy(dtype=float)

b = work["y"].to\_numpy(dtype=float).reshape(-1,1)

U, S, VT = np.linalg.svd(A, full\_matrices=False)

Sinv = np.diag(1.0 / S)

x\_svd = VT.T @ Sinv @ U.T @ b

a\_svd, b\_svd = float(x\_svd[0,0]), float(x\_svd[1,0])

x\_pinv = np.linalg.pinv(A) @ b

a\_pinv, b\_pinv = float(x\_pinv[0,0]), float(x\_pinv[1,0])

#4 Metryki

y\_hat = (A @ x\_svd).ravel()

residuals = (b.ravel() - y\_hat)

ss\_res = float(np.sum(residuals\*\*2))

ss\_tot = float(np.sum((b.ravel() - np.mean(b))\*\*2))

r2 = 1.0 - ss\_res/ss\_tot if ss\_tot > 0 else float("nan")

rmse = float(np.sqrt(np.mean(residuals\*\*2)))

cond\_number = float(np.max(S) / np.min(S))

print("\n=== Współczynniki ===")

print(f"SVD : a = {a\_svd:.6f}, b = {b\_svd:.6f}")

print(f"pinv: a = {a\_pinv:.6f}, b = {b\_pinv:.6f}")

print("\n=== Metryki ===")

print(f"RMSE = {rmse:.6f}")

print(f"R^2  = {r2}")

print(f"cond\_number(A) = {cond\_number:e}")

#5 Wykresy

plt.figure()

plt.plot(b.ravel(), label="y (rzeczyw.)")

plt.plot(y\_hat, label="y\_hat (regresja)")

plt.xlabel("próbka"); plt.ylabel("wartość"); plt.legend()

plt.show()

plt.figure()

plt.scatter(b.ravel(), y\_hat)

plt.xlabel("y (rzeczyw.)"); plt.ylabel("y\_hat (prognoza)")

minv = float(min(np.min(b), np.min(y\_hat))); maxv = float(max(np.max(b), np.max(y\_hat)))

plt.plot([minv, maxv], [minv, maxv])  # linia y = x

plt.title("Zgodność: im bliżej linii y=x, tym lepiej")

plt.show()

plt.figure()

plt.plot(residuals, marker="o", linestyle="")

plt.axhline(0)

plt.xlabel("próbka"); plt.ylabel("reszty")

plt.show()

plt.figure()

plt.hist(residuals, bins=20)

plt.xlabel("reszty"); plt.ylabel("liczność")

plt.show()

Obraz zawierający tekst, zrzut ekranu, Czcionka, design

Zawartość wygenerowana przez AI może być niepoprawna.

Github: <https://github.com/MarikaDan/MK/tree/main/Lab3>

1. Wnioski
   * Zastosowanie SVD umożliwia rozwiązanie układu równań również w przypadkach, gdy macierz nie jest kwadratowa lub jest osobliwa. Metoda ta zapewnia stabilność numeryczną i możliwość kontrolowania błędów związanych z dużą liczbą warunkową macierzy.
   * Macierz pseudoodwrotna Moore’a–Penrose’a pozwala na uzyskanie rozwiązania w sensie najmniejszych kwadratów, minimalizując normę błędu . Wyniki uzyskane metodą SVD i pinv były identyczne (różnice poniżej dokładności maszynowej).
   * W badanym pliku war1.csv wartości były bliskie zeru, co spowodowało brak zróżnicowania danych i w efekcie zerowe współczynniki oraz bardzo wysoką liczbę warunkową macierzy .  
     Oznacza to, że dane wejściowe nie pozwalają na dopasowanie modelu — układ jest źle uwarunkowany.
   * W praktyce, aby uzyskać sensowny model regresyjny, konieczne jest użycie danych o dostatecznej zmienności oraz niewspółliniowych zmiennych i .
   * Ćwiczenie pokazało, że metoda SVD stanowi uniwersalne i niezawodne narzędzie do rozwiązywania problemów regresji oraz analizy systemów o różnej liczbie równań i niewiadomych.