Tomasz Głąbek, 09.06.2025

MOWNIT laboratorium 11

Optymalizacja

Zadanie 1

Rozwiąż ponownie problem predykcji typu nowotworu (laboratorium 2), używając metody spadku wzdłuż gradientu (ang. gradient descent). Stałą uczącą możesz wyznaczyć na podstawie najmniejszej i największej wartości własnej macierzy $A^T A$. Porównaj uzyskane rozwiązanie z metodą najmniejszych kwadratów, biorąc pod uwagę następujące kryteria:

- Dokładność predykcji na zbiorze testowym
- Teoretyczną złożoność obliczeniową
- Czas obliczeń.

Rozwiązanie

Metoda najmniejszych kwadratów

Obliczenie macierzy wag metodą najmniejszych kwadratów jest identyczne jak w laboratorium 2 i wymaga rozwiązania równania:

$$A^T Ax = A^T B$$

Złożoność czasowa rozwiązania tego równania to $O(m^2 n + m^3)$, gdzie n to ilość punktów danych, a m to ilość parametrów próbki.

Metoda spadku wzdłuż gradientu

Żeby obliczenia były poprawne, musiałem przeskalować dane wejściowe:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

A_train_scaled = scaler.fit_transform(A_train)
A_validate_scaled = scaler.transform(A_validate)
```

Stałą uczenia wyznaczyłem za pomocą wartości własnych macierzy danych:

$$\gamma = \frac{2}{\lambda_{\max} + \lambda_{\min}}$$

Złożoność obliczeniowa to $O(nmk + m^3)$, gdzie k jest liczbą iteracji spadku gradientu, a n i m jak wyżej, ilość danych i liczba parametrów. Bierze się to z powtórzenia k razy liczenia gradientu, które ma złożoność $O(mn)$ i wyznaczenia wartości własnych ze złożonością O(m^3), chociaż w tym przypadku to nmk dominuje.

Wyniki

Dokładność metody najmniejszych kwadratów: 97.31%

Dokładność metody spadku wzdłuż gradientu: 94.23%

Zadanie 2

Należy wyznaczyć najkrótszą ścieżkę robota pomiędzy dwoma punktami $x^{(0)}$ i $x^{(n)}$. Problemem są przeszkody usytuowane na trasie robota, których należy unikać. Zadanie polega na minimalizacji

funkcja kosztu, która sprowadza problem nieliniowej optymalizacji z ograniczeniami do problemu nieograniczonej optymalizacji. Macierz $X \in R^{(n+1)\times 2}$ opisuje ścieżkę złożoną z $n+1$ punktów $x^{(0)}, x^{(1)}, x^{(2)}, ...x^{(n)}$. Każdy punkt posiada 2 współrzędne, $x^{(i)} \in R^2$. Punkty początkowy i końcowy ścieżki, $x^{(0)}$ i $x^{(n)}$ są ustalone. Punkty z przeszkodami (punkty o 2 współrzędnych), $r^{(i)}$ dane są w macierzy przeszkód $R \in R^{k\times 2}$. W celu optymalizacji ścieżki robota należy użyć metody największego spadku. Funkcja celu użyta do optymalizacji $F\left(x^{(0)}, x^{(1)}, ..., x^{(n)}\right)$ zdefiniowana jest jako:

$$F\left(x^{(0)}, x^{(1)}, ..., x^{(n)}\right) = \lambda_1 \sum_{i=0}^{n}\sum_{j=1}^{k} \frac{1}{\varepsilon + \left\|x^{(i)} - r^{(j)}\right\|_2^2} + \lambda_2 \sum_{i=0}^{n-1} \left\|x^{(i+1)} - x^{(i)}\right\|_2^2$$

Symbole użyte we wzorze mają następujące znaczenie:

- Stałe λ_1 i λ_2 określają wpływ każdego członu wyrażenia na wartość $F(X)$.
  - λ_1 określa wagę składnika zapobiegającego zbytniemu zbliżaniu się do przeszkody
  - λ_2 określa wagę składnika zapobiegającego tworzeniu bardzo długich ścieżek
- n jest liczbą odcinków, a $n+1$ liczbą punktów na trasie robota
- k jest liczbą przeszkód, których robot musi unikać
- Dodanie ϵ w mianowniku zapobiega dzieleniu przez zero

1. Wyprowadź wyrażenie na gradient ∇F funkcji celu F względem $x^{(i)}$:

$$\nabla F = \left[\frac{\partial F}{\partial x^{(0)}}, ..., \frac{\partial F}{\partial x^{(n)}}\right]$$

2. Opisz matematycznie i zaimplementuj kroki algorytmu największego spadku z przeszukiwaniem liniowym, który służy do minimalizacji funkcji celu F. Do przeszukiwania liniowego (ang. line search) użyj metody złotego podziału (ang. golden section search). W tym celu załóż, że F jest unimodalna (w rzeczywistości tak nie jest) i że można ustalić początkowy przedział, w którym znajduje się minimum.
3. Znajdź najkrótszą ścieżkę robota przy użyciu algorytmu zaimplementowanego w w poprzednim punktcie. Przyjmij następujące wartości parametrów:
   - $n = 20$, $k = 50$
   - $x^{(0)} = [0, 0]$, $x^{(n)} = [20, 20]$
   - $r^{(i)} \sim \mathcal{U}(0, 20) \times \mathcal{U}(0, 20)$
   - $\lambda_1 = \lambda_2 = 1$
   - $\varepsilon = 10^{-13}$
   - liczba iteracji = 400

Wyznaczenie gradientu

$$\nabla F = \left[\frac{\partial F}{\partial x^{(0)}}, ..., \frac{\partial F}{\partial x^{(n)}}\right]$$

$$\frac{\partial}{\partial x^{(i)}} \sum_{i=0}^{n}\sum_{j=1}^{k} \frac{1}{\varepsilon + \left\|x^{(i)} - r^{(j)}\right\|_2^2} = \sum_{j=1}^{k} -2\frac{x^{(i)} - r^{(j)}}{\left(\varepsilon + \left\|x^{(i)} - r^{(j)}\right\|_2^2\right)^2}$$

$$\frac{\partial}{\partial x^{(i)}} \sum_{i=0}^{n-1} \left\| x^{(i+1)} - x^{(i)} \right\|_2^2 = 2\left(x^{(i)} - x^{(i-1)}\right) + 2\left(x^{(i)} - x^{(i+1)}\right) = -2x^{(i-1)} + 4x^{(i)} - 2x^{(i+1)}$$

$$\frac{\partial F}{\partial x^{(i)}} = -2\lambda_1 \sum_{j=1}^{k} \frac{x^{(i)} - r^{(j)}}{\left(\varepsilon + \left\| x^{(i)} - r^{(j)} \right\|_2^2\right)^2} - 2x^{(i-1)} + 4x^{(i)} - 2x^{(i+1)}$$

Powyższe równanie jest właściwe dla $i \in [1, n-1]$, ale nam to wystarcza, bo ustalamy $\frac{\partial F}{\partial x^{(0)}} = \frac{\partial F}{\partial x^{(n)}} = 0$, żeby te punkty pozostały stacjonarne.

Algorytm największego spadku

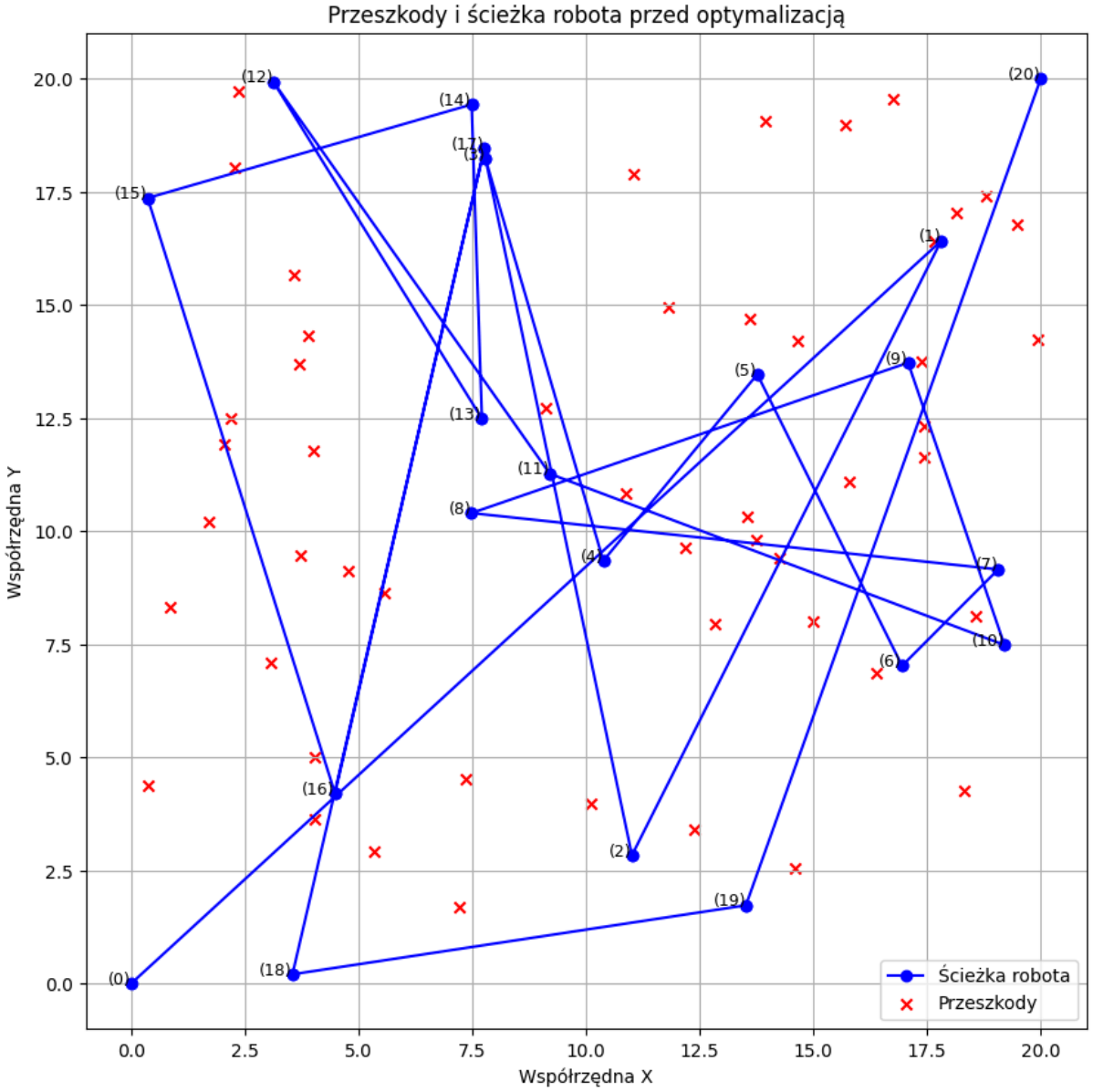
W każdej iteracji będziemy obniżać wartości wektora punktów o wartość gradientu ∇F w tym punkcie, przemnożonym przez stałą uczenia. Tą stałą wyznaczymy wzorem:

$$\gamma = \arg\min_{\gamma} F(X_k - \gamma \nabla F(X_k))$$

Żeby obliczyć γ użyjemy metody złotego podziału, która wymaga aby funkcja była unimodalna, czyli żeby miała tylko jedno minimum. Nasza funkcja nie spełnia tego założenia, ale przyjmujemy że spełnia. Skorzystam tutaj z funkcji `scipy.optimize.golden`.

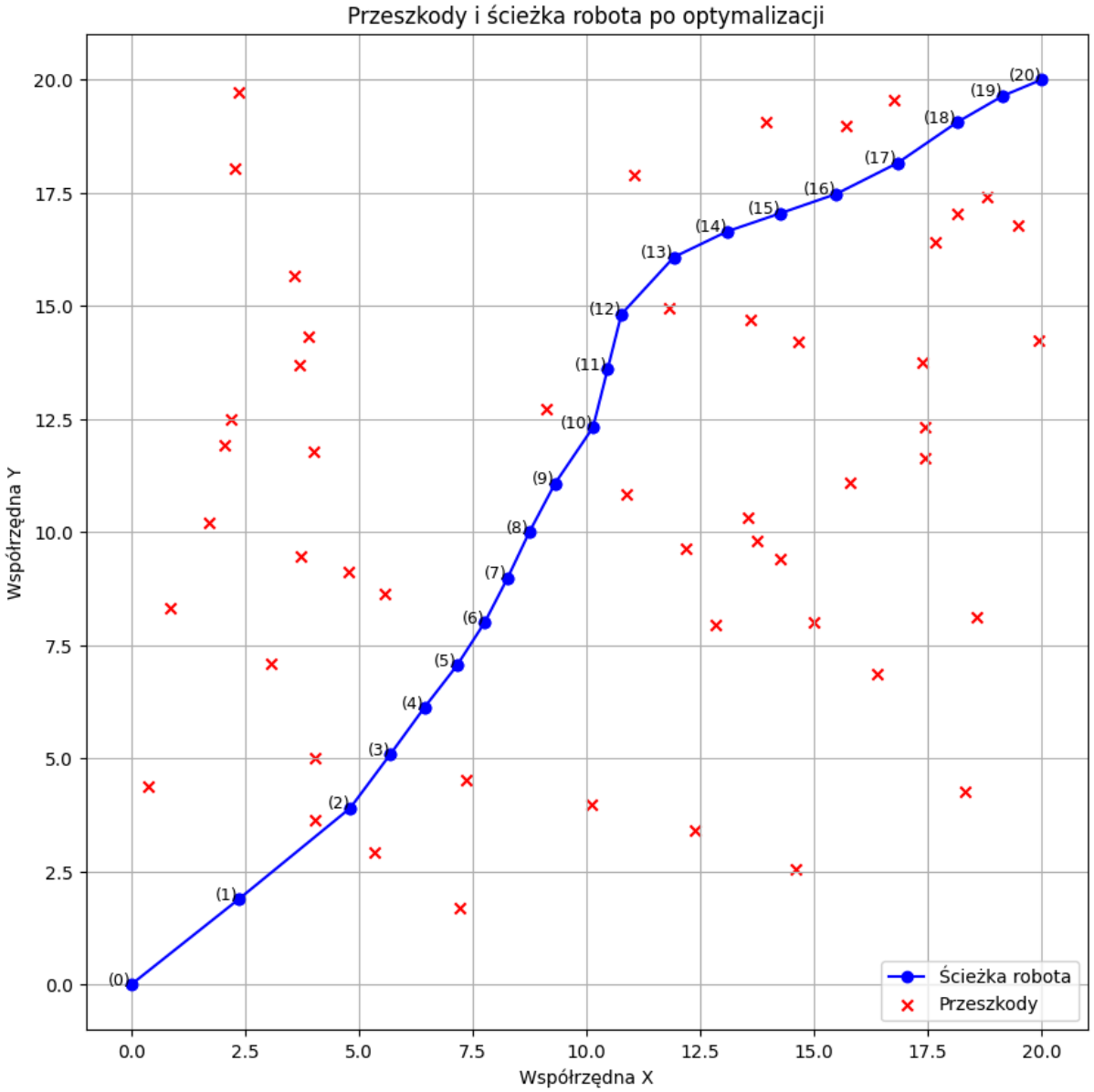
Wyniki obliczeń

Wykresy dla jednego z losowych ułożeń punktów, przed i po optymalizacji



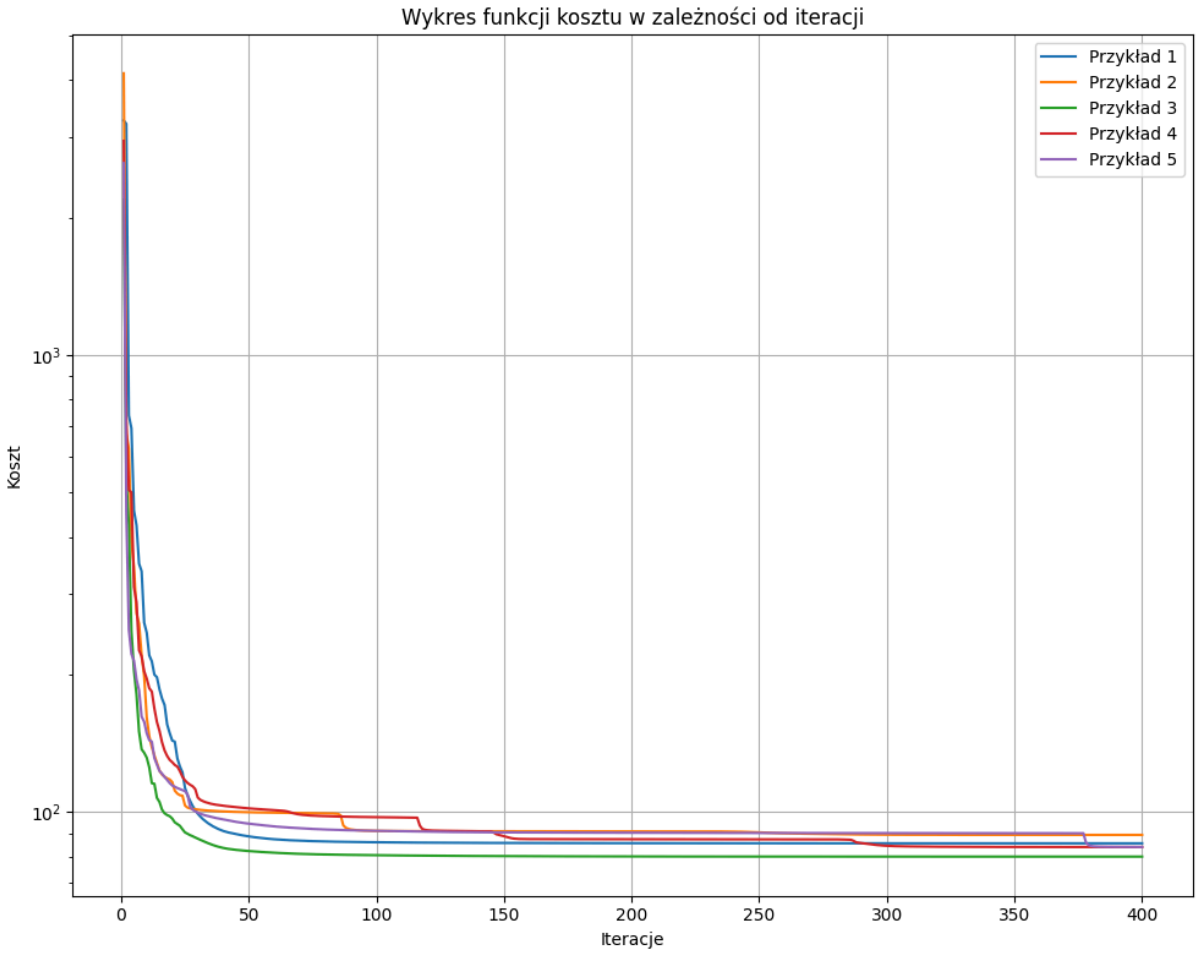
Rysunek 1: Ścieżka robota z losowo dobranymi punktami

Wartość funkcji kosztu: $F(X_0) \approx 3259.368$



Rysunek 2: Ścieżka robota po optymalizacji rozłożenia punktów

Wartość funkcji kosztu: $F(X_N) \approx 85.486$



Rysunek 3: Funkcje kosztu w zależności od iteracji dla 5 losowych wejściowych układów punktów

Wykresy pokazują że algorytm działa i znajduje odpowiednie punkty blisko siebie i z dala od przeszkód, zgodnie z funkcją kosztu. Na wykresie 3 można zobaczyć że nie ma znaczących różnic przy minimalizacji funkcji koszt dla różnych danych wejściowych. Widać również że 400 iteracji to przesada, bo już przy około 100 prawie nie widać zmian w wartości funkcji kosztu.

Wykresy dla pozostałych wejściowych ustawień punktów wyglądają podobnie i są dostępne w notebooku z kodem.

Wnioski

W dzisiejszym laboratorium porównaliśmy 2 metody optymalizacji funkcji. Metoda najmniejszych kwadratów może na pozór wypadać lepiej, dzięki swojej lepszej złożoności, dokładności (w tym porównaniu) i prostocie, jednak ma też swoje problemy: nie radzi sobie w dużych zbiorach danych i działa tylko dla problemów liniowych. Metoda spadku wzdłuż gradientu jest dużo bardziej uniwersalna, co pokazuje zadanie 2, którego nie da się rozwiązać stosując metodę najmniejszych kwadratów.

Źródła

- lab11-intro.pdf wysłany na Teamsach
- wykład „*Regresja Logistyczna i Wieloklasowa Regresja Logistyczna*” z przedmiotu Elementy Statystycznego Uczenia Maszynowego