Programowanie Matematyczne
Raport z laboratorium 3

Adam Przemysław Chojecki
298814, Grupa lab B

2023.11.10

1 Zadanie

Celem zadania jest rozwiązanie zagadnienia regresji za pomocą metod programowania matematycznego.

Dane mamy

$$n \in \mathbb{N}, x \in \mathbb{R}^n, y \in \mathbb{R}^n$$

Szukamy:

$$\min_{a\in\mathbb{R}, b\in\mathbb{R}} \max_{i\in\{1,2,\ldots,n\}} |\varepsilon_i|, \quad \text{gdzie} \quad \varepsilon_i = y_i - ax_i - b$$

To zadanie regresji jest równoważne następującemu zadaniowi liniowemu:

$$\min_{\Omega}(0 \cdot a + 0 \cdot b + 1 \cdot e_{\max})$$

gdzie:

$$\Omega = \begin{cases} ax_i + b - e_{\max} & \leqslant y_i, \quad i \in \{1,2,\ldots,n\} \\ -ax_i - b - e_{\max} & \leqslant -y_i, \quad i \in \{1,2,\ldots,n\} \\ a, b & \in \mathbb{R} \\ 0 \leqslant \quad e_{\max} & \end{cases}$$

1.1 Przekształcenie zadania do wygodniejszej postaci

W tym podrozdziale przekształcę zadanie programowania liniowego do prostszej postaci. Dla zwiększenia czytelności, każde przekształcenie będzie znajdowało się w oddzielnym pod-podrozdziale:

1.1.1 Pozbycie się ograniczeń na $e_{\max}$

Zauważmy, że dla $i = 1$ mamy ograniczenia:

- $ax_1 + b \leqslant y_1 + e_{\max}$

- $-ax_1 - b \leqslant -y_1 + e_{\max}$

a stąd wynika, że:

$$y_1 - e_{\max} \leqslant ax_1 + b \leqslant y_1 + e_{\max}$$

I dalej:

$$0 \leqslant 2 \cdot e_{\max}$$

Zatem nierówność $0 \leqslant e_{\max}$ wynika z poprzednich i można ją pominąć w definicji zbioru Ω.

Zatem mamy zadanie:

$$\min_{\Omega}(e_{\max})$$

gdzie:

$$\Omega = \begin{cases} ax_i + b - e_{\max} & \leqslant y_i, & i \in \{1, 2, \ldots, n\} \\ -ax_i - b - e_{\max} & \leqslant -y_i, & i \in \{1, 2, \ldots, n\} \\ a, b & \in \mathbb{R} \end{cases}$$

1.1.2 Dodatkowe ograniczenie na $e_{\max}$

Zauważmy, że szukamy minimalnego $e_{\max}$. Z interpretacji zadania widzimy, że dla $a = 0, b = 0$ mielibyśmy $e_{\max} = \max_{i \in \{1,2,\ldots,n\}} |y_i| =: y_{\max}$. Stąd można dodatkowo ograniczyć

$$e_{\max} \leqslant y_{\max}$$

Zatem mamy zadanie:

$$\min_{\Omega}(e_{\max})$$

gdzie:

$$\Omega = \begin{cases} ax_i + b - e_{\max} & \leqslant y_i, & i \in \{1, 2, \ldots, n\} \\ -ax_i - b - e_{\max} & \leqslant -y_i, & i \in \{1, 2, \ldots, n\} \\ a, b & \in \mathbb{R} \\ e_{\max} & \leqslant y_{\max} \end{cases}$$

1.1.3 Podstawienie e' zamiast $e_{\max}$

Możemy dokonać podstawienia $e' = y_{\max} - e_{\max}$. Wtedy ograniczeniem na e' jest

$$0 \leqslant e'$$

Zatem mamy zadanie:

$$\min_{\Omega'}(y_{\max} - e')$$

gdzie:

$$\Omega' = \begin{cases} ax_i + b - (y_{\max} - e') & \leqslant y_i, & i \in \{1, 2, \ldots, n\} \\ -ax_i - b - (y_{\max} - e') & \leqslant -y_i, & i \in \{1, 2, \ldots, n\} \\ a, b & \in \mathbb{R} \\ 0 \leqslant e' \end{cases}$$

1.1.4 zamiana zadania minimalizacji na zadanie maksymalizacji

Zadanie $\min_{\Omega'}(y_{\max} - e')$ jest równoważne zadaniowi $\max_{\Omega'}(e')$

Zatem mamy zadanie:

$$\max_{\Omega'}(e')$$

gdzie:

$$\Omega' = \begin{cases} ax_i + b - (y_{\max} - e') & \leqslant y_i, \quad i \in \{1, 2, \ldots, n\} \\ -ax_i - b - (y_{\max} - e') & \leqslant -y_i, \quad i \in \{1, 2, \ldots, n\} \\ a, b & \in \mathbb{R} \\ 0 \leqslant e' \end{cases}$$

1.1.5 W ograniczeniach przerzućmy $y_{\max}$ na prawą stronę nierówności

Zatem mamy zadanie:

$$\max_{\Omega'}(e')$$

gdzie:

$$\Omega' = \begin{cases} ax_i + b + e' & \leqslant y_i + y_{\max}, \quad i \in \{1, 2, \ldots, n\} \\ -ax_i - b + e' & \leqslant -y_i + y_{\max}, \quad i \in \{1, 2, \ldots, n\} \\ a, b & \in \mathbb{R} \\ 0 \leqslant e' \end{cases}$$

1.1.6 Ujednolicenie ograniczeń

Podstawiamy $a = a_1 - a_2$ oraz $b = b_1 - b_2$, gdzie nowe zmienne uważać będziemy za nieujemne.

Zatem mamy zadanie:

$$\max_{\Omega'}(e')$$

gdzie:

$$\Omega' = \begin{cases} (a_1 - a_2) \cdot x_i + (b_1 - b_2) + e' & \leqslant y_i + y_{\max}, \quad i \in \{1, 2, \ldots, n\} \\ -(a_1 - a_2) \cdot x_i - (b_1 - b_2) + e' & \leqslant -y_i + y_{\max}, \quad i \in \{1, 2, \ldots, n\} \\ 0 \leqslant & a_1, a_2, b_1, b_2, e' \end{cases}$$

1.2 Obserwacja

Zauważmy, że w ostatecznej interpretacji zadania jako zadania programowania liniowego mamy $2 \cdot n$ nierówności $\leqslant$ takie, że po prawej ich stronie stoi liczba nieujemna.

Zatem jest to Zadanie Programowania Liniowego (ZPL) w postaci kanonicznej. Zwykły algorytm Simplex będzie się nadawał na rozwiązanie tego zadania w tej formie. Bazowe rozwiązanie $(a_1, a_2, b_1, b_2, e') = (0, 0, 0, 0, 0)$ jest dopuszczalne, czyli jest BRD (bazowe rozwiązanie dopuszczalne).

2 Krótki opis algorytmu

Algorytm Simplex to skuteczna metoda rozwiązywania problemów programowania liniowego. Algorytm Simplex nie jest wielomianowy, ale zazwyczaj w praktycznych problemach okazuje się wielomianowy, patrz [1]. Działa poprzez iteracyjne przesuwanie się wzdłuż krawędzi simpleksu. Rozpoczyna się od dowolnego dopuszczalnego rozwiązania BRD. Iteracyjnie przemieszcza się w każdym kroku poprawiając wartość funkcji celu. Dzięki temu znajduje optimum lokalne, co w wypukłym zadaniu programowania liniowego jest równoważna znalezieniu optimum globalnego. Poniżej znajdują się kluczowe kroki algorytmu Simplex:

1. **Inicjalizacja:** Wybór początkowego dopuszczalnego rozwiązania i bazy zmiennych bazowych. Gdy w zadaniu trudno jest je ustalić, zaleca się użycie metody dwukrokowej.

2. **Wybór zmiennej wejściowej:** Wybór zmiennej niebazowej, której wartość można zwiększyć, aby poprawić wartość funkcji celu.

3. **Wybór zmiennej wyjściowej:** Wybór zmiennej bazowej, która opuszcza bazę, aby ustąpić miejsca nowej zmiennej wejściowej. Musi to być pierwsza zmienna, która spowoduje wysycenie jednego z ograniczeń. Dzięki temu nie wyjdziemy poza simpleks dopuszczalności.

4. **Aktualizacja bazy:** Aktualizacja bazy zmiennych bazowych i obliczenie nowego dopuszczalnego rozwiązania.

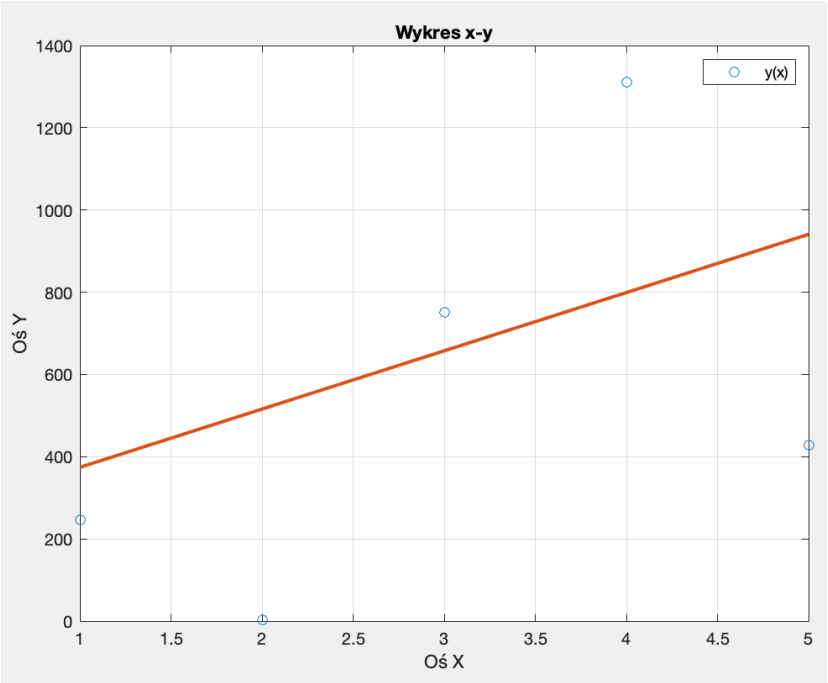
5. **Sprawdzenie warunku zakończenia:** Sprawdzenie warunków zakończenia, takich jak brak ujemnych kosztów dla zmiennych niebazowych.

6. **Powtarzanie kroków:** Powtarzanie kroków 3-6, dopóki nie osiągnięte zostanie optymalne rozwiązanie lub ustalony limit iteracji.

7. **Odczytanie rozwiązania:** Odczytanie optymalnego rozwiązania z ostatecznej tabeli sympleksowej.

3 Metodologia

W celu przetestowania działania algorytmu porównany zostanie wynik jego działania z wbudowaną do pakietu MATLAB funkcją `linprog()`. Porównanie to nastąpi na danych, gdzie $x = (1, 2, 3, 4, 5)$, natomiast y będzie losowymi liczbami całkowitymi z zakresu od 1 do 1500.



Rysunek 1: Rozwiązania zadania programowania liniowego na przykładowo wylosowanych danych.

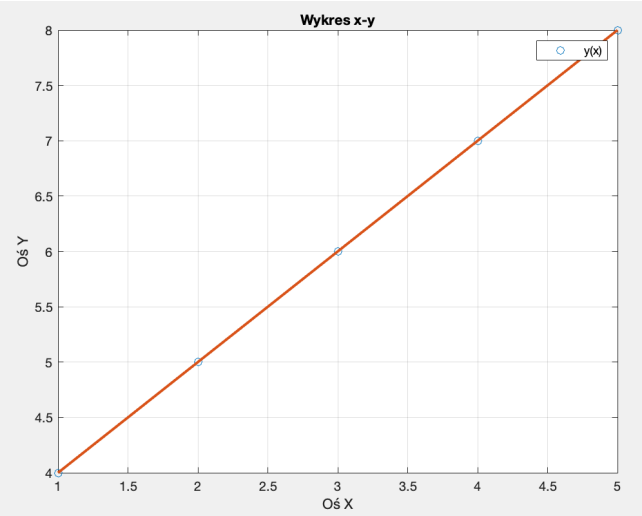
Na obrazku 1 znajduje się wizualizacja przykładowo wylosowanych danych wraz z rozwiązaniem tego ZPL.

Na takich danych porównano otrzymane wartości zmiennej decyzyjnej e'. Porównania dokonano 100 razy. Dodatkowo policzono i porównano wykonano liczbę iteracji algorytmów.

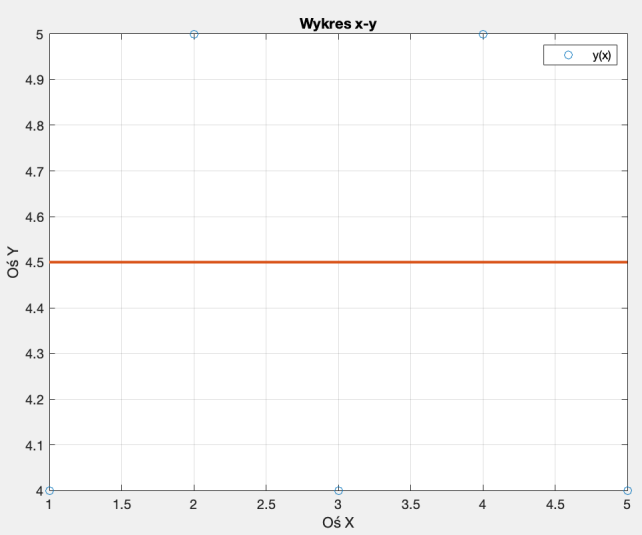
4 Ciekawe przykłady obliczeniowe

Sprawdzono jak algorytm zachowa się w ciekawych przypadkach.

- Na obrazku 2 znajduje się przykład w którym punkty układają się idealnie na linii, a więc rozwiązanie ma $e_{\max} = 0$. Algorytm prawidłowo znajduje to rozwiązanie.
- Na obrazku 3 znajduje się przykład w którym punkty są równoodległe od optymalnej prostej. Algorytm prawidłowo znajduje to rozwiązanie.



Rysunek 2: Rozwiązania zadania gdzie punkty układają się idealnie na linii. Oznacza to, ze optymalne rozwiązanie ma $e_{\max} = 0$.



Rysunek 3: Przykład, gdzie rozwiązanie optymalne ma wszystkie punkty równoodległe od prostej.

5 Wyniki

Zgodność wyniku implementowanej funkcji z funkcją wbudowaną w MATLAB wynosi 98%. Jest to wysoki i satysfakcjonujący wynik. Sprawdzono, że błędy

wynikają z zapętlenia się algorytmu w niektórych specyficznych przypadkach.

Gdy zaimplementowany algorytm się nie zapętlał, to średnio wykonywał 4.6 iteracji algorytmu Simplex. Dla porównania, dualny algorytm Simplex wbudowany w MATLAB wykonywał średnio 5.19 iteracji, więc bardzo zbliżoną liczbę.

6 Szczegóły techniczne

Cały kod rozwiązania znajduje się w jednym pliku `rozwiazanie.m`. Na jego górze w oddzielnych sekcjach znajdują się 4 testy - w tym oba przykłady z rozdziału 4. "Ciekawe przykłady obliczeniowe". W dalszej części skryptu znajduje się implementacja użytych funkcji. Komentarze informują w przystępny sposób zadanie każdej z funkcji oraz krótko opisują algorytm Symplex.

7 Podsumowanie

Udało się osiągnąć założenia projektu. Cel został osiągnięty, implementacja algorytmu działa, a wyniki są prawidłowe.

8 Oświadczenie o samodzielności

Oświadczam, że niniejsza praca stanowiąca podstawę do uznanie osiągnięcia efektów uczenia się z przedmiotu Programowanie Matematyczne została wykonana przeze mnie samodzielnie.

298814, Adam Przemysław Chojecki

Bibliografia

[1] Daniel A. Spielman i Shang-Hua Teng. *Smoothed Analysis of Algorithms: Why the Simplex Algorithm Usually Takes Polynomial Time*. 2003. arXiv: `cs/0111050 [cs.DS]`.