Laboratorium 11 - Optymalizacja

Jan Sarba, Dariusz Rozmus

09.06.2025

1 Treść zadań

1.1 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.

1.2 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 $X \in \mathbb{R}^{(n+1)\times 2}$ opisuje ścieżkę złożoną z $n+1$ punktów $x^{(0)}, x^{(1)}, \ldots, x^{(n)}$. Każdy punkt posiada 2 współrzędne, $x^{(i)} \in \mathbb{R}^2$. Punkty początkowy i końcowy ścieżki, $x^{(0)}$ i $x^{(n)}$, są ustalone. Punkty z przeszkodami (punkty o 2 współrzędnych), $r^{(j)}$ dane są w macierzy przeszkód $R \in \mathbb{R}^{k\times 2}$.

W celu optymalizacji ścieżki robota należy użyć metody największego spadku. Funkcja celu użyta do optymalizacji $F(x^{(0)}, x^{(1)}, \ldots, x^{(n)})$ zdefiniowana jest jako:

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

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.

W zadaniu należy wykonać następujące kroki:

1. Wyprowadź wyrażenie na gradient ∇F funkcji celu F względem $x^{(i)}$.
2. Opisz matematycznie i zaimplementuj kroki algorytmu największego spadku z przeszukiwaniem liniowym (metoda złotego podziału), który służy do minimalizacji funkcji celu F.
3. Znajdź najkrótszą ścieżkę robota przy użyciu zaimplementowanego algorytmu dla zadanych parametrów ($n = 20, k = 50, \lambda_1 = \lambda_2 = 1, \epsilon = 10^{-13}$, 400 iteracji), przeprowadzając obliczenia dla 5 różnych losowych inicjalizacji ścieżki.

2 Argumentacja

2.1 Zadanie 1

Celem zadania jest rozwiązanie problemu najmniejszych kwadratów za pomocą iteracyjnej metody spadku wzdłuż gradientu (Gradient Descent, GD) i porównanie jej z rozwiązaniem analitycznym (poprzez równania normalne), zaimplementowanym w laboratorium 2.

Metoda spadku wzdłuż gradientu
Problem minimalizacji błędu średniokwadratowego sprowadza się do znalezienia wektora wag w, który minimalizuje funkcję kosztu:

$$J(w) = \|Aw - b\|_2^2$$

Gradient tej funkcji względem w wynosi:

$$\nabla J(w) = 2A^T(Aw - b)$$

Algorytm GD aktualizuje wagi w w kolejnych krokach, poruszając się w kierunku przeciwnym do gradientu:

$$w_{k+1} = w_k - \alpha \nabla J(w_k)$$

gdzie α to stała ucząca (learning rate).

Wybór stałej uczącej
Kluczowym elementem metody jest dobór stałej uczącej α. Zbyt duża wartość może prowadzić do dywergencji, a zbyt mała do bardzo wolnej zbieżności. Dla problemów kwadratowych, jak ten, optymalną stałą uczącą można wyznaczyć na podstawie wartości własnych macierzy Hessego, która wynosi $2A^TA$. Zgodnie z treścią zadania, stała α została wyznaczona jako:

$$\alpha = \frac{1}{\lambda_{max}(A^TA) + \lambda_{min}(A^TA)}$$

gdzie λ_{max} i λ_{min} to odpowiednio największa i najmniejsza wartość własna macierzy A^TA. Taki wybór α gwarantuje zbieżność algorytmu.

Implementacja
Proces implementacji obejmował następujące kroki:

1. Przygotowanie macierzy cech A oraz wektora docelowego b dla modelu liniowego (30 cech) i kwadratowego (14 cech).
2. Obliczenie macierzy A^TA dla obu modeli.
3. Wyznaczenie wartości własnych macierzy A^TA i obliczenie stałej uczącej α.

4. Zaimplementowanie pętli algorytmu GD, inicjalizując wektor wag w zerami i aktualizując go w każdej iteracji, aż do osiągnięcia zbieżności (mała zmiana normy wektora wag) lub maksymalnej liczby iteracji.

```python
def solve_gradient_descent(A, b, learning_rate, iterations=10000, tol=1e-6):
    """Rozwiazuje problem najmniejszych kwadratow metoda spadku gradientu."""
    w = np.zeros(A.shape[1])
    for i in range(iterations):
        gradient = 2 * A.T @ (A @ w - b)
        w_new = w - learning_rate * gradient
        if np.linalg.norm(w_new - w) < tol:
            break
        w = w_new
    return w
```

Listing 1: Implementacja algorytmu spadku wzdłuż gradientu.

2.2 Zadanie 2

Celem zadania jest znalezienie optymalnej ścieżki robota, minimalizując zadaną funkcję kosztu $F(X)$. Problem ten został rozwiązany przy użyciu metody największego spadku z przeszukiwaniem liniowym.

Gradient funkcji celu

Pierwszym krokiem jest analityczne wyznaczenie gradientu funkcji celu F. Ze względu na liniowość operatora gradientu, możemy rozbić funkcję celu na dwa składniki i obliczyć ich gradienty osobno. Zdefiniujmy:

- Funkcję kary za bliskość przeszkód: $F_{obs}(X) = \sum_{i=0}^{n} \sum_{j=1}^{k} \frac{1}{\epsilon + \|x^{(i)} - r^{(j)}\|_2^2}$
- Funkcję kary za długość ścieżki: $F_{len}(X) = \sum_{i=0}^{n-1} \|x^{(i+1)} - x^{(i)}\|_2^2$

Całkowita funkcja kosztu to $F(X) = \lambda_1 F_{obs}(X) + \lambda_2 F_{len}(X)$. Szukamy jej gradientu względem współrzędnych i-tego punktu na ścieżce, $x^{(i)}$ (dla $i \in \{1, \ldots, n-1\}$, gdyż punkty krańcowe są stałe).

Gradient składnika F_{obs} względem $x^{(i)}$: Tylko te składniki sumy, które zawierają $x^{(i)}$, mają niezerową pochodną.

$$\nabla_{x^{(i)}} F_{obs}(X) = \nabla_{x^{(i)}} \left(\sum_{j=1}^{k} \frac{1}{\epsilon + \|x^{(i)} - r^{(j)}\|_2^2} \right) = \sum_{j=1}^{k} \frac{-2(x^{(i)} - r^{(j)})}{(\epsilon + \|x^{(i)} - r^{(j)}\|_2^2)^2}$$

Gradient składnika F_{len} względem $x^{(i)}$: W tym przypadku niezerowy wkład wnoszą tylko dwa człony sumy: $\|x^{(i)} - x^{(i-1)}\|_2^2$ oraz $\|x^{(i+1)} - x^{(i)}\|_2^2$.

$$\nabla_{x^{(i)}} F_{len}(X) = \nabla_{x^{(i)}} \left(\|x^{(i)} - x^{(i-1)}\|_2^2 + \|x^{(i+1)} - x^{(i)}\|_2^2 \right)$$

$$= 2(x^{(i)} - x^{(i-1)}) - 2(x^{(i+1)} - x^{(i)}) = 2(2x^{(i)} - x^{(i-1)} - x^{(i+1)})$$

Ostatecznie, pełny gradient funkcji F względem $x^{(i)}$ wynosi:

$$\nabla_{x^{(i)}} F(X) = \lambda_1 \nabla_{x^{(i)}} F_{obs}(X) + \lambda_2 \nabla_{x^{(i)}} F_{len}(X)$$

```
# Obliczenie gradientu dla jednego punktu wewnetrznego x_i
diffs_obs = path[i, :] - obstacles
denom = (epsilon + np.sum(diffs_obs ** 2, axis=1)) ** 2
grad_obs = -2 * lambda1 * np.sum(diffs_obs / denom[:, np.
    newaxis], axis=0)

grad_len = 2 * lambda2 * (2 * path[i, :] - path[i-1, :] - path[
    i+1, :])

grad[i, :] = grad_obs + grad_len
```

Listing 2: Implementacja obliczania gradientu funkcji celu F dla jednego punktu.

Algorytm optymalizacyjny

Zastosowano algorytm największego spadku (gradient descent). W każdej iteracji k cała ścieżka X (czyli macierz współrzędnych wszystkich punktów $x^{(i)}$) jest aktualizowana:

$$X_{k+1} = X_k - \alpha_k \nabla F(X_k)$$

Wektor $\nabla F(X_k)$ zawiera gradienty obliczone dla każdego punktu wewnętrznego $x^{(i)}$.

Przeszukiwanie liniowe - Metoda Złotego Podziału

Współczynnik kroku α_k nie jest stały, lecz jest dynamicznie dobierany w każdej iteracji za pomocą metody przeszukiwania liniowego. Zgodnie z treścią zadania, użyto metody złotego podziału (*golden section search*). Metoda ta służy do znalezienia minimum funkcji jednej zmiennej $f(\alpha) = F(X_k - \alpha \nabla F(X_k))$ w zadanym przedziale $[a, b]$. Działa ona poprzez iteracyjne zawężanie przedziału poszukiwań, aż do osiągnięcia zadanej tolerancji. W implementacji przyjęto stały początkowy przedział poszukiwań $[0, 0.1]$.

```
# Petla algorytmu najwiekszego spadku
for j in range(ITERATIONS):
    # 1. Oblicz gradient w biezacym punkcie (dla calej sciezki)
    grad = calculate_gradient(path, obstacles, LAMBDA_1,
        LAMBDA_2, EPSILON)

```

```
    # 2. Zdefiniuj funkcje 1D do minimalizacji wzgledem alfa
    line_search_func = lambda alpha: cost_function(path - alpha
        * grad, obstacles, LAMBDA_1, LAMBDA_2, EPSILON)

    # 3. Znajdz optymalne alfa uzywajac metody zlotego podzialu
    alpha = golden_section_search(line_search_func, 0, 0.1)

    # 4. Zaktualizuj sciezke
    path = path - alpha * grad
```

Listing 3: Główna pętla algorytmu optymalizacyjnego z przeszukiwaniem liniowym.

3 Wyniki

3.1 Zadanie 1

Poniżej zestawiono wyniki uzyskane za pomocą metody spadku gradientu (GD) oraz, dla porównania, wyniki z metod analitycznych (Least Squares, LS) z poprzedniego laboratorium.

Tabela 1: Porównanie wyników metod dla modeli liniowego i kwadratowego.

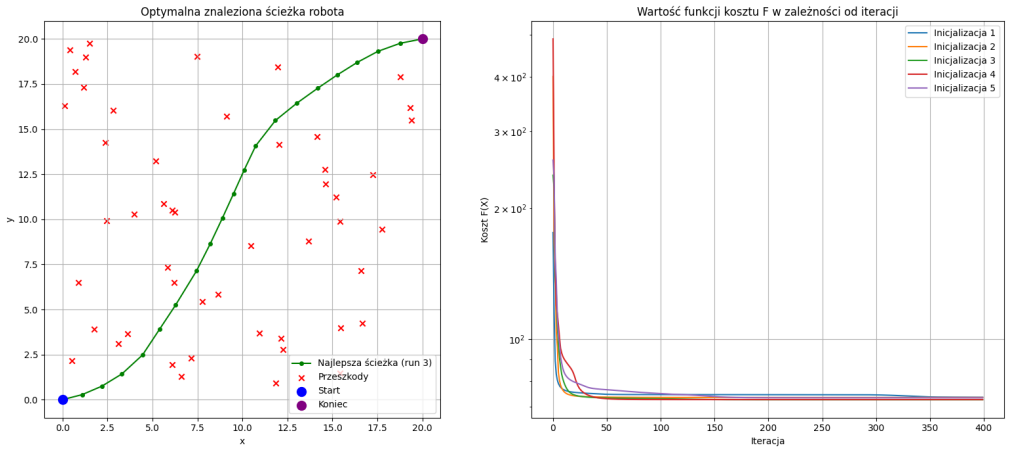
Model	**Metoda**	**Dokładność (Accuracy)**	**Czas obliczeń [s]**
Liniowy	LS (Równania Normalne)	0.9691	0.00010
Liniowy	LS (SVD)	0.9691	0.00744
Liniowy	LS (Ridge)	**0.9768**	0.00004
Liniowy	Gradient Descent (GD)	0.9154	0.08540
Kwadratowy	LS (Równania Normalne)	0.9228	0.00063
Kwadratowy	Gradient Descent (GD)	0.7692	0.00010

Analiza wyników

- Najwyższą dokładność (97.7%) osiągnęła analityczna metoda najmniejszych kwadratów z regularyzacją Ridge dla modelu liniowego. Standardowe równania normalne oraz SVD dla modelu liniowego dały bardzo dobrą, identyczną dokładność (96.9%). Metoda spadku gradientu dla modelu liniowego osiągnęła zauważalnie niższy, choć wciąż przyzwoity wynik (91.5%). Model kwadratowy w każdej wersji okazał się gorszy, a jego implementacja za pomocą GD całkowicie zawiodła.
- Metody analityczne (LS, Ridge) są ekstremalnie szybkie dla tego rozmiaru problemu, z czasami rzędu setnych części milisekundy. Metoda oparta na SVD jest wolniejsza o rząd wielkości. Najwolniejsza okazała się metoda spadku gradientu dla modelu liniowego (ok. 85 ms), co wynika z jej iteracyjnego charakteru. Paradoksalnie, GD dla modelu kwadratowego zakończyło się niemal natychmiast, ale było to spowodowane brakiem postępu w optymalizacji.

3.2 Zadanie 2

Algorytm optymalizacji ścieżki został uruchomiony dla 5 różnych losowych inicjalizacji punktów wewnętrznych. Poniższy wykres przedstawia uzyskane rezultaty.



Rysunek 1: Wyniki optymalizacji ścieżki. **Po lewej:** Najlepsza znaleziona ścieżka (o najniższym koszcie końcowym) spośród 5 losowych inicjalizacji. **Po prawej:** Zmiana wartości funkcji kosztu F w kolejnych iteracjach dla każdej z 5 inicjalizacji (w skali logarytmicznej).

4 Wnioski

Zadanie 1:

- Dla analizowanego problemu, analityczna metoda najmniejszych kwadratów (zwłaszcza z regularyzacją Ridge) jest zdecydowanie lepszym wyborem. Oferuje najwyższą dokładność przy znikomym koszcie obliczeniowym. Iteracyjna metoda spadku gradientu, mimo że teoretycznie prowadzi do tego samego rozwiązania, w praktyce jest znacznie wolniejsza i osiąga gorszą dokładność.
- Wyniki, zwłaszcza dla modelu kwadratowego, dobitnie pokazują wrażliwość metody GD na złe uwarunkowanie problemu. Dodanie cech wielomianowych wprowadziło silną współliniowość, co doprowadziło do bardzo dużej liczby uwarunkowania macierzy A^TA. Skutkowało to wyborem ekstremalnie małej stałej uczącej, co uniemożliwiło algorytmowi efektywną optymalizację i doprowadziło do całkowitej porażki.
- W przypadku problemów regresji liniowej z małą i umiarkowaną liczbą cech, metody analityczne (LS, SVD, Ridge) są preferowane. Są szybkie, proste w implementacji i dają dokładne rozwiązanie bez potrzeby strojenia hiperparametrów, takich jak stała ucząca. Metody iteracyjne, takie jak GD, są niezbędne w scenariuszach z ogromną liczbą cech (gdzie A^TA nie zmieściłoby się w pamięci lub jej odwrócenie byłoby zbyt kosztowne) lub w kontekstach uczenia online.

Zadanie 2:

- Zaimplementowany algorytm największego spadku z przeszukiwaniem liniowym (metodą złotego podziału) okazał się skuteczny w minimalizacji złożonej, nieliniowej funkcji kosztu. Wykresy pokazują, że algorytm konsekwentnie redukuje wartość funkcji kosztu, co prowadzi do znalezienia ścieżki, która unika przeszkód i jednocześnie nie jest nadmiernie długa. Jednocześnie, zyski szybko maleją.
- Wyniki uzyskane dla pięciu różnych losowych inicjalizacji pokazują, że końcowy koszt i kształt ścieżki zależą od punktu startowego optymalizacji, choć w tym przypadku różnica nie jest szczególnie znacząca.

5 Bibliografia

- prof. Heath M. T. – CS 450—numerical analysis, Chapter 6