Lab6c

Wojciech Klusek 305943 grupa C

3 stycznia 2023

Spis treści

1 Zadanie

Zadaniem było znalezienie najbliższego punktu od początku układu współrzędnych spełniającego ogarniczenia:

$$\Omega = \{x \in R^n : Ax = b, x \geqslant 0, A \in R^{mxn}, m < n, r(A) = m, b \in R^m\}$$

Macierz A powyżej zawiera wartości całkowite z przedziału [-5, 5],

$$n = 10$$

$$m = 3, 5, 7$$

Celem jest zminimalizowanie odległości punktu x od początku układu współrzędnych, co można przedstawić jako minimalizację funkcji kwadratowej:

$$\min_{x \in \Omega} \|x\|^2$$

Przekształcając, otrzymujemy:

$$\min_{x \in \Omega} x^T x$$

Własności:

- Zadanie jest wypukłe, ponieważ funkcja celu (norma kwadratowa) jest funkcją wypukłą, a ograniczenia są liniowe.
- W niektórych przypadkach, zależnie od konkretnych wartości w A i b, może nie istnieć rozwiązanie spełniające wszystkie ograniczenia.

Z własności powyżej wynika, że warunki KT są nie tylko niezbędne, ale również wystarczające dla znalezienia RO. Jeśli zostanie znaleziony punkt, który spełnia warunki KT, oznacza to, że jest to RO problemu optymalizacyjnego.

2 Opis algorytmów

2.1 Zewnętrzna funkcja kary

Funkcja kary f oraz jej gradient ∇f są dane przez:

$$f = \|\mathbf{x}\|^2 + \text{penalty} \cdot \|\mathbf{Ax} - \mathbf{b}\|^2 + \text{penalty} \cdot \|\max(0, -\mathbf{x})\|^2,$$
$$\nabla f = 2\left(\mathbf{x} + \text{penalty} \cdot \mathbf{A}^T(\mathbf{Ax} - \mathbf{b}) - \text{penalty} \cdot \max(0, -\mathbf{x})\right).$$

2.2 Algorytm Neldera-Meada

Algorithm 1 Nelder-Mead Algorithm

```
function NelderMead(A, b, x_0, e)
    Inicjalizuj parametry (α, γ, ρ, σ, penalty)
    Ustaw max_iter i it ← 0
    Inicjalizuj simpleks z x_0
    while it < max_iter do
        Oceń funkcję kary dla każdego punktu w simpleksie
        Sortuj simpleks wg wartości kary
        if kryterium zakończenia spełnione then
            break
        end if
        Oblicz środek ciężkości simpleksu bez najgorszego punktu
        Wykonaj odbicie najgorszego punktu
        if odbicie poprawia rozwiązanie then
            Rozważ ekspansję
        else if odbicie nie poprawia wystarczająco then
            Wykonaj kontrakcję
            if kontrakcja nieudana then
                Skurcz simpleks
            end if
        end if
        Aktualizuj parametr kary
        it ← it + 1
    end while
    xx ← simpleks[:, 1]
end function
```

3 Rozwiązanie

Rozwiązanie składa się z 6 plików:

1. `quadprog_solution.m` - Rozwiązanie z wykorzystaniem funkcji `quadprog`. Przyjmuje jako wejście:

   - `A`,
   - `b`,

   Funkcja zwraca wektor `xx` będący rozwiązaniem zadania oraz `exitflag` określającą czy rozwiązanie zostało znalezione.

2. `penalty_function.m` - Definicja funkcji kary z jej gradientem. Przyjmuje jako wejście:

   - `x`,

- `A`,
- `b`,
- `penalty` - kara,

Funkcja zwraca wartość funkcji kary `f` oraz wartość gradientu funkcji kary `grad`.

3. `ZFK.m` - Rozwiązanie z wykorzystaniem zewnętrznej funkcji kary i `fminunc`. Przyjmuje jako wejście:
   - `A`,
   - `b`,
   - `x0` - rozwiązanie początkowe,
   - `e` - dokładność obliczeń,

   Funkcja zwraca wektor `xx` będący rozwiązaniem zadania, `exitflag` określającą czy rozwiązanie zostało znalezione oraz `it` określającą liczbę wykonanych iteracji.

4. `NM.m` - Rozwiązanie z wykorzystaniem algorytmu Neldera-Meada. Przyjmuje jako wejście:
   - `A`,
   - `b`,
   - `x0` - rozwiązanie początkowe,
   - `e` - dokładność obliczeń,

   Funkcja zwraca wektor `xx` będący rozwiązaniem zadania, `exitflag` określającą czy rozwiązanie zostało znalezione oraz `it` określającą liczbę wykonanych iteracji.

5. `test.m` - Porównanie rozwiązań `quadprog`, `ZFK` oraz `NM`.

6. `generate.m` - Generuje macierz A, wektor b. Przyjmuje jako wejście zmienne:
   - `m` - liczba wierszy,
   - `n` - liczba kolumn.

   Funkcja zwraca macierz A, wektor b zgodnie z zadaniem.

4 Sprawdzenie optymalności rozwiązania

Warunki KKT są wystarczające dla RO (zgodnie z własnościami przedstawionymi w sekcji Zadanie). Sprawdźmy czy rozwiązania zwrócone przez `quadprog` oraz `ZFK` są RO:

Problem optymalizacyjny zdefiniowany jako:

$$\begin{aligned} \min \quad & \|x\|^2 \\ \text{s.t.} \quad & Ax = b, \\ & x \geqslant 0, \end{aligned}$$

Lagranżjan dla tego problemu jest dany przez:

$$\mathcal{L}(x, \lambda, \mu) = x^T x + \lambda^T (Ax - b) - \mu^T x,$$

gdzie λ i μ są wektorami mnożników Lagrange'a.

Warunki Karusha-Kuhna-Tuckera dla tego problemu są następujące:

1. $$\frac{\partial \mathcal{L}}{\partial x} = 2x + A^T \lambda - \mu = 0.$$

2. $$Ax = b, \quad x \geqslant 0.$$

3. $$\mu \geqslant 0.$$

4. $$\mu x = 0.$$

Niech:

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 1 & 2 & -1 \end{bmatrix} \quad b = \begin{bmatrix} 3.5 \\ 1.5 \end{bmatrix}$$

Oba algorytmy zwróciły rozwiązanie: $x = [0.5, 1, 1]$. Sprawdźmy każdy z warunków:

1. $2x + A^T \lambda - \mu$.

   Jako, że z warunku 4 mamy $\mu = 0$ otrzymujemy następujący układ równań:

   $$3\lambda_1 + \lambda_2 = -1$$
   $$2\lambda_1 + 2\lambda_2 = -2$$
   $$\lambda_1 - \lambda_2 = -2$$

   Po rozwiązaniu powyższego układu otrzymujemy: $\lambda_1 = -1.5, \lambda_2 = 0.5$, więc warunek spełniony.

2. $Ax = b, \quad x \geqslant 0.$

   Dla $x = [0.5, 1, 1]$ warunki są spełnione.

3. $\mu \geqslant 0.$

   Jako, że z warunku 4 $\mu = 0$ warunek spełniony.

4. $\mu x = 0$.

   Jako, że $x_1 = 0.5$, $x_2 = 1$ oraz $x_3 = 1$ mamy $\mu = 0$.

Jak widać wszystkie warunki są spełnione więc znaleziony punkt jest RO dla tego zadania.

5 Test dokładności rozwiązania

Do testu dokładności wykorzystane zostały następujące zmienne:

- Liczba kolumn (n) - 10,
- Liczba różnych wejść - 100,
- Ziarno generatora liczb losowych - 10,
- Dokładność obliczeń - $1e - 4$,

Poniżej przedstawione zostały wyniki dla powyższych parametrów:

5.1 m = 3

5.1.1 quadprog

- Liczba rozwiązań spełniających warunki: 100.
- Średnia norma rozwiązania (dla wejścia dla którego wszystkie algorytmy znalazły rozwiązanie): 2.148899.

5.1.2 ZFK

- Liczba rozwiązań spełniających warunki: 99.
- Średnia norma rozwiązania: 2.148859.
- Średnia norma różnicy rozwiązania z `quadprog`: 0.000185.
- Średnia liczba iteracji: 233.

5.1.3 NM

- Liczba rozwiązań spełniających warunki: 99.
- Średnia norma rozwiązania: 2.784798.
- Średnia norma różnicy rozwiązania z `quadprog`: 0.997913.
- Średnia liczba iteracji: 986.

5.2 m = 5

5.2.1 quadprog

- Liczba rozwiązań spełniających warunki: 99.
- Średnia norma rozwiązania: 3.133513.

5.2.2 ZFK

- Liczba rozwiązań spełniających warunki: 69.
- Średnia norma rozwiązania: 3.133350.
- Średnia norma różnicy rozwiązania z `quadprog`: 0.000481.
- Średnia liczba iteracji: 280.

5.2.3 NM

- Liczba rozwiązań spełniających warunki: 69.
- Średnia norma rozwiązania: 3.930537.
- Średnia norma różnicy rozwiązania z `quadprog`: 1.257470.
- Średnia liczba iteracji: 1243.

5.3 m = 7

5.3.1 quadprog

- Liczba rozwiązań spełniających warunki: 97.
- Średnia norma rozwiązania: 7.874762.

5.3.2 ZFK

- Liczba rozwiązań spełniających warunki: 19.
- Średnia norma rozwiązania: 7.874028.
- Średnia norma różnicy rozwiązania z `quadprog`: 0.000973.
- Średnia liczba iteracji: 220.

5.3.3 NM

- Liczba rozwiązań spełniających warunki: 19.
- Średnia norma rozwiązania: 8.203999.
- Średnia norma różnicy rozwiązania z `quadprog`: 0.805527.
- Średnia liczba iteracji: 1127.

6 Wnioski

Na podstawie przeprowadzonych testów można wyciągnąć wnioski dotyczące efektywności i dokładności trzech zbadanych algorytmów: quadprog, metody z zewnętrzną funkcją kary (ZFK) i algorytmu Neldera-Meada (NM). Algorytm quadprog wykazał się najwyższą skutecznością w znalezieniu rozwiązań spełniających warunki zadania. Metoda ZFK osiągała wyniki bardzo zbliżone do quadprog, jednak była mniej skuteczna w niektórych przypadkach, zwłaszcza przy większych wartościach m. Algorytm NM z kolei wykazał niższą precyzję w porównaniu z quadprog, co manifestowało się w większych średnich normach rozwiązania. Dodatkowo, NM wymagał znacznie więcej iteracji, co wskazuje na jego niższą efektywność obliczeniową, zwłaszcza w przypadkach bardziej skomplikowanych zestawów danych. Podsumowując algorytm ZFK wykazał się zbliżoną skutecznością co quadprog dla małych m, a algorytm NM był nieco gorszy. Algorytmy ZFK oraz NM radziły sobie gorzej niż quadprog dla dużego m w znajdowaniu rozwiązań spełnających założenia.

7 Oświadczenie o samodzielności

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

Wojciech Klusek 305934