WSI Zadanie laboratoryjne 1.

Metoda gradientu prostego

Celem zadania jest:

- implementacja algorytmu gradientu prostego
- wykorzystanie algorytmu do zbadania zbieżności algorytmu dla różnych parametrów kroku
- wyciągnięcie wniosków z rezultatu badań

Opis metody gradientu prostego:

Metoda gradientu prostego ma za zadanie znaleźć minimum lokalne funkcji.

1. Wybierany jest punkt startowy x_0.
2. W tym punkcie obliczmy kierunek poszukiwań, który w tej metodzie jest antygradientem funkcji celu $-\nabla F(x_n)$.
3. Punkt w następnym kroku obliczany jest wzorem $x_{n+1}=x_n-\beta*-\nabla F(x_n)$, gdzie β to parametr kroku.
4. Jeżeli Punkt ten nie spełnia warunku stopu powtarzamy czynności dla tego punktu od 2 podpunktu

W algorytmie zastosuję 2 warunki stopu:

- $||\nabla F(x_n)|| <= \varepsilon$ (gdy gradient jest bliski 0)
- $||x_{n+1}-x_n|| <= \varepsilon$ (gdy różnica między kolejnymi punktami jest bliska zeru)

Oba te warunki sprawdzają, czy algorytm doszedł do minimum lokalnego z dokładnością ε.

W algorytmie zastosuję również parametr max_iter mający przeciwdziałać zbyt długiemu działaniu algorytmu. Jeżeli powodem zatrzymania algorytmu jest przekroczenie max_iter oznacza to, że algorytm nie doszedł do minimum lokalnego.

Algorytm da się zapisać przy pomocy pseudokodu w następujący sposób:

F – funkcja celu

x_0 – punkt startowy

β – parametr kroku

ε - precyzja

max_iter – maksymalna ilość kroków

```
gradient_prosty(F, x0, β, ε, max_iter):
        powtórz max_iter razy:
                gradient = ∇F(xn)
                Jeżeli gradient <= ε:
                        Zakończ pętlę
                xn+1=xn-β*gradient
                Jeżeli xn+1-xn <= ε:
                        Zakończ pętlę
                xn = xn+1
        Zwróć xn
```

Implementacja algorytmu gradientu prostego oraz programu użytego do wygenerowania grafów w języku python znajduje się w dołączonych plikach źródłowych.

Opis przeprowadzonych eksperymentów numerycznych:

Za funkcję celu w moich rozważaniach przyjmuję funkcję $\mathbb{R}^n \rightarrow \mathbb{R}$:

$$g(x) = \sum_{i=1}^{n} \alpha^{\frac{i-1}{n-1}} x_i^2, \ x \in [-100, 100]^n \subset \mathbb{R}, n = 10$$

Funkcja rozdzielona na 3 warianty dla wartości parametru α = {1, 10, 100}

Za punkt startowy przyjmuję punkt (100, 100, … , 100) o n współrzędnych. Nie losuję ich, aby losowość punktu nie wpływała na rozważania

Maksymalną liczbę iteracji ustawiam na 100

Precyzję ustawiam na 1e-6

Celem jest zbadanie wpływu wartości parametru kroku na działanie funkcji gradientu prostego, zamierzam również sprawdzać powód zatrzymania funkcji, oraz czas jej działania.

W tym celu tworzę 3 wykresy (lin-log), iteracji funkcji od wartości funkcji g w aktualnym punkcie, odpowiadające wartościom parametru kroku (learning rate) = {0.1, 0.01, 0.001}

Wyniki działania programu:

Prarametr kroku = 0.001

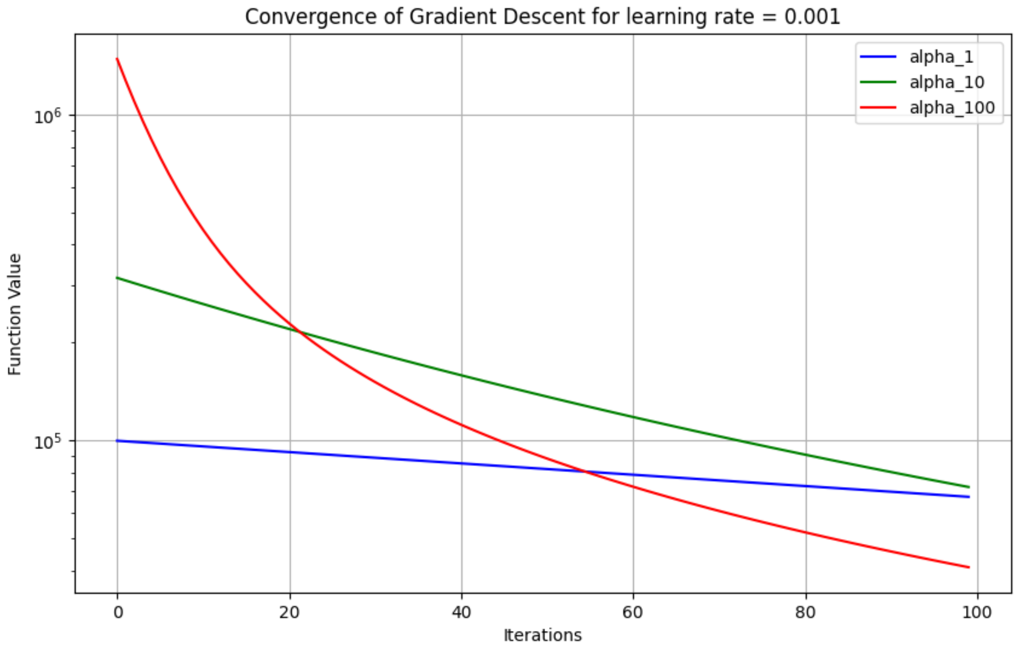

Table for learning rate = 0.001

	Alpha	Reason for stop	Time (s)
0	1	Reached max iterations	0.102100
1	10	Reached max iterations	0.103084
2	100	Reached max iterations	0.092992

Linie powoli zbliżają się do minimum lokalnego (które istnieje dla g(x) = 0), ale zatrzymują się przez przekroczenie maksymalnej liczby iteracji.

Dla tego wykresu im większy parametr α, tym wyższa wartość w punkcie początkowym, ale i szybsze wyszukiwanie minimum lokalnego.

Czasy działania funkcji są porównywalne.

Parametr kroku = 0.01

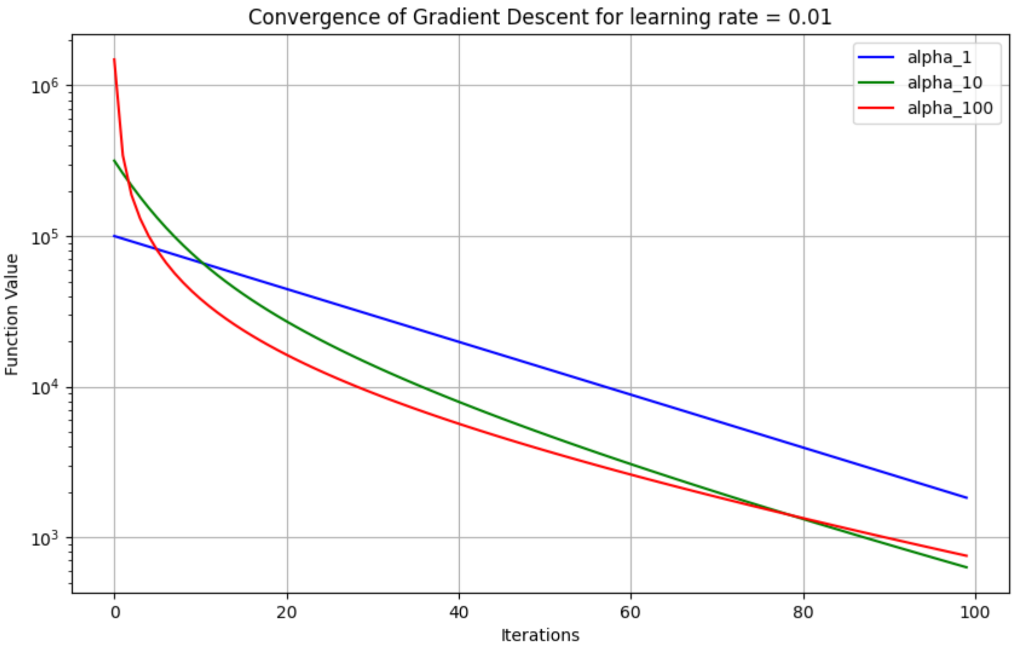

Table for learning rate = 0.01

	Alpha	Reason for stop	Time (s)
0	1	Reached max iterations	0.107579
1	10	Reached max iterations	0.110569
2	100	Reached max iterations	0.097525

Linie zbliżają się do minimum w znacznie szybszym tępie, lecz nadal nie dochodzą do celu.

Linia dla α =100 pomimo początkowo szybszego tępa znajdywania zostaje „prześcignięta" przez linię dla α =10. Oznacza to, że w dłuższej perspetywie mniejsza α może oznaczać szybsze tępo poszukiwań minimum.

Czasy działania podobnie do poprzedniego wykresu są porównywalne.

Parametr kroku = 0.1

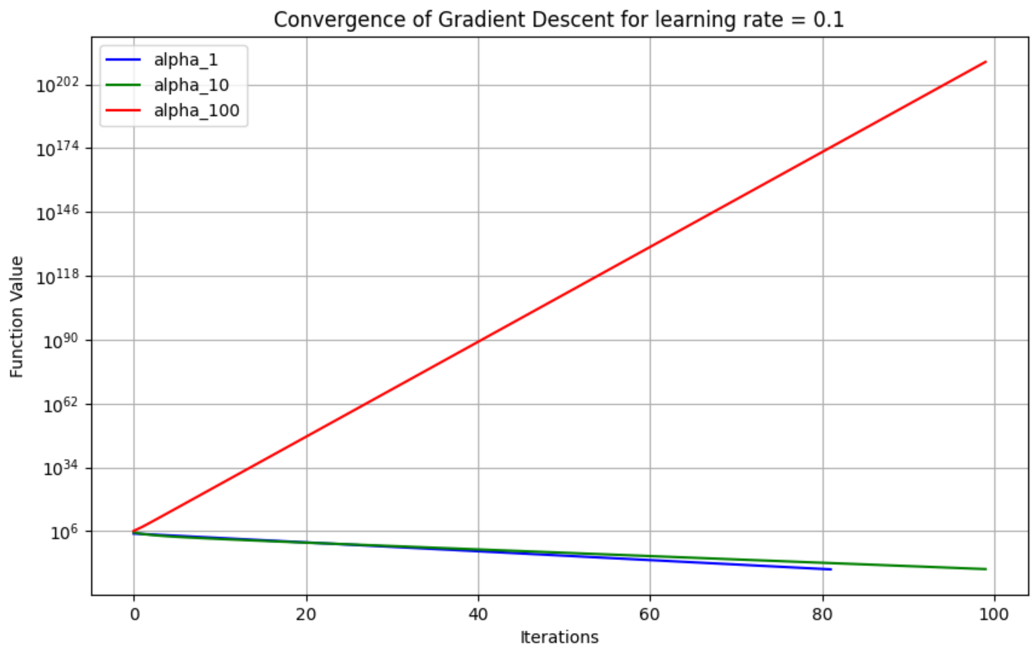

Table for learning rate = 0.1

	Alpha	Reason for stop	Time (s)
0	1	Small difference between points. Close to loca...	0.079526
1	10	Reached max iterations	0.094517
2	100	Reached max iterations	0.093002

2 z 3 linii zbliżają się do minimum w jeszcze większym tępie.

Linia dla α =100 zamiast zbliżać się do minimum, oddala się w znczanie szybszym tępie niż reszta się przybliża.

Linia dla α =1 osiąga minimum z podaną precyzją. Warunek stopu odnoszący się do różnicy kolejnych punktów zadziałał szybciej od warunku sprawdzającego gradient.

Czas nie jest prównywalny jedynie w linii gdzie α =1, gdyż poszukiwanie minimum zakończyło się przed osiągnięciem maksymalnej liczby iteracji.

Wywołuję pogram jeszcze raz. Tym razem ustawiam maksymalną liczbę iteracji na 1000 i wykluczam parametr kroku równy 0.1, gdyż w tej liczbie iteracji osiąga zbyt duże liczby.

Parametr kroku = 0.001

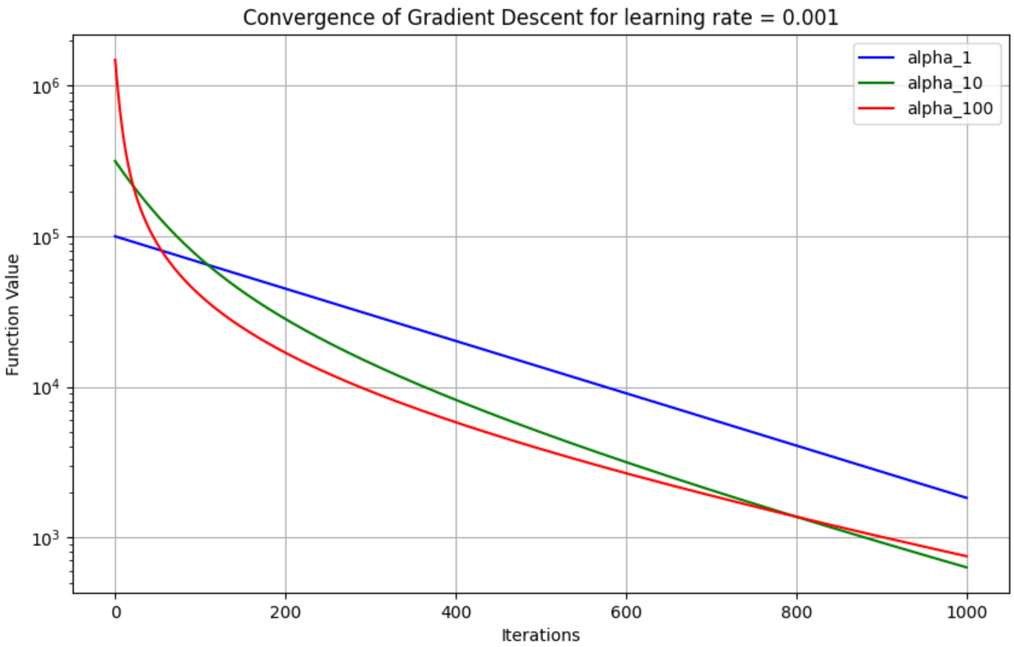

Table for learning rate = 0.001

	Alpha	Reason for stop	Time (s)
0	1	Reached max iterations	0.793159
1	10	Reached max iterations	0.917290
2	100	Reached max iterations	0.906125

Nadal żadna z linii nie dochodzi do minimum.

Czasy są większe 10 razy od czasów dla maksymalnej liczby iteracji równej 100.

Parametr kroku = 0.01

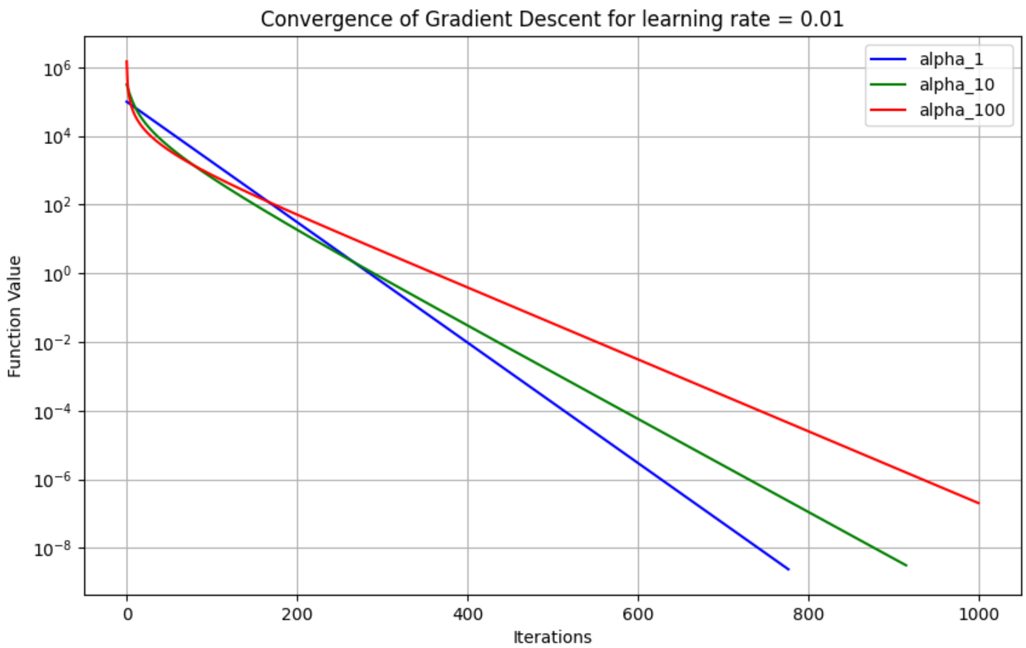

Table for learning rate = 0.01

	Alpha	Reason for stop	Time (s)
0	1	Small difference between points. Close to loca...	0.626983
1	10	Small difference between points. Close to loca...	0.732398
2	100	Reached max iterations	0.910798

Dwie z trzech linii osiąga minimum.

Znowu zadziałał najpierw warunek stopu odwołujący się do różnicy kolejnych punktów.

Na tym wykresie wyraźnie widać, że dla mniejszych ilości iteracji większa α w funkcji będzie skutkowała szybszym zbliżaniem się do minimum. Tendencja ta zmienia się całkowicie przy większych ilościach iteracji.

Wnioski:

Do pewnego momentu im większa wartość parametru kroku, tym szybciej funkcja zbliża się do minimum.

Powyżej pewnej wartości parametru kroku metoda gradientu prostego przestaje działać prawidłowo.

Optymalną wartość parametru kroku znajduje się metodą prób i błędów.

Większa wartość parametru α w funkcji g(x) powoduje przyśpieszenie tempa zbliżania w początkowych iteracjach oraz zwolnienie w dalszej perspektywie w porównaniu do mniejszej wartości parametru α.

Warunek stopu odwołujący się do różnicy kolejnych punktów zadziała szybciej niż warunek odwołujący się do gradientu.

Czas działania funkcji gradientu prostego jest linowo zależny od ilości iteracji.