WSI - ćwiczenie 1.

Zagadnienie przeszukiwania i podstawowe podejścia do niego

Dokumentacja – Mikołaj Olejnik

Treść zadania

Celem ćwiczenia jest implementacja algorytmu gradientu prostego oraz zastosowanie go do znalezienia minimum funkcji f i g. Ponadto należy zbadać wpływ rozmiaru kroku oraz różne punkty początkowe.

Funkcje (Uwaga: funkcja f jest funkcją dwuwymiarową. x_i oznacza i-ty element wektora x):

$$f(x) = x_1^2 + x_2^2$$

$$g(x) = x^2 - 10\cos(2\pi x) + 10$$

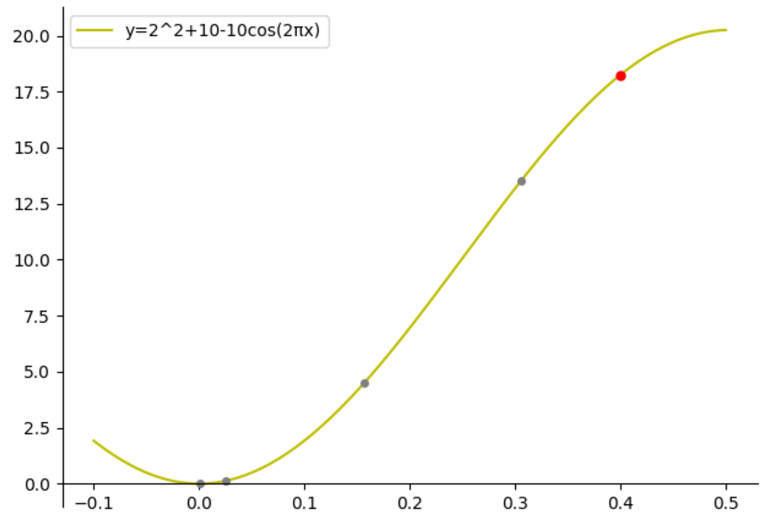
Gradienty funkcji:

$$\nabla f(x) = \begin{bmatrix} 2x_1 \\ 2x_2 \end{bmatrix}$$

$$\nabla g(x) = 2x + 20\pi \sin(2\pi x)$$

Wyniki eksperymentów

Każdą z funkcji przetestowałem 4 razy dla różnych parametrów - wielkości kroku oraz początkowego punktu.

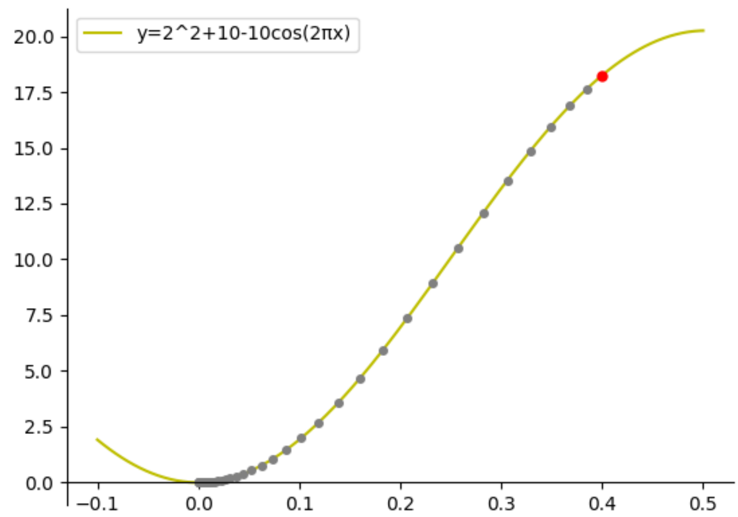


Wykres 1.

starting_x = 0.4

learn_rate = 0.0025

Liczba iteracji - 4

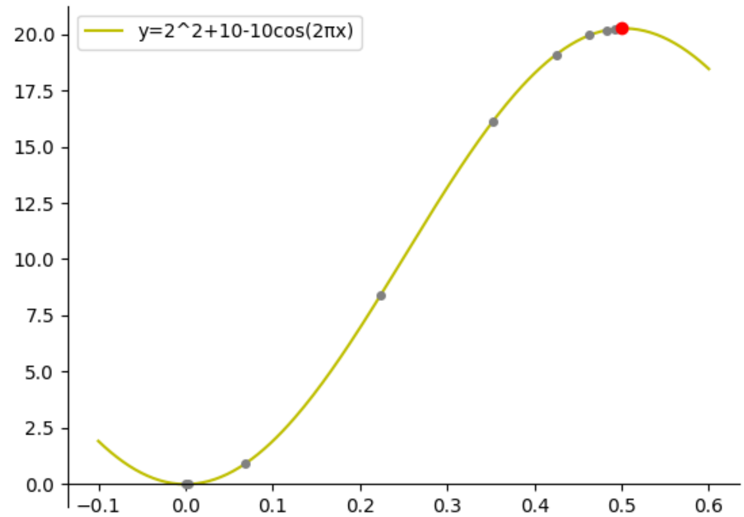


Wykres 2.

starting_x = 0.4

learn_rate = 0.0004

Liczba iteracji - 44

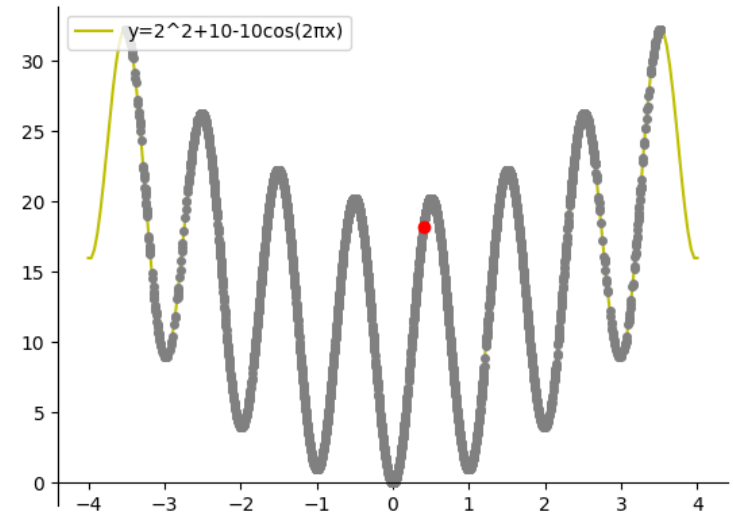


Wykres 3.

starting_x = 0.5

learn_rate = 0.0025

Liczba iteracji – 10

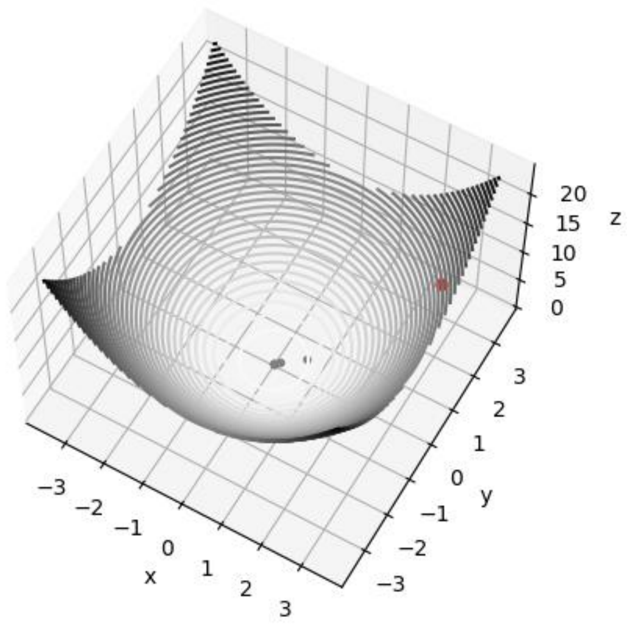


Wykres 4.

starting_x = 0.4

learn_rate = 0.013

Liczba iteracji – 5000 (przyjęta maksymalna liczba iteracji)

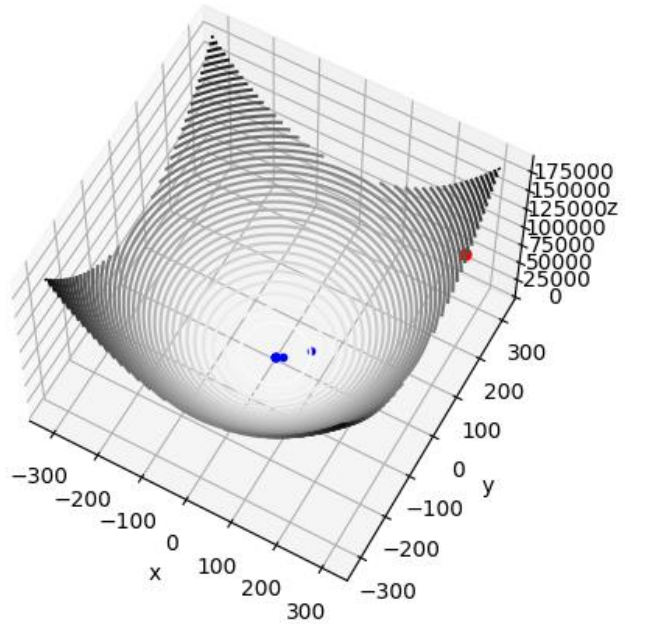


Wykres 5.

starting_x1 = 3, starting_x2 = 2

learn_rate = 0.4

Liczba iteracji – 7

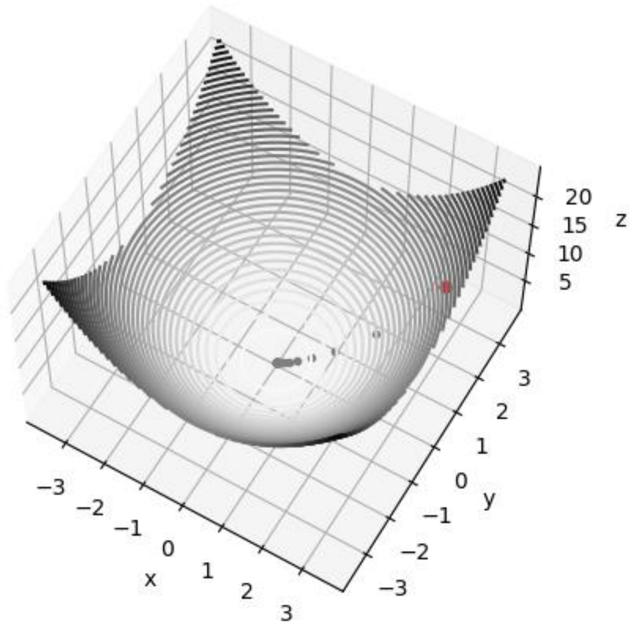


Wykres 6.

starting_x1 = 300, starting_x2 = 200

learn_rate = 0.4

Liczba iteracji – 10

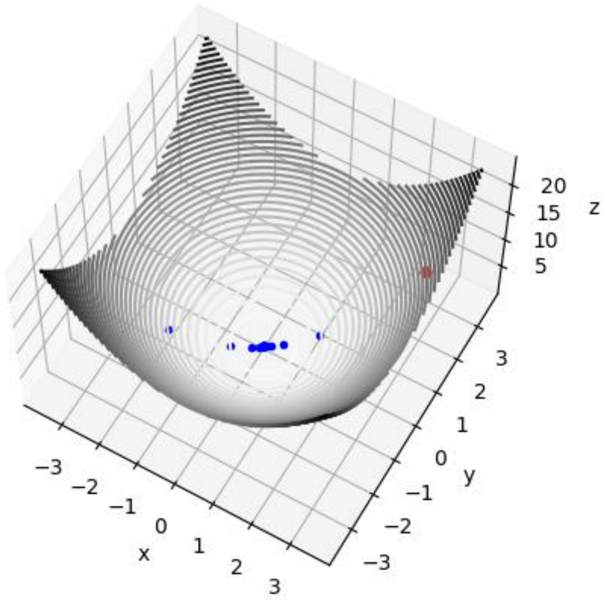


Wykres 7.

starting_x1 = 3, starting_x2 = 2

learn_rate = 0.2

Liczba iteracji – 21

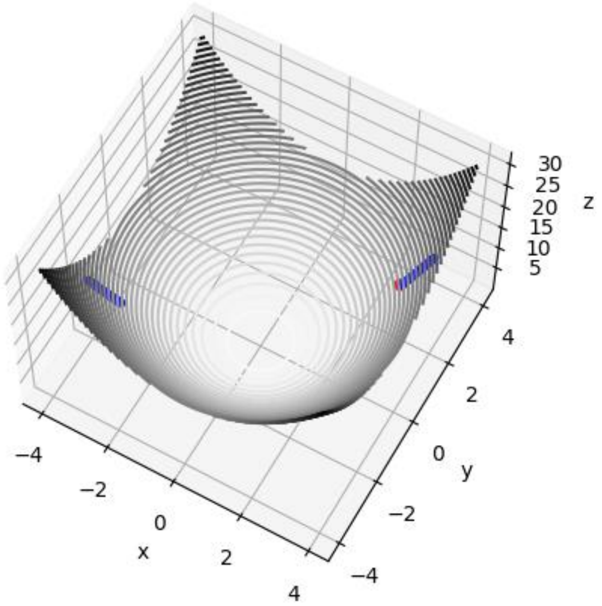


Wykres 8.

starting_x1 = 3, starting_x2 = 2

learn_rate = 0.8

Liczba iteracji – 21



Wykres 9.

starting_x1 = 3, starting_x2 = 2

learn_rate = 1.001

Liczba iteracji – 100 (przyjęta maksymalna liczba iteracji)

Obserwacje i wnioski

Szybkość znalezienia minimum lokalnego zależy od wielkości kroku, więc jakoś zaimplementowanego przeze mnie algorytmu zależy od parametru learn_rate.

Dobrze dobrana wartość learn_rate powoduje, że szybko zbliżamy się do minimum, z każdą iteracją znacząco się do niego zbliżając.

Jeśli wybrana wartość wielkości kroku jest za duża, to możliwe jest, że nigdy nie znajdziemy lokalnego minimum. Nawet jeśli mamy trochę szczęścia I algorytm znajdzie jakieś minimum, to może to zająć bardzo dużo iteracji. Sytuację to widać bardzo dobrze na wykresie 4 i 9. W obydwu przypadkach algorytm zatrzymał się po wcześniej zdefiniowanej liczbie kroków – inaczej mógłby działać w nieskończoność nie znajdując żadnego minimum. Ale nawet z za dużą wartością learn_rate ciągle możliwe jest, że znajdziemy minimum. Widoczne jest to na wykresie 7 i 8.

Jeśli wartość kroku jest za mała, to na pewno znajdziemy minimum, ale będzie to bardzo powolne i zajmie o wiele za dużo iteracji algorytmu niż jest potrzebne. Widać to na wykresie 2. Algorytm potrzebował aż 44 iteracji, aby znaleźć minimum. Dla ponda 6 razy większej wartości learn_rate algorytm znajduje minimum tylko w 4 iteracje.

To jak szybko znajdziemy minimum i które zależy też od tego jaki punkt wybierzemy jako startowy. Eksperymenty, które przedstawiają wykresy 1 i 3 mają taką samą wartość learn_rate, ale różny punkt startowy starting_x. Podejście z starting_x = 0.5 potrzebuje o 250% iteracji więcej niż to z starting_x = 0.4. W przypadku funkcji f, punkt startowy również ma znaczenie, ale dość niewielkie. Widać to porównując wynik algorytmu na wykresie 5 i 6.

Ważnym aspektem algorytmu gradientu prostego jest to, że znajduje on minimum lokalne a nie globalne. Jeśli nasz algorytm znajdzie punkt, który znajduje się w obszarze jakiegoś minimum lokalnego, to już z niego nie wyjdzie (przez co nie znajdzie innego minimum – globalnego), chyba że wielkość kroku będzie wystarczająco duża (chociaż raczej chcemy dobierać taką wielkość kroku, żeby algorytm „szedł w dół”).

Minimalizując funkcję zazwyczaj chcemy znaleźć minimum globalne. Niestety ten algorytm nie jest w stanie rozróżnić minimum globalnego od lokalnego.