Politechnika Warszawska

Wydział Elektroniki i Technik Informacyjnych

Wstęp do Sztucznej Inteligencji

ćwiczenie 1 - Zagadnienie przeszukiwania i podstawowe podejścia do niego

Autor:

Jarosław Jaworski 342189

Data oddania ćwiczenia:

25.03.2025

SPIS TREŚCI

1 Cel ćwiczenia

Celem ćwiczenia było zaimplementowanie algorytmu najszybszego wzrostu dla znajdowania ekstremum (minimum) funkcji. W ramach projektu należało prowadzić kontrole źródłową, zapewnić elastyczność głównej funkcjonalności oraz przeprowadzić porównania działania programu dla różnych punktów startowych i zbadać wpływ ilości argumentów na efektywność.

2 Pogląd zaimplementowanych funkcjonalności

Nazwa pliku	Krótki opis zmian
main.py	plik główny wywoływany w trybach: 1. Dwuargumentowy - wywołanie programu dla funkcji jednoargumentowej (2D) 2. Trójargumentowy - wywołanie programu dla funkcji dwuargumentowej (3D) 3. Pozostałe przypadki - informacja zwrotna o niepoprawnym wywołaniu programu
paint_figure.py	wizualizacja funkcji dwu lub trójwymiarowej, wywołanie algorytmu gradient_descent
gradient_solve.py	główna funkcjonalność, egzekucja algorytmu dla dowolnej liczby argumentów (dowolnej liczby wielkości wektora zmiennych decyzyjnych), obliczenie gradientu przez aproksymację numeryczną
problem_functions.py	zawiera definicje funkcji celu
scipy_optimize.py	program referencyjny z biblioteką scipy do porównania z ekstremum znalezionego algorytmem gradient_descent

3 Porównanie efektywności dla różnych punktów startowych i rozmiaru kroku

3.1 Metodologia

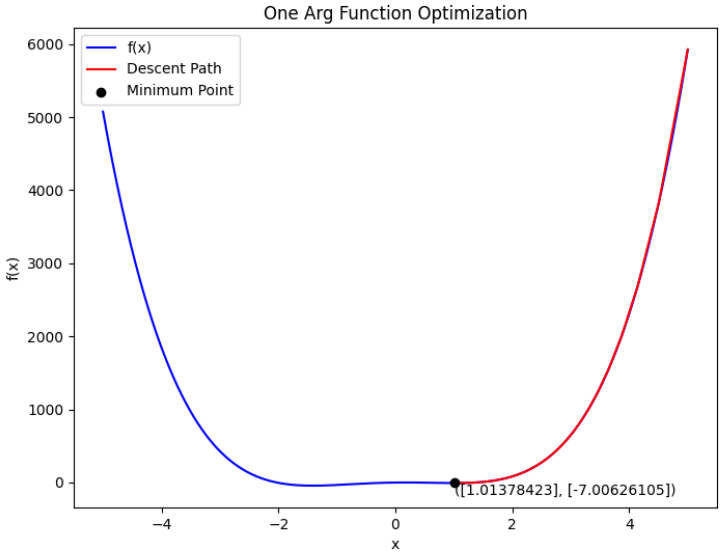
W celu sprawdzenia wpływu wielkości kroku na efektywność działania funkcji wybierano punkty startowe zgodnie z następującym schematem:

1. Punkt startowy niedaleko minimum funkcji,
2. Punkt startowy w otoczeniu minimum funkcji,
3. Punkt startowy średnio oddalony od minimum funkcji,
4. Punkt startowy znacząco oddalony od minimum funkcji,

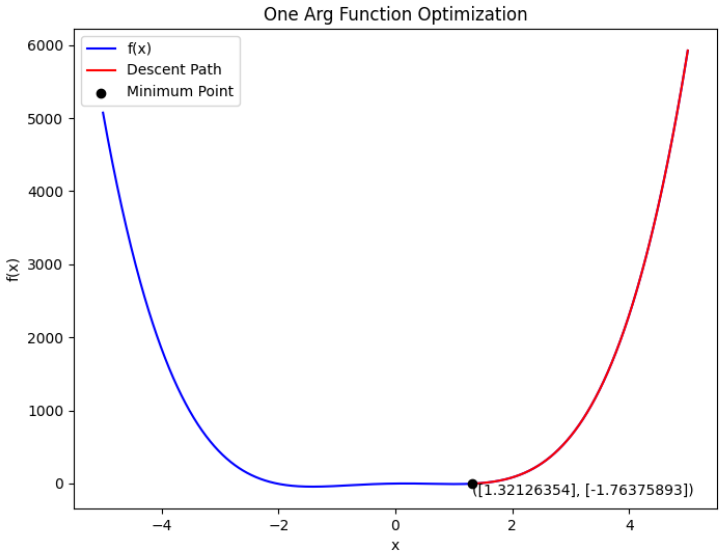
i badano dla nich przejście gradientu przy różnym kroku. Wyniki pomiarów zamieszczono w kolejnych podpunktach.

3.2 Funkcja jednoargumentowa f(x)

Dla funkcji jednoargumentowej zaimplementowano 4 różne rozmiary kroku (dokładnie współczynnik definiujący jego wielkość). Testy wykonywano poprzez zadanie punktu startowego zgodnie ze schematem opisanym w punkcie *3.1 Metodologia* i przypisywanie kolejnych współczynników wielkości kroku. Następnie badano czy minimum zostawało osiągane poprzez porównanie z wartością wyznaczaną przez gotowe funkcje z biblioteki *scipy* oraz organoleptyczne oceny wykresów. Przykładowe testy pozytywny i negatywny przedstawiono na Rysunkach 3.1.1-3.1.2.



Rysunek 3.2.1 Przykład sukcesywnego znalezienia minimum dla punktu startowego rodzaju 2 i kroku 1e-4



Rysunek 3.2.2 Przykład nieudanego szukania minimum dla punktu startowego rodzaju 2 i kroku 1e-6

Tabela 3.1 zawiera podsumowanie wyników pomiarów.

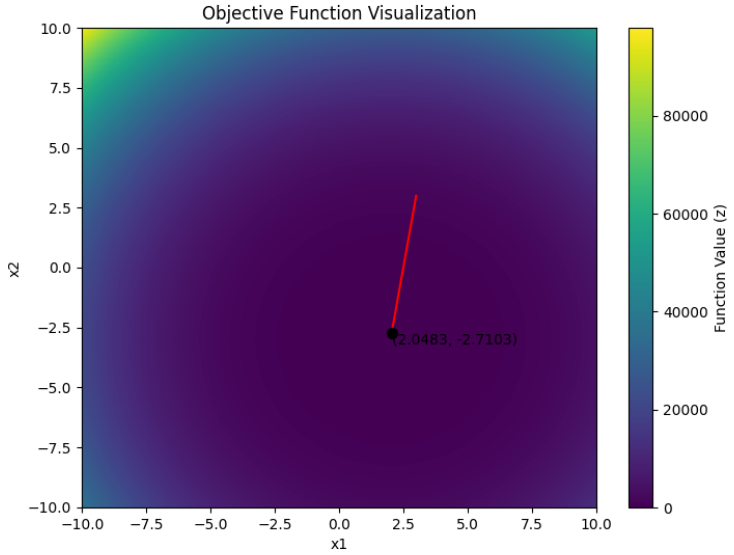
Tabela 3.1 Wyniki pomiarów dla funkcji jednoargumentowej

Wielkość kroku β	**Rodzaje punktów dla których osiągnięto minimum**	**Czas dla danego rodzaju punktu [s]**
1e-3	1,2	1- 0.002, 2- 0.001
1e-4	1,2,3	1- 0.031, 2- 0.01, 3- 0.0290
1e-5	1,2,3	1- 0.0968, 2- 0.0, 3- 0.0971
1e-6	2	2- 0.0

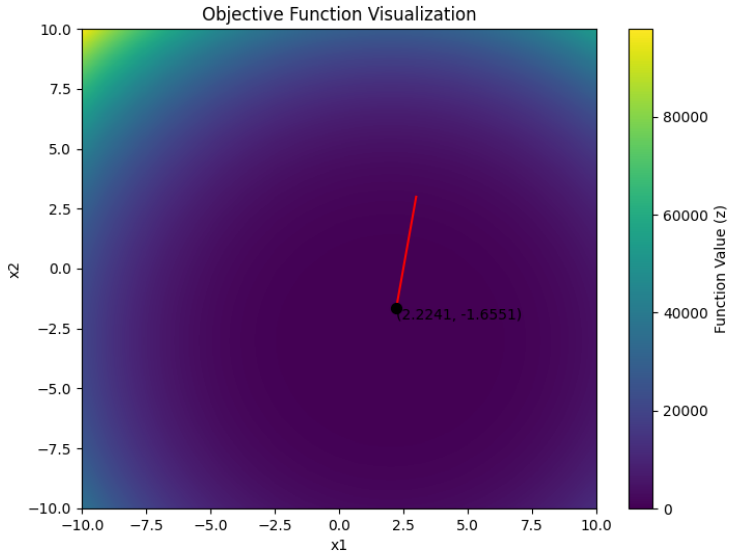
3.3 Funkcja dwuargumentowa g(x1,x2)

Dla funkcji dwuargumentowej postępowano analogicznie do funkcji jednoargumentowej. Przykładowe testy przedstawiono na Rysunkach 3.3.1 i 3.3.2.

Tabela 3.2 przedstawia wyniki pomiarów.



Rysunek 3.3.1 Przykład sukcesywnego znalezienia minimum dla punktu startowego rodzaju 2 i kroku 1e-4



Rysunek 3.3.2 Przykład nieudanego szukania minimum dla punktu startowego rodzaju 2 i kroku 1e-6

Tabela 3.2 Wyniki pomiarów dla funkcji dwuargumentowej

Wielkość kroku β	**Rodzaje punktów dla których osiągnięto minimum**	**Czas dla danego rodzaju punktu [s]**
1e-3	1,2,3	1- 0.1273, 2- 0.0, 3- 0.1034
1e-4	1,2,3	1- 0.3318, 2- 0.0, 3- 0.2614
1e-5	1,2,3,4	1- 0.7819, 2- 0.0, 3- 0.5321, 4- 0.516
1e-6	1,2,3,4	1- 1.2944, 2- 0.0, 3- 1.0873, 4- 1.0579

4 Wnioski

Przy stosowaniu algorytmu najszybszego wzrostu (gradient descent) należy zważać na równowagę między uzyskaniem najlepszej możliwej dokładności wyników a czasem wykonywania, a tym samym ilością iteracji. Na podstawie wyników można zauważyć, że z wyższą dokładnością kroku znacznie rośnie czas wykonywania programu, jednak znajdowane rozwiązania są bliżej rzeczywistej wartości minimum funkcji. Niestety, przy stosowaniu ograniczeń liczby operacji (iteracji) algorytm może nie dojść do poprawnego rozwiązania z małym krokiem lub/i zaczynając ze znacznie oddalonego punktu startowego.

Dla wykorzystanej metody należy uważnie interpretować otrzymywane wyniki. Algorytm nie bierze pod uwagę znalezienia wielu ekstremów przy pojedynczym wywołaniu, co wiąże się z koniecznością wielokrotnego wykonywania

testów dla zadanych różnych punktów startowych i różnych kroków. Mimo tego, możliwe jest nieznalezienie wszystkich ekstremów funkcji. W efekcie algorytm najszybszego wzrostu stanowi skuteczne i proste w implementacji narzędzie, jednak jego użyteczność zależy bezpośrednio od zadania.