Igor Kraszewski

310164

Wprowadzenie do sztucznej inteligencji – ćwiczenie 1, spotkanie 2 (za 4 punkty)

Zadania:

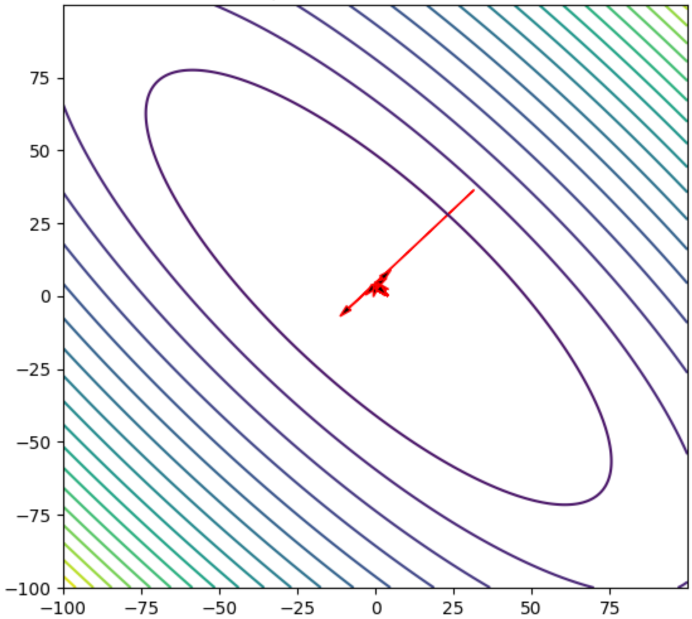
- Zaimplementować metodę najszybszego wzrostu/spadku (minimalizacja, spodziewam się stałego współczynnika kroku, jeśli jednak ktoś chce zrobić więcej i zastosować zmienny współczynnik to ma taką możliwość). Gradient wyliczamy numerycznie.
- Narysować zachowanie algorytmu (kolejne kroki algorytmu jako strzałki na tle poziomic funkcji celu). Uwaga: w praktycznych zadaniach optymalizacji nie da się narysować funkcji celu ponieważ zadania mają wiele wymiarów (np. 100), oraz koszt wyznaczenia oceny jednego punktu jest duży.
- Zastosować metodę do znalezienia optimum funkcji booth w 2 wymiarach, po czym do znalezienia optimum funkcji o numerach od 1 do 3 z CEC 2017 w 10 wymiarach (na wykresie narysować kroki w wybranych 2 wymiarach z 10). Ograniczenia kostkowe przestrzeni to -100, 100.

W sprawozdaniu należy zawrzeć wykresy uzyskane stworzonym oprogramowaniem (np. po 3 dla każdej funkcji, dla różnych punktów startowych). Należy podać wartość funkcji celu w punkcie uznanym za optimum.

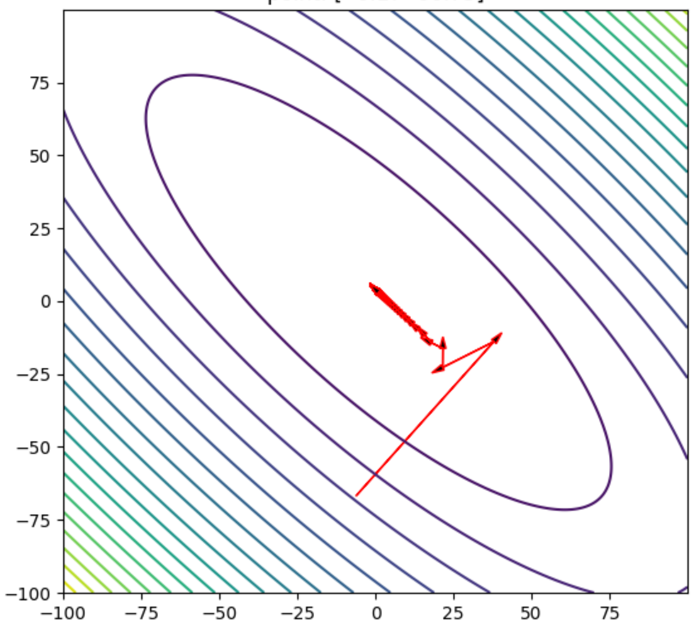
Pytania:

1. Jak wartość parametru beta wpływa na szybkość dojścia do optimum i zachowanie algorytmu? Jakiej bety użyto dla każdej z funkcji?
2. Zalety/wady algorytmu?
3. Wnioski

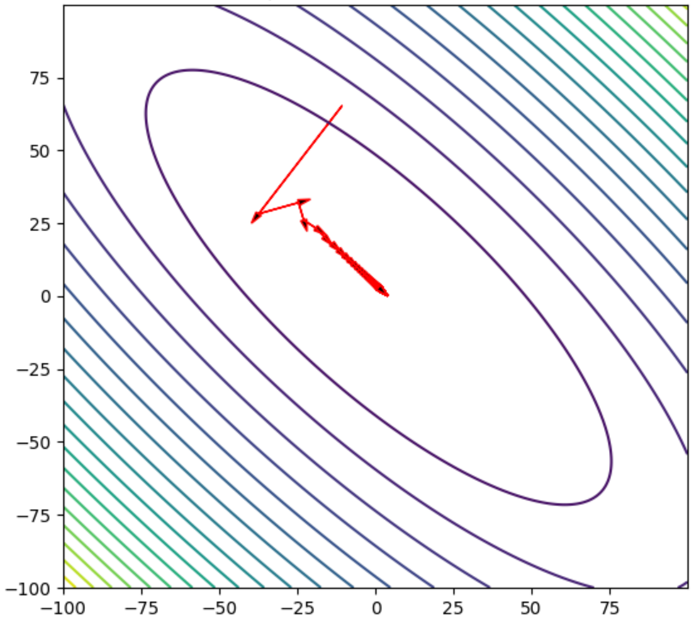
Function booth_function, Minimum = 0.0, Starting point: [31.51 36.39]



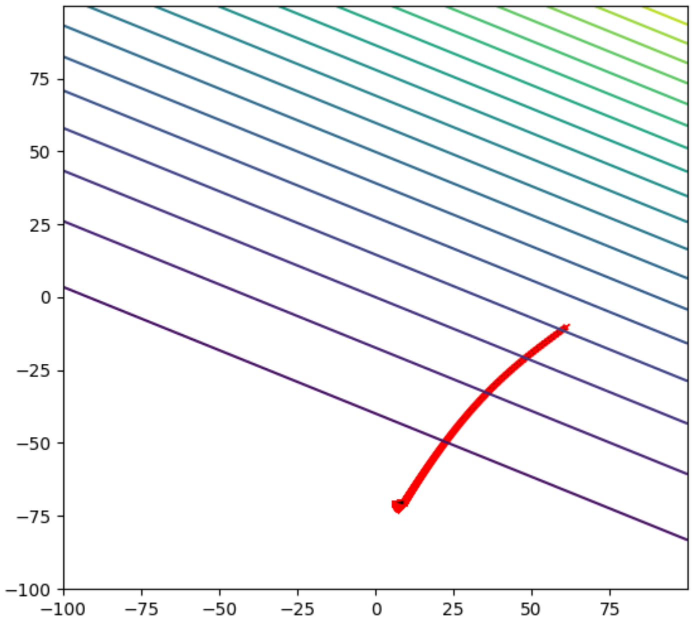
Function booth_function, Minimum = 0.0, Starting point: [-6.17 -66.73]



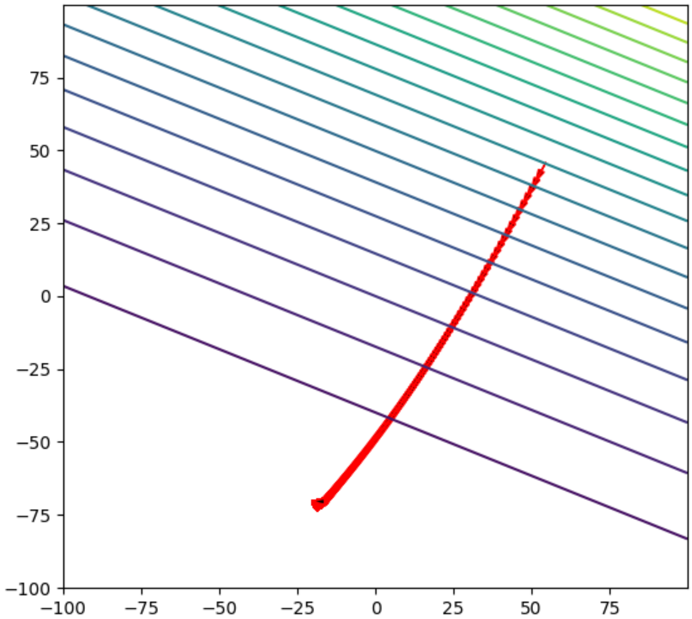
Function booth_function, Minimum = 0.0, Starting point: [-10.69 65.3]



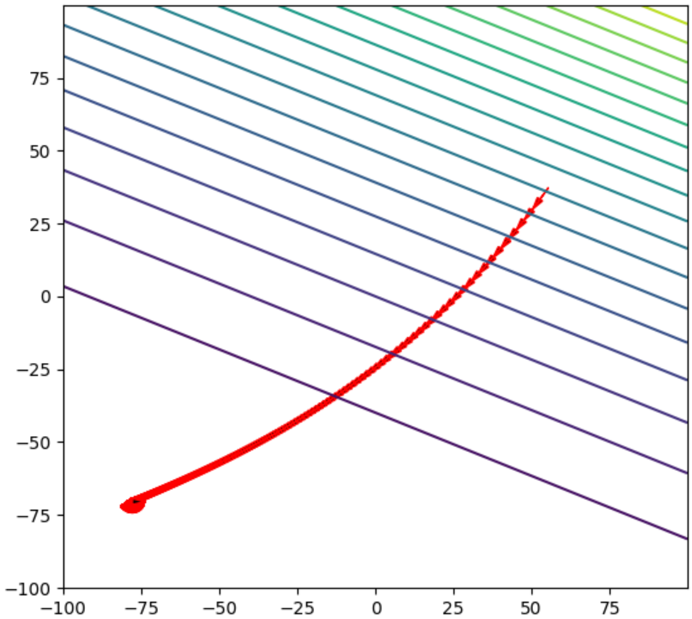
Function f1, Minimum = 11620.59, Starting point: [62.09 -9.42   7.   -60.71 -8.48 12.76 96.59 -24.57 -98.57 -24.26]



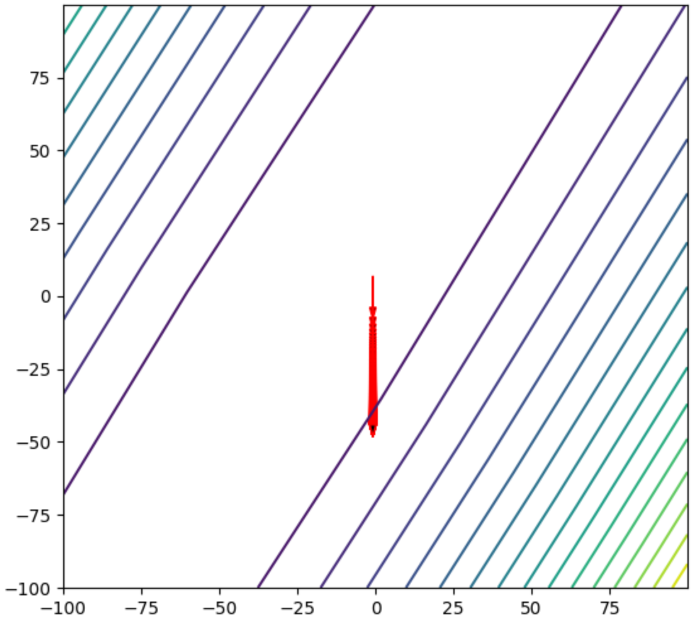
Function f1, Minimum = 4291.3, Starting point: [
54.54 45.81 88.55 -51.35   5.09 -45.85 63.
14.05 73.54   9.89]



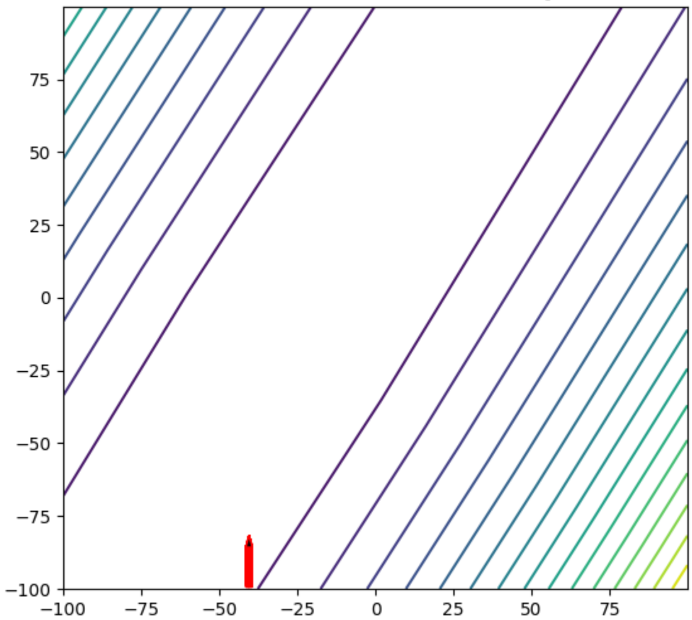
Function f1, Minimum = 1509.32, Starting point:
[55.34 37.22 -95.28   9.94 -45.9   -5.99 -62.64
44.59 -67.49 59.54]



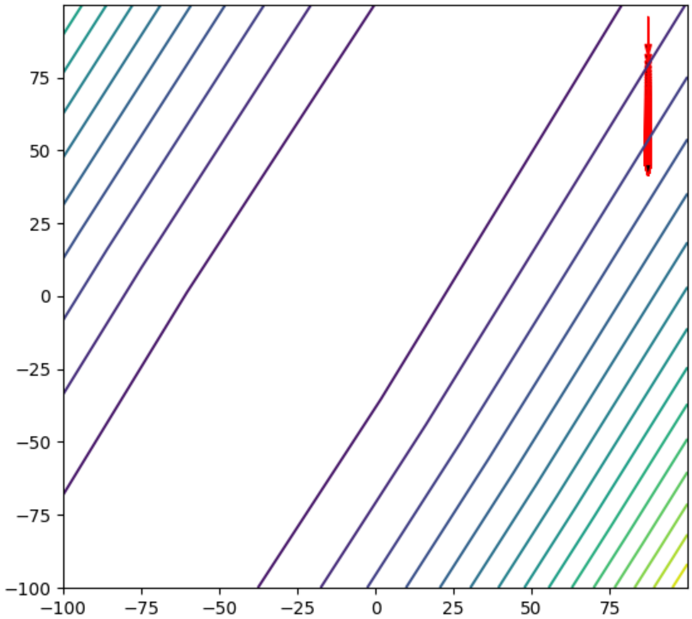
Function f2, Minimum = 1.2008479308500578e+17,
Starting point: [-0.84   6.86 81.11 -46.02
-61.19 -93.97 -45.39 -31.86 -80.87 -93.82]



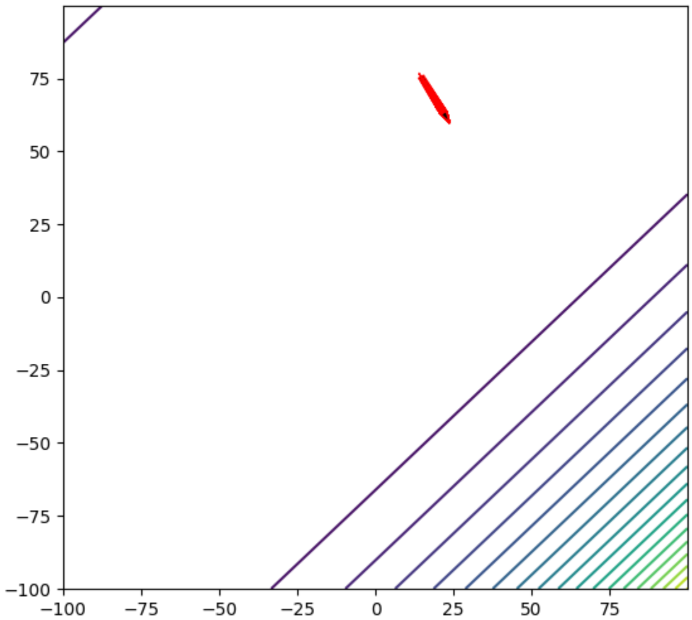
Function f2, Minimum = 2.1047539032673788e+16,
Starting point: [-40.46 -99.47 -9.08 -65.31 86.6
-54.1   -4.98 29.17 52.45 95.69]

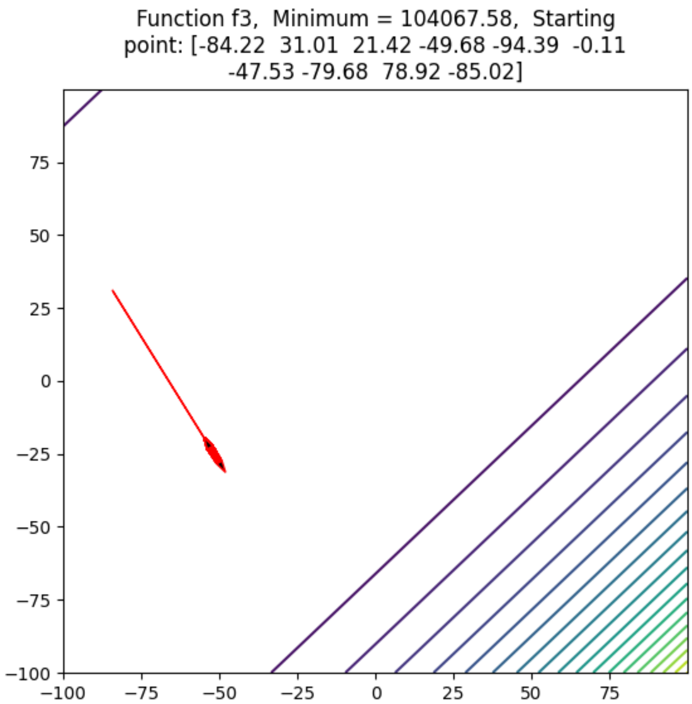


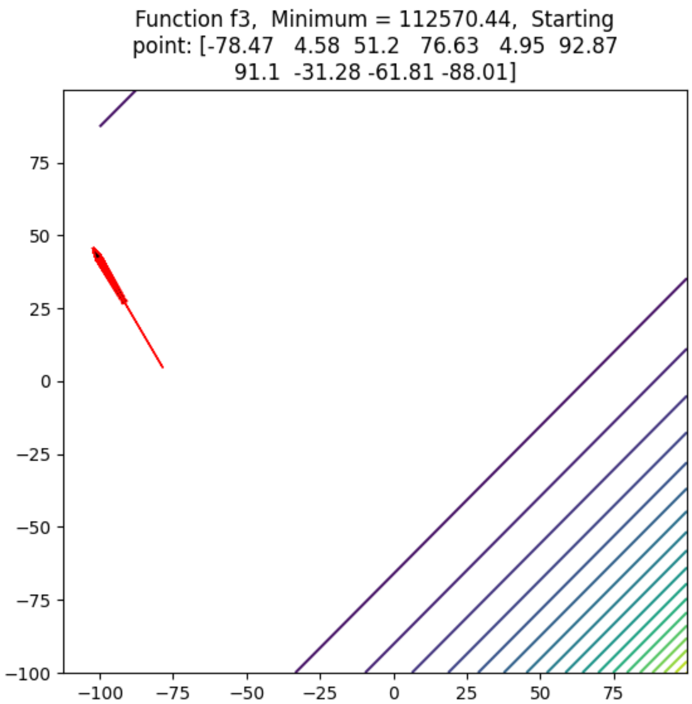
Function f2, Minimum = 2.33576317666988e+16,
Starting point: [87.33 95.86 -47.56 -4.89
25.34 -59.36 15.93 78.24 -49.15 56.96]



Function f3, Minimum = 169737.0, Starting point:
[13.94 76.7 -53.97 10.22 -23.13 32.14 -11.68
92.5   41.34 69.18]







1. Jak wartość parametru beta wpływa na szybkość dojścia do optimum i zachowanie algorytmu? Jakiej bety użyto dla każdej z funkcji?

   Przy zbyt dużej wartości parametru beta algorytm może oscylować wokół optimum, a dla zbyt małej wartości czas dojścia do optimum może się wydłużyć. Dodatkowo w przypadku funkcji z cec2017 zbyt duża beta powodowało wyjście poza zakres wartości, a więc błąd wykonania programu. Dla funkcji booth przyjęta beta to 0.07, dla f1 10^-9, f2 10^-18, f3 2*10^-9. Dla funkcji z cec2017 były to wartości graniczne, aby program nie wyrzucił błędu.

2. Zalety/wady algorytmu?

   Zalety: działa dla funkcji każdego wymiaru, prosty w zrozumieniu, osiąga co najmniej minimum lokalne w skończonym czasie przy odpowiednio dobranym kroku

   Wady: Nie gwarantuje znalezienia minimum globalnego – może zatrzymać się w minimum lokalnym, może być bardzo wolny

3. Wnioski

   Wykorzystana metoda bardzo dobrze radzi sobie ze znalezieniem minimum funkcji booth niezależnie od punktu początkowego, podobnie w przypadku funkcji f1 i f3 (raczej minimum lokalne). Jeżeli chodzi o funkcję f2, bardzo ciężko było dostroić algorytm, jeżeli chodzi o odpowiedni krok i ilość iteracji. Jak widać po wykresach funkcja zachowuje się dziwnie, a dodatkowo bardzo zależy od początkowego punktu, w którym wystartowaliśmy algorytm.