Wstęp do sztucznej inteligencji
Sprawozdanie 1

Jan Kwiatkowski

Marzec 2023

1 Wstęp

Tematem ćwiczenia jest zagadnienie przeszukiwania i związane z tym podejścia. Zadanie polega na minimalizacji funkcji celu będącą funkcją Bootha w postaci:

$$J = f(x, y) = (x+2*y-7)^2+(2*x+y-5)^2, -5 <= x <= 5, -5 <= y <= 5 \quad (1)$$

W zadaniu optymalizacji powyższa funkcja jest także nazywana wskaźnikiem jakości, funkcją straty lub funkcją kosztu. Jest ona określona w przestrzeni 'D', i to z tej danej przestrzeni szukany jest punkt x ∈ X, y ∈ Y minimalizujący funkcję J.

W celu poszukiwania ekstremum zaimplementowano poniższe algorytmy numeryczne:

- metodę najszybszego spadku gradientu,
- metodę newtona.

Dla obu wyżej wymienionych metod na samym początku działania algorytmu wybierany jest punkt startowy (w ćwiczeniu jest on losowany) o współrzędnych $x_0 \in X, y_0 \in Y$. Poza tym ustalana jest maksymalna liczba iteracji, współczynnik proporcjonalności β oraz tolerancja ϵ (wraz z maksymalną liczbą iteracji wykorzystywana przy kryterium stopu, tzn. zatrzymaniu algorytmu i zwróceniu najlepszego osiągniętego rozwiązania).

W ramach badań testowano różne kryteria stopu i przerywano działanie algorytmu jeśli:

- różnica między wartościami funkcji kosztu w dwóch kolejnych iteracjach była mniejsza niż zadana tolerancja,
- liczba iteracji była równa maksymalnej dozwolonej liczbie iteracji,
- norma gradientu była mniejsza niż zadana tolerancja.

Ostatecznie zdecydowano się zatrzymywać algorytm gdy spełnione zostało któreś z dwóch ostatnich wymienionych powyżej kryteriów.

1.1 Użyte przy implementacji biblioteki

- numpy,
- autograd - wyznaczenie gradientu i hesjana,
- matplotlib.pyplot -wykresy
- random - losowanie punktów początkowych,
- time - pomiar czasu
- gc - operacje na garbage collector'ze.

2 Wykres badanej funkcji celu

Na wykresie z rysunku 1 przedstawiona została analizowana funkcja celu J. Z rysunku wynika, że badana funkcja w określonej na wstępie dziedzinie nie posiada punktów siodłowych dzięki czemu algorytm powinien bez problemu znaleźć minimum globalne.

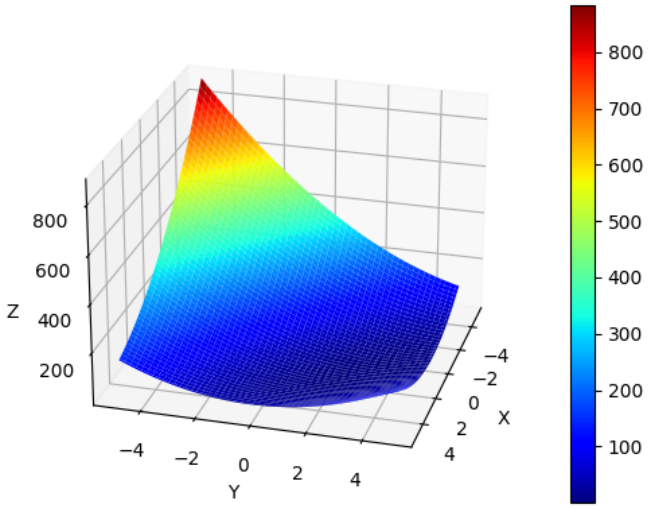

Figure 1: Wykres funkcji celu J.

3 Metoda najszybszego spadku gradientu

Każda iteracja tej metody polega na wykonaniu kroku w lokalnie najlepszym kierunku czyli w kierunku wyznaczonym przez ujemny gradient. Gradient funkcji w punkcie, $\nabla J(x)$ wskazuje w jakim kierunku w jej dziedzinie rośnie ona najszybciej, zaś w kierunku przeciwnym $-\nabla J(x)$, ona maleje.

Metody gradientowe są stosowalne w sytuacji, w której przestrzeń jest ciągła, funkcja celu jest ciągła oraz różniczkowalna, a także znany jest jej gradient.

3.1 Działanie algorytmu

Algorytm gradientowy stosuje wyżej opisane zależności do wyznaczania kolejnych wartości x_{k+1}, y_{k+1} przez przeszukanie funkcji J począwszy od punktu startowego x_0, y_0 w kierunku $-\nabla J(x_k)$, $-\nabla J(y_k)$ lub w kierunku w pewien sposób wynikającym z wyznaczonej w ten sposób wartości.

Algorithm 1 Algorytm metody najszybszego spadku gradientu

Input: x_0, y_0 - punkt początkowy, I - maksymalna liczba iteracji, β - współczynnik proporcjonalności, ϵ - kryterium stopu

```
1: function SteepestGradientDescent(x_0, y_0, β, I, ε)
2:     for n : od 1 do I: do
3:         Wyznacz gradient funkcji w punkcie (x,y)
4:         Wyznacz nowe współrzędne punktu (x,y)
5:         if Norma gradientu < ε then
6:             Zwróć: xmin,ymin, liczbaIteracji
7:         end if
8:     end for
9:     Zwróć: xmin,ymin, liczbaIteracji
10: end function
```

Metoda najszybszego spadku stanowi modyfikację metody gradientu prostego. W metodzie najszybszego spadku po określeniu kierunku poszukiwań wyznaczane jest minimum funkcji w tym kierunku, a nie przesunięcie ze stałym krokiem. Nowe współrzędne są wyznaczane zgodnie ze wzorem:

$$x_{t+1}, y_{t+1} = x_t, y_t - \epsilon * f'(x, y) \tag{2}$$

4 Metoda Newtona

Metoda Newtona jest metodą gradientową drugiego rzędu. Wymaga ona znajomości drugich pochodnych funkcji celu J. W metodzie tej oprócz gradientu wykorzystywany jest także hesjan. Fundamentalne znaczenie dla optymalizacji ma przybliżenie minimalizowanej funkcji J funkcją kwadratową. Z rozwinięcia funkcji J w szereg Taylora wokół punktu x otrzymujemy:

$$J(x + r) \approx J(x) + \nabla J(x)^T r + (1/2) r^T + \nabla^2 J(x) r \tag{3}$$

, gdzie x $\in \mathbb{R}$,$r \in \mathbb{R}$

Powyższa przybliżona równość oznacza aproksymację funkcji J gładko przylegającą do niej w punkcie x funkcją kwadratową. Metoda ta polega na znajdowaniu kolejnych punktów x_{t+1}, stanowiących minimum funkcji kwadratowej przylegającej do funkcji J w poprzednich punktach x_t.

Sama funkcja kwadratowa o ile jej hesjan $\nabla^2 J(x)$ jest dodatnio określony, ma minimum tam gdzie zeruje się jej gradient. Aby zminimalizować prawą stronę

równania 3 szukamy takiego r, aby gradient tego równania względem r był równy 0. Zatem

$$\nabla J(x) + \nabla^2 J(x) r = 0 \tag{4}$$

skąd wynika:

$$r = -(\nabla^2 J(x))^{-1} \nabla J(x) \tag{5}$$

Prowadzi to do metody Newtona wyznaczającej kolejne punkty, które mają stanowić coraz lepsze przybliżenie minimum funkcji J. Punkty te określone są w następujący sposób:

$$x_{t+1} = x_t - (\nabla^2 J(x))^{-1} \nabla J(x) \tag{6}$$

$$y_{t+1} = y_t - (\nabla^2 J(y))^{-1} \nabla J(y) \tag{7}$$

, gdzie t = $1, 2, ...$

4.1 Działanie algorytmu

W tym przypadku kierunek poprawy jest wyznaczany nie tylko w kierunku spadku gradientu ale również w kierunku spadku hesjana. Hesjan $\nabla^2 J(x)$ zawiera przydatne informacje o krzywiźnie funkcji. Dzięki uwzględnieniu drugiej pochodnej styczna do funkcji nie jest liniowa tylko kwadratowa (lepsza aproksymacja) a wybrany kierunek zawsze jest kierunkiem w stronę optimum. Tą funkcję złożoną z gradientu oraz hesjana minimalizujemy.

Algorithm 2 Algorytm metody newtona

Input: x_0, y_0 - punkt początkowy, I - maksymalna liczba iteracji, β - współczynnik proporcjonalności, ϵ - kryterium stopu

```
1:  function NewtonsMethod(x_0, y_0, β, I, ε)
2:      for n : od 1 do I: do
3:          Wyznacz gradient funkcji w punkcie (x,y)
4:          Wyznacz hesjan funkcji w punkcie (x,y)
5:          Wyznacz odwrotność hesjanu
6:          Wyznacz nowe współrzędne punktu (x,y)
7:          if Norma gradientu < ε then
8:              Zwróć: xmin,ymin, liczbaIteracji
9:          end if
10:     end for
11:     Zwróć: xmin,ymin, liczbaIteracji
12: end function
```

5 Badania i eksperymenty

W przypadku tych algorytmów nie możemy zagwarantować, że znalezione zostanie optimum globalne, tylko lokalne. Mając wiedze o całej dziedzinie (rys 1) można stwierdzić, że w tym przypadku znalezione optima są optimami globalnymi. Działania algorytmów przetestowano dla różnych wartości początkowych w każdym przypadku wywołując algorytmy 25 razy z losowymi punktami początkowymi. Uzyskane wyniki przedstawiono na stosownych wykresach oraz w poniższej tabeli:

Xmin	Ymin	f(Xmin,Ymin)
1	3	0

Podsumowanie wyników: Znalezione optimum funkcji celu J.

W badanym przypadku hesjan jest macierzą złożoną ze stałych wartości i nie zależy od zmiennych x oraz y, w związku z czym program wykonuje sie szybciej w porównaniu do przypadków w którym wartość hesjana zależałaby od x,y. Jedną z najbardziej kosztownych operacji metody newtona jest właśnie wyznaczanie odwrotności hesjana. W związku z tym faktem dla badanej funkcji celu J działanie metody newtona i metody największego spadku gradientu niewiele się od siebie różnią ponieważ hesjan jest traktowany jako stały współczynnik. Stąd wynikać mogą niewielkie różnice w czasie działania obu algorytmów. Jedyną znaczącą różnicą jest odwracanie hesjana w każdej iteracji mimo, że w tym przypadku te obliczenia również można by wyciągnąć przed pętlę for.

Wpływa to także na jednakową ilość iteracji obu metod dla każdego przykładu. Analizując poszególne iteracje algorytmów widać, że trajektoria punktu roboczego zależy od wartości współczynnika β. Im współczynnik jest mniejszy tym trajektoria jest gładsza, ale za to wydłuża się czas poszukiwania minimum. Z kolei zbyt duża wartość współczynnika β może doprowadzić do rozbieżności procesu i w konsekwencji zatrzymania algorytmu. Istotne jest zatem odpowiednie dobranie wartości współczynnika β.

5.1 Testy

W celu przeprowadzenia testów należy uruchomić plik tests.py. Dla najmniejszej wartości współczynnika β algorytm wykonuje się najdłużej. To w tym przypadku najbardziej widać różnice w czasie działania algorytmów z których wynika, że metoda newtona działa wolniej. Na rysunku 2 oraz 4 zauważyć można, że mimo ogólnie długiego czasu wykonywania się programu zdażają się przypadki w których znacząco szybciej kończy on swoje działanie. Oznacza to, że czas szukania optimum funkcji istotnie zależy od punktu początkowego x0, y0 i nawet dla małego współczynnika β możliwe jest szybkie znalezienie minimum.

Porównując wykres 2 z 3, 5 z 6 oraz 8 z 9 możemy stwierdzić, że czas działania algorytmów istotnie zależy od ilości wykonanych iteracji.

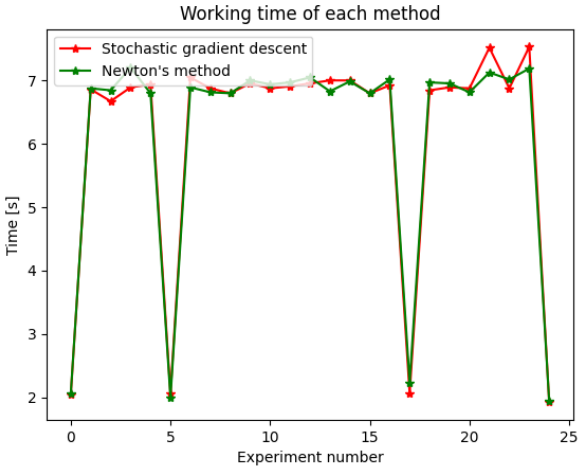

Figure 2: Wykres funkcji czasu działania dla kazdego przypadku. Dla: maksymalnej liczby iteracji = 100000; beta = 0,001.

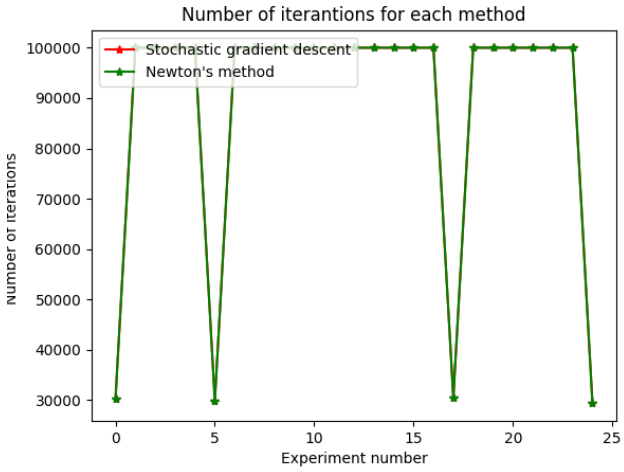

Figure 3: Liczba iteracji w każdym przypadku. Dla: maksymalnej liczby iteracji = 100000; beta = 0,001.

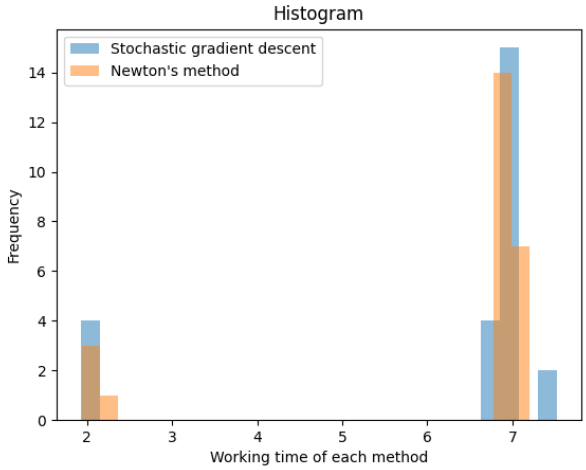

Figure 4: Histogram czasu działania programu dla różnych przypadków. Dla: maksymalnej liczby iteracji = 100000; beta = 0,001.

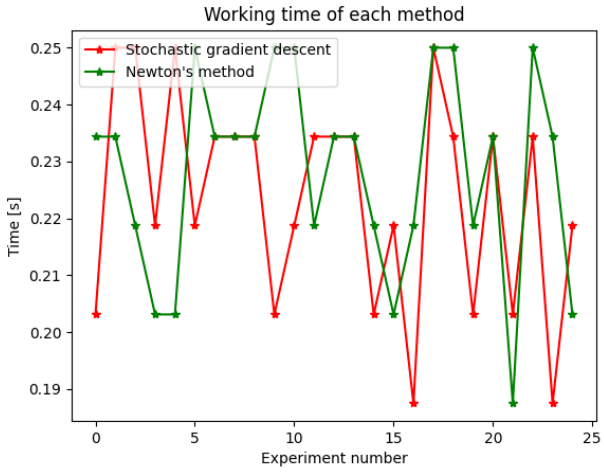

Figure 5: Wykres funkcji czasu działania dla kazdego przypadku. Dla: maksymalnej liczby iteracji = 100000; beta = 0,01.

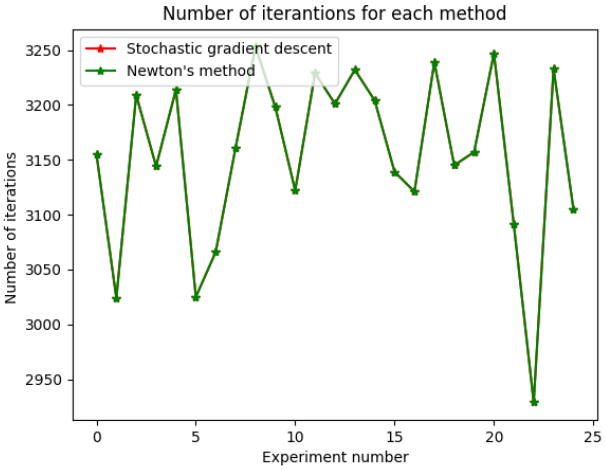

Figure 6: Liczba iteracji w każdym przypadku. Dla: maksymalnej liczby iteracji = 100000; beta = 0,01.

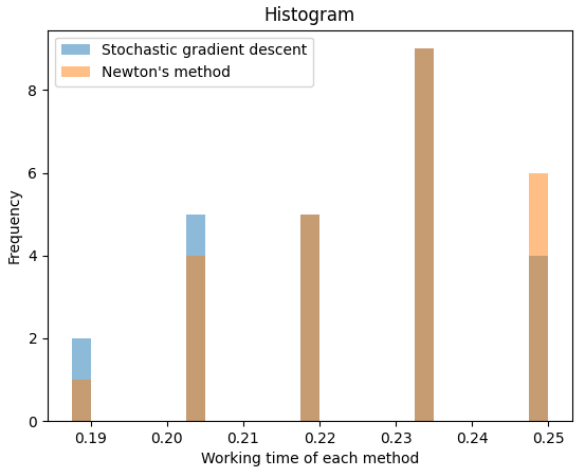

Figure 7: Histogram czasu działania programu dla różnych przypadków. Dla: maksymalnej liczby iteracji = 100000; beta = 0,01.

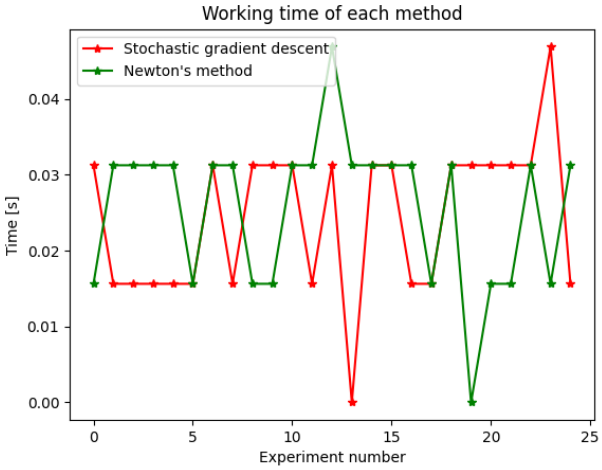

Figure 8: Wykres funkcji czasu działania dla kazdego przypadku. Dla: maksymalnej liczby iteracji = 100000; beta = 0,1.

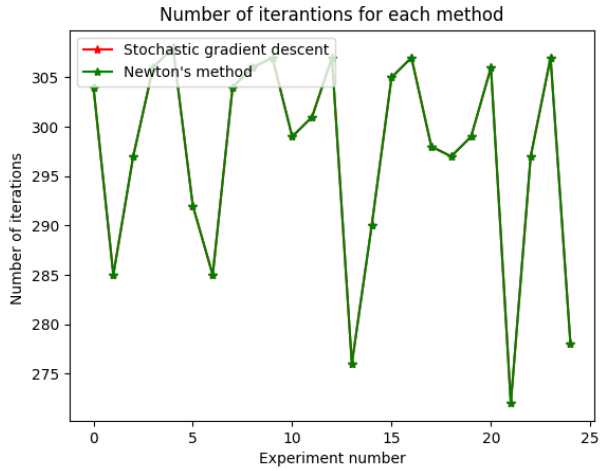

Figure 9: Liczba iteracji w każdym przypadku. Dla: maksymalnej liczby iteracji = 100000; beta = 0,1.

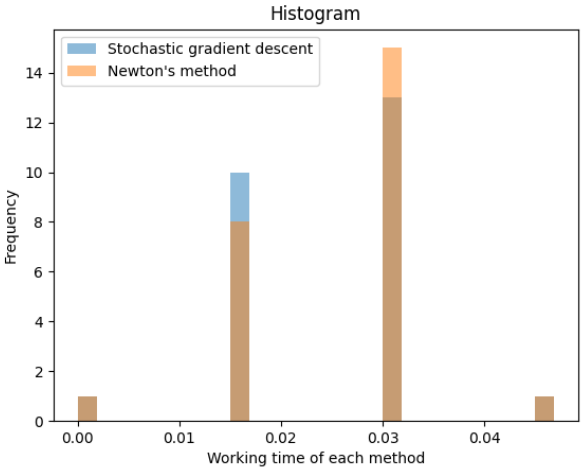

Figure 10: Histogram czasu działania programu dla różnych przypadków. Dla: maksymalnej liczby iteracji = 100000; beta = 0,1.