# Praca domowa 2
# Metoda gradientowa oraz minimalizacja wybranej funkcji

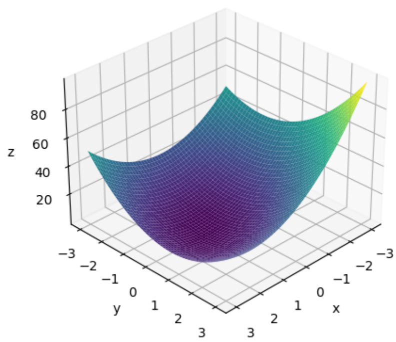
Damian Jankowski s188597

23 kwietnia 2023

## 1 Wstęp

Tematem pracy domowej jest zapoznanie się z minimalizacją funkcji metodą gradientową. W tym celu wybrałem funkcję $f(x, y) = 3(x-1)^2 - 2(x-1)y + 3y^2$. Minimum funkcji to $f(1, 0) = 0$.

Poszukiwanie minimum funkcji zacząłem od wybrania losowego punktu startowego $X_0$ z zakresu $-3 \leq x \leq 3$ oraz $-3 \leq y \leq 3$.




Rysunek 1: Wykres funkcji $f(x, y)$

## 2 Zadanie

Do wyznaczenia minimum funkcji metodą gradientową trzeba użyć wzoru:

$$X_{k+1} = X_k - \alpha_k \nabla f(X_k) \tag{1}$$

gdzie $\alpha_k$ jest stałą kroku, a $\nabla f(X_k)$ jest gradientem funkcji $f(x, y)$ w punkcie $X_k$.

Gradient funkcji $f(x, y)$ to inaczej wektor pochodnych cząstkowych:

$$\nabla f(x, y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) \tag{2}$$

W przypadku wybranej funkcji gradient ma postać:

$$\nabla f(x, y) = \begin{pmatrix} 6(x-1) - 2y \\ -2(x-1) + 6y \end{pmatrix} \tag{3}$$

Dla tego przykładu stała kroku $\alpha_k = 0.1$ natomiast warunek zakończenia to wartość funkcji mniejsza niż $10^{-4}$ lub ilość iteracji większa niż 1000.

## 3 Przykładowe obliczenia

Dla przykładowego punktu startowego $X_0 = (-2, 3)$ wyznaczenie 5 kolejnych punktów wygląda następująco:

$$X_1 = \begin{pmatrix} -2 \\ 3 \end{pmatrix} - 0.1 \begin{pmatrix} 6(-2-1) - 2 \cdot 3 \\ -2 \cdot (-2-1) + 6 \cdot 3 \end{pmatrix} = \begin{pmatrix} 0.4 \\ 0.6 \end{pmatrix}$$

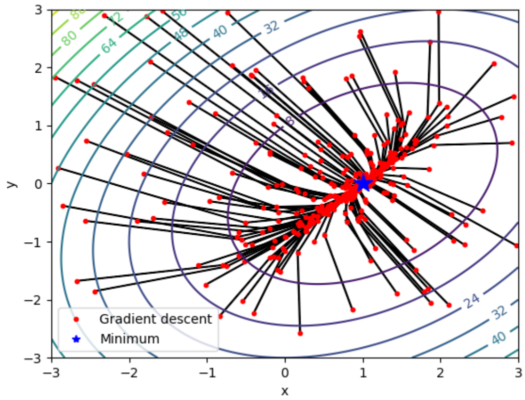
$$X_2 = \begin{pmatrix} 0.4 \\ 0.6 \end{pmatrix} - 0.1 \begin{pmatrix} 6(0.4-1) - 2 \cdot 0.6 \\ -2 \cdot (0.4-1) + 6 \cdot 0.6 \end{pmatrix} = \begin{pmatrix} 0.88 \\ 0.12 \end{pmatrix}$$

$$X_3 = \begin{pmatrix} 0.88 \\ 0.12 \end{pmatrix} - 0.1 \begin{pmatrix} 6(0.88-1) - 2 \cdot 0.12 \\ -2 \cdot (0.88-1) + 6 \cdot 0.12 \end{pmatrix} = \begin{pmatrix} 0.976 \\ 0.024 \end{pmatrix}$$

$$X_4 = \begin{pmatrix} 0.976 \\ 0.024 \end{pmatrix} - 0.1 \begin{pmatrix} 6(0.976-1) - 2 \cdot 0.024 \\ -2 \cdot (0.976-1) + 6 \cdot 0.024 \end{pmatrix} = \begin{pmatrix} 0.9952 \\ 0.0048 \end{pmatrix}$$

$$X_5 = \begin{pmatrix} 0.9952 \\ 0.0048 \end{pmatrix} - 0.1 \begin{pmatrix} 6(0.9952-1) - 2 \cdot 0.0048 \\ -2 \cdot (0.9952-1) + 6 \cdot 0.0048 \end{pmatrix} = \begin{pmatrix} 0.99904 \\ 0.00096 \end{pmatrix}$$

Dla tego przykładu wartość funkcji w punkcie $X_5$ wynosi $f(X_5) = 7.3727 \cdot 10^{-6}$, algorytm się kończy.




Rysunek 2: Rzut 2D funkcji $f(x, y)$ oraz kolejne kroki minimalizacji metodą gradientową dla wylosowanych 100 punktów startowych

Po wykonaniu obliczeń dla 100 losowo wybranych punktów startowych średnia wartość iteracji do osiągnięcia minimum wynosiła około 10.

## 4 Wnioski

Metoda gradientowa jest efektywną metodą szukania minimum funkcji, która polega na iteracyjnym kroku w kierunku przeciwnym do gradientu funkcji w danym punkcie. Metoda ta wymaga znajomości pochodnych cząstkowych funkcji, co może być trudne lub czasochłonne dla bardziej skomplikowanych funkcji.

W przypadku funkcji o jednym minimum globalnym, metoda gradientowa zwykle osiąga minimum w niewielkiej liczbie kroków. Jednak w przypadku funkcji z wieloma minimami lokalnymi, metoda ta może "utknąć"w minimum lokalnym i nie osiągnąć minimum globalnego.

## 5 Kod programu

```
import numpy as np
import matplotlib.pyplot as plt

RANGE = 3
```

```python
def f(x, y):
    return 3 * (x - 1) ** 2 - 2 * (x - 1) * y + 3 * y ** 2


def gradient(x, y):
    grad = np.array([6 * (x - 1) - 2 * y, -2 * (x - 1) + 6 * y])
    return grad


def gradient_descent(gradient, w0):
    w = w0
    w_hist = [w0]
    k = 1
    while True:
        w = w - 0.1 * gradient(w[0], w[1])
        w_hist.append(w)
        if f(*w) < 1e-4 or k > 1000:
            break
        k += 1
    return w_hist, k


x = np.linspace(-RANGE, RANGE, 1000)
y = np.linspace(-RANGE, RANGE, 1000)

X, Y = np.meshgrid(x, y)
Z = f(X, Y)

fig = plt.figure()
ax = fig.add_subplot(projection='3d')
ax.plot_surface(X, Y, Z, cmap='viridis', edgecolor='none')
ax.set_xlabel('x')
ax.set_ylabel('y')
ax.set_zlabel('z')
ax.view_init(30, 45)
plt.savefig('function.png')
plt.show()
k_hist = []
for i in range(100):
    w_hist, k = gradient_descent(gradient, np.array([
        np.random.uniform(-RANGE, RANGE),
        np.random.uniform(-RANGE, RANGE)
    ]))

    k_hist.append(k)

    for i in range(len(w_hist)):
        plt.plot(w_hist[i][0], w_hist[i][1], 'r.')
        if i < len(w_hist) - 1:
            plt.arrow(w_hist[i][0], w_hist[i][1], w_hist[i + 1][0] -
                      w_hist[i][0], w_hist[i + 1][1] - w_hist[i][1],
                      color='k', width=0.01)

cp = plt.contour(X, Y, Z, 15)
plt.clabel(cp, inline=True, fontsize=10)
plt.xlabel('x')
plt.ylabel('y')

plt.plot(1, 0, 'b*', markersize=15)
plt.plot([], [], 'r.', label='Gradient descent')
plt.plot([], [], 'b*', label='Minimum')
plt.legend()
```

```
print("Average number of iterations: ", np.mean(k_hist))
plt.savefig('plot.png')
plt.show()
```