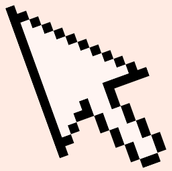
File Edit Format View

KENAN KAYEMBE
FURKAN SALMAN

Artificial Intelligence

B *I* U

Lab 1: Report



Function implementation

main.py

```python
import numpy as np


# x -> vector contains x1 , x2 and x3 consequently
# a,b,c -> coefficients from user input
def func(x, a, b, c):
    return abs(a) * (1 - x[0]) ** 2 + abs(b) * (x[1] - x[0] ** 2) ** 2 + abs(c) * (x[2] - x[1] ** 2) ** 2


# grad -> [∂x1,∂x2,∂x3]
def gradient(x, a, b, c):
    grad = np.zeros(3, dtype=np.longdouble)  # initializing a numpy array with zeros.
    grad[0] = -2 * abs(a) * (1 - x[0]) - 4 * abs(b) * x[0] * (x[1] - x[0] ** 2)  # mathematical equation to compute ∂x1
    grad[1] = 2 * abs(b) * (x[1] - x[0] ** 2) - 4 * abs(c) * x[1] * (x[2] - x[1] ** 2)  # ∂x2
    grad[2] = 2 * abs(c) * (x[2] - x[1] ** 2)  # ∂x3
    return grad


# alpha -> learning rate
```

Calculating function

Gradient calculations

Function implementation

```python
# max_iter -> maximum number of iterations till convergence
def gradient_descent(a, b, c, x_init=None, alpha=1e-3, tol=1e-3, max_iter=100000):
    if x_init is None:  # if X parameter isn't given.
        x_init = np.random.uniform(low=-10, high=10, size=3)  # randomly selecting the starting points for x1,x2,x3
    x = x_init.copy()
    for i in range(max_iter):
        grad = gradient(x, a, b, c)
        if np.linalg.norm(np.abs(grad)) < tol:  # checking if the magnitude of gradient vector is less than tol
            print("Breaking the loop at iteration ", i)
            break  # f(x) converged to the minimum, there is no need to continue optimizing
        x -= alpha * grad  # x1 = x1 - alpha*∂x1 , x2 = x2 - alpha*∂x2 , x3 = x3 - alpha∂dx3
        old_grad = grad
    return x, func(x, a, b, c)
```

Optimizing using gradient

User prompt and final result implementation

```
# parameters: a, b, c, alpha and max_iter will be asked to user:
a = float(input("Enter a value for parameter a: "))
b = float(input("Enter a value for parameter b: "))
c = float(input("Enter a value for parameter c: "))
alpha = float(input("Enter a value for alpha: "))
max_iter = int(input("Enter a value for max_iter: "))

x_star, f_x_star = gradient_descent(a, b, c, alpha=alpha, max_iter=max_iter)

print("Minimum point x*: ", x_star)
print("Minimum value f(x*): ", f_x_star)
```

Iteration value effect on convergence

```
Enter a value for parameter a: 3
Enter a value for parameter b: 4
Enter a value for parameter c: 5
Enter a value for alpha: 1e-3
Enter a value for max_iter: 1000
Minimum point x*:  [-1.50476779  2.88640974  8.41504352]
Minimum value f(x*):  20.404550921722926

Process finished with exit code 0
```

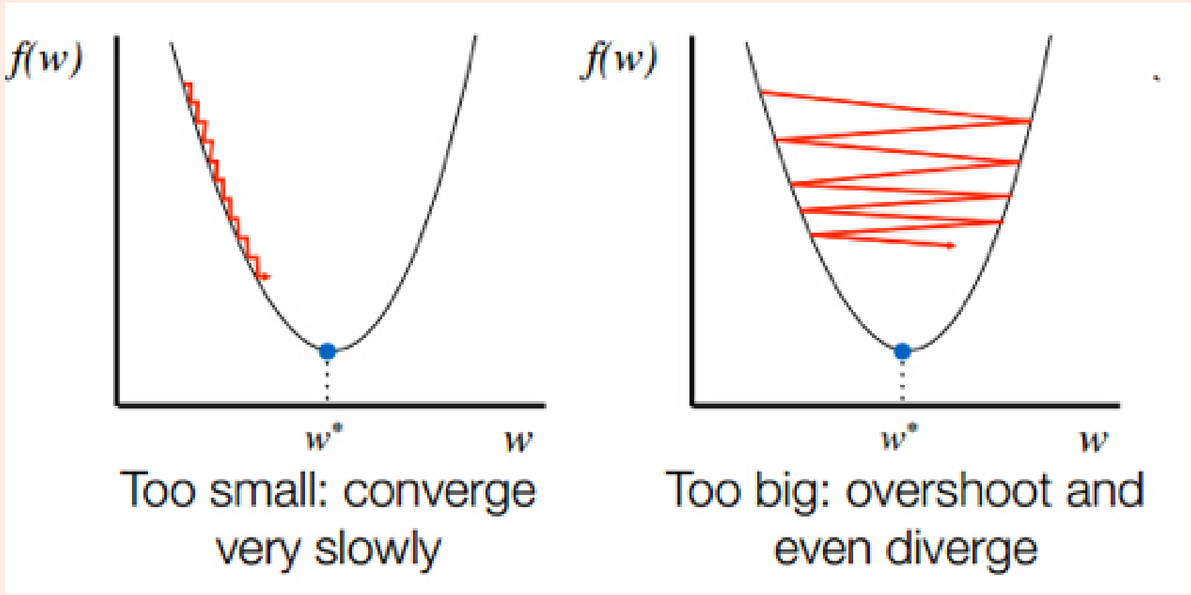
Example 1 - small iteration value

```
Enter a value for parameter a: 3
Enter a value for parameter b: 4
Enter a value for parameter c: 5
Enter a value for alpha: 1e-3
Enter a value for max_iter: 100000
Breaking the loop at iteration  39590
Minimum point x*:  [1.00076483 1.00180545 1.00370255]
Minimum value f(x*):  2.0969130081289946e-06

Process finished with exit code 0
```

Example 2 - sufficient iteration value

Alpha value effect on convergence

```
Enter a value for parameter a: 3
Enter a value for parameter b: 4
Enter a value for parameter c: 5
Enter a value for alpha: 1e-6
Enter a value for max_iter: 100000
Minimum point x*:  [-0.03236651  0.00297531 -2.38858822]
Minimum value f(x*):  31.724336545642974

Process finished with exit code 0
```



Example 3 - Small alpha value

Example 4- High Alpha Value

```
Enter a value for parameter a: 2
Enter a value for parameter b: 1
Enter a value for parameter c: 4
Enter a value for alpha: 2
Enter a value for max_iter: 3
Minimum point x*:  [1.09548000e+39 2.23324747e+44 5.84311676e+29]
Minimum value f(x*):  9.949640699269424e+177

Process finished with exit code 0
```