Computational Methods of Optimization
Assignment 01

Submitted by: Shatakshi Gupta

MTech. Artificial Intelligence
SR Number: 2288

9th September, 2023

Question 2:

Consider the polynomial

$$p(x,y) = x^4y^2 + x^2y^4 - 9x^2y^2.$$

Find $f^* = \inf_{x,y} p(x,y)$. Is $x = [0,0]^T$ a stationary point, and is it the global minimum? If not, can you identify a global minimum for this function? Is it unique (that is, if x^* is the unique global minimum, then $f(x^*) > f(x)$ for all $x \neq x^*$)?

Answer 2:

Given,

$$p(x,y) = x^4y^2 + x^2y^4 - 9x^2y^2$$

$$\lim_{(x,y)\to\infty} p(x,y) = \lim_{(x,y)\to\infty} x^4y^2 + x^2y^4 - 9x^2y^2$$
$$= \infty$$

This means the function is coercive.
Since the function is coercive we now know that there exists a global minima for the given function.

To find critical points:

$\nabla p(x,y) = \begin{bmatrix} \frac{\partial p}{\partial x} \\ \frac{\partial p}{\partial y} \end{bmatrix} = 0$

$\Rightarrow \nabla p(x,y) = \begin{bmatrix} 4x^3y^2 + 2xy^4 - 18xy^2 \\ 2x^2y + 4x^2y^3 - 18x^2y \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

$\Rightarrow \nabla p(x,y) = \begin{bmatrix} 2xy^2(2x^2 + y^2 - 9) \\ 2x^2y(x^2 + 2y^2 - 9) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

Thus we have
$xy^2 = 0$ or $(2x^2 + y^2 - 9) = 0$
and $x^2y = 0$ or $(x^2 + 2y^2 - 9) = 0$

solving which we get the critical points as:
(x,y)=$\{(+\sqrt{3},+\sqrt{3}),(+\sqrt{3},-\sqrt{3}),(-\sqrt{3},+\sqrt{3}),(-\sqrt{3},-\sqrt{3}),(0,y_1),(x_1,0)\}$

Hence we can see that $\begin{bmatrix} 0 & 0 \end{bmatrix}^T$ is a critical point for the given function $p(x,y)$.

Now, to check for global minimum , we need to find the double derivatives of the function. The same is calculated as follows:

$\Rightarrow \nabla^2 p(x,y) = \begin{bmatrix} \frac{\partial^2 p}{\partial x^2} & \frac{\partial^2 p}{\partial x \partial y} \\ \frac{\partial^2}{\partial y \partial x} & \frac{\partial^2 p}{\partial y^2} \end{bmatrix}$

$= \begin{bmatrix} 12x^2y^2 + 2y^4 - 18y^2 & 8x^3y + 8xy^3 - 36xy \\ 8x^3y + 8xy^3 - 36xy & 2x^4 + 12x^2y^2 - 18x^2 \end{bmatrix}$

$= H(x,y)$

$H(x_1, 0) = \begin{bmatrix} 0 & 0 \\ 0 & 2x_1^4 - 18x_1^2 \end{bmatrix}$ eigen values$=\{2x_1^4 - 18x_1^2, 0\}$

$H(0, y_1) = \begin{bmatrix} 2y_1^4 - 18y^2 & 0 \\ 0 & 0 \end{bmatrix}$ eigen values$=\{0, 2y_1^4 - 18y_1^2\}$

$H(+\sqrt{3}, +\sqrt{3}) = \begin{bmatrix} 72 & 36 \\ 36 & 72 \end{bmatrix}$ eigen values$=\{108, 36\}$

$H(+\sqrt{3}, -\sqrt{3}) = \begin{bmatrix} 72 & -36 \\ -36 & 72 \end{bmatrix}$ eigen values$=\{108, 36\}$

$H(-\sqrt{3}, +\sqrt{3}) = \begin{bmatrix} 72 & -36 \\ -36 & 72 \end{bmatrix}$ eigen values$=\{108, 36\}$

$H(-\sqrt{3}, -\sqrt{3}) = \begin{bmatrix} 72 & 36 \\ 36 & 72 \end{bmatrix}$ eigen values$=\{108, 36\}$

$\Rightarrow$ The Hessian is positive definite.

Thus we can proceed with finding the value of function at critical points.

$p(x_1, 0) = 0$
$p(0, y_1) = 0 \quad \Rightarrow p(0,0) = 0$
$p(+\sqrt{3}, +\sqrt{3}) = -27$
$p(+\sqrt{3}, -\sqrt{3}) = -27$
$p(-\sqrt{3}, +\sqrt{3}) = -27$
$p(-\sqrt{3}, -\sqrt{3}) = -27$

Since we have a function value lower than that at $\begin{bmatrix} 0 & 0 \end{bmatrix}^T$, i.e.,$p(x,y) < 0$ for values that are not $\begin{bmatrix} 0 & 0 \end{bmatrix}^T$, therefore $\begin{bmatrix} 0 & 0 \end{bmatrix}^T$is not the global minima.

$\therefore$ The global minimum of $p(x,y)$ is -27 and it is not a unique global minimum.
i.e.,$f^* = inf_{x,y} p(x,y) = -27$ and is attained and points ,$(+\sqrt{3}, +\sqrt{3})$,$(+\sqrt{3}, -\sqrt{3})$, $(-\sqrt{3}, +\sqrt{3})$ and $(-\sqrt{3}, -\sqrt{3})$.

Question 3:

Consider the function

$$f(x) = \frac{e^{x^\top Ax} e^{-x^\top (B+C)x}}{1 + e^{-x^\top (C-B)x}}.$$

Suppose $\lambda_{\min}(B) = 1$, $\lambda_{\max}(B) = 4$. Can you find the range of values of $\lambda_{\min}(A)$, $\lambda_{\min}(C)$, $\lambda_{\max}(A)$, and $\lambda_{\max}(C)$ such that $f(x)$ is a coercive function?

Answer 3:

Given,

$$f(x) = e^{x^TAx}\frac{e^{-x^T(B+C)x}}{1+e^{-x^T(C-B)x}}$$

To determine the range of values for the eigenvalues $\lambda_{min}(A)$,$\lambda_{max}(A)$,$\lambda_{min}(C)$ and $\lambda_{max}(C)$ such that $f(x)$ is a coercive function, we need to analyze the behavior of $f(x)$ as $||x||$ approaches ∞.

If A is an $n \times n$ matrix, $\lambda_{\text{min}}(A)$ and $\lambda_{\text{max}}(A)$ are the minimum and maximum eigenvalues of A, then

$$\lambda(A)_{\text{min}}\, ||x||_2 \leq x^TAx \leq \lambda(A)_{\text{max}}\, ||x||_2 \tag{1}$$

If matrix A has eigenvalues $\lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_{n-1} \leq \lambda_n$, then matrix $-A$ has eigenvalues $-\lambda_n \leq -\lambda_{n-1} \leq \ldots \leq -\lambda_2 \leq -\lambda_1$. Therefore,

$$\lambda(A)_{\min} = \lambda_1 = -\lambda(-A)_{\max} \tag{2}$$

Given A, B, and C are symmetric matrices, let's break down the terms:

1. e^{x^TAx}:
This term will grow large if $\lambda_{\min}(A) > 0$ and $\lambda_{\max}(A)$ is unbounded.

2. $e^{-x^T(B+C)x}$:
This term will converge to zero as $||x||$ goes to infinity. Hence we want $(B+C)$ to be negative.

Using (1) the function can also be written as :

$$f(x) \geq (g(x) = e^{\lambda_{\min}(A)\|x\|^2}\frac{e^{-\lambda_{\max}(B+C)\|x\|^2}}{1+e^{-\lambda_{\min}(C-B)\|x\|^2}}$$

from here we can see that we want $-\lambda_{\min}(C-B) \leq 0$ i.e., $\lambda_{\max}(B-C) \leq 0$

We can write g(x) as :$(g(x) = \dfrac{e^{\lambda_{\min}(A)\|x\|^2-\lambda_{\max}(B+C)\|x\|^2+\lambda_{\min}(C-B)\|x\|^2}}{1+e^{\lambda_{\min}(C-B)\|x\|^2}}$

As $g(x)$ approaches ∞ as $||x||$ goes to infinity, $f(x)$ also approaches ∞. For $g(x)$ to approach ∞ as $||x||$ goes to infinity$e^{\lambda_{\min}(C-B)\|x\|^2}$ should approach zero.
For this to happen, we need $\lambda_{\min}(C-B) < 0$.

And $e^{\lambda_{\min}(A)\|x\|^2-\lambda_{\max}(B+C)\|x\|^2+\lambda_{\min}(C-B)\|x\|^2} = e^{(\lambda_{\min}(A)-\lambda_{\max}(B+C)+\lambda_{\min}(C-B))\|x\|^2}$ should become very large.
i.e.,$\lambda_{\min}(A) - \lambda_{\max}(B+C) + \lambda_{\min}(C-B)$ should be positive.

Weyl's inequality provides a bound on the eigenvalues of a sum of Hermitian matrices in terms of the eigenvalues of the individual matrices. For symmetric matrices, the same inequality applies.

The inequality states that for two Hermitian matrices X and Y, and for each index i, we have:

$$\lambda_i(X) + \lambda_1(Y) \leq \lambda_i(X+Y) \leq \lambda_i(X) + \lambda_n(Y) \tag{3}$$

where $\lambda_i(X)$ denotes the i-th eigenvalue of X, and $\lambda_1(Y)$ and $\lambda_n(Y)$ denote the smallest and largest eigenvalues of Y, respectively.

Thus we can say that,
$\lambda(A)_{\min} - [\lambda(B)_{\max} + \lambda(C)_{\max}] + [\lambda(C)_{\min} + \lambda(-B)_{\min}] \leq \lambda_{\min}(A) - \lambda_{\max}(B+C) + \lambda_{\min}(C-B)$

$\Rightarrow$ Using equation (3) we can say,
$\lambda(A)_{\min} - [\lambda(B)_{\max} + \lambda(C)_{\max}] + [\lambda(C)_{\min} + \lambda(-B)_{\min}] = \lambda(A)_{\min} - [\lambda(B)_{\max} + \lambda(C)_{\max}] + [\lambda(C)_{\min} -$

$\lambda(B)_{\max}]$
$= \lambda_{\min}(A) - 2\lambda_{\max}(B) + [\lambda_{\min}(C) - \lambda_{\max}(C)]$

Given $\lambda_{\min}(B) = 1$ and $\lambda_{\max}(B) = 4$, we can now establish the ranges:

- For $\lambda_{min}(A)$ and $\lambda_{max}(A)$:
  $= \lambda_{\min}(A) - 2\lambda_{\max}(B) + [\lambda_{\min}(C) - \lambda_{\max}(C)] > 0$
  $\Rightarrow \lambda_{\min}(A) > 2\lambda_{\max}(B) - [\lambda_{\min}(C) - \lambda_{\max}(C)]$
  $\Rightarrow \lambda_{\min}(A) > 8 - [\lambda_{\min}(C) - \lambda_{\max}(C)]$ and $\lambda_{\max}(A)$ is unbounded(as found earlier).

- For $\lambda_{min}(C)$ and $\lambda_{max}(C)$:
  By using Weyl's inequality $\lambda_{\min}(C) + \lambda_{\min}(-B) < \lambda_{\min}(C - B) < \lambda_{\max}(C) + \lambda_{\max}(-B)$
  Also, we know from above that $\lambda_{\min}(C - B) \leq 0$.
  $\therefore$ We can say that $\lambda_{\min}(C) + \lambda_{\min}(-B) \leq 0$
  $\Rightarrow \lambda_{\min}(C) \leq -\lambda_{\min}(-B)$
  $\Rightarrow \lambda_{\min}(C) \leq \lambda_{\max}(B)$
  $\Rightarrow \lambda_{\min}(C) \leq 4$
  Also, we know from above that $\lambda_{\max}(B - C) \leq 0$
  $\lambda_{\min}(B) + \lambda_{\min}(-C) < \lambda_{\max}(B - c) < \lambda_{\max}(B) + \lambda_{\max}(-C)$
  $\therefore$ We can say that $\lambda_{\min}(B) + \lambda_{\min}(-C) \leq 0$
  $\Rightarrow \lambda_{\min}(B) - \lambda_{\max}(C) \leq 0$
  $\Rightarrow 1 - \lambda_{\max}(C) \leq 0$
  $\Rightarrow \lambda_{\max}(C) \geq 1$

Thus the range of values are as follows:
$\lambda_{min}(A) > 8 - [\lambda_{\min}(C) - \lambda_{\max}(C)]$,
$\lambda_{max}(A)$ is unbounded ,
$\lambda_{\max}(C) \geq 1$ and,
$\lambda_{min}(C) \leq 4$

Question 4:

You are each given 100 pairs of data points (x_i, y_i), where $x_i \in \mathbb{R}^5$ and $y_i \in \mathbb{R}$. We know that the data is generated by the equation:

$$y_i = w^\top x_i + b.$$

Using the provided data, find w that minimizes the least squares error between y_i and $w^\top x_i + b$. Furthermore, for the general case where $x \in \mathbb{R}^n$ and we are given m data points, what is the closed form solution to this problem? Is this solution unique?
Suppose the number of linearly independent data points is less than n - how would you solve this problem, and is the solution unique?

Answer 4:

The python code for the question is as follows :

```
import subprocess

# Run the external program and capture the output

data =subprocess.run(["./getDataPoints.exe", "22888"], stdout=subprocess.PIPE).stdout.decode("utf-8")
.split('\n')

# Initialize lists to store X and Y data
X_data = []
Y_data = []

for line in data:
    if line.strip():
        y, x1,x2,x3,x4,x5 = line.split(', ')
        x1=x1.replace('[' , '')
        x5=x5.replace(']' , '')
        Y_data.append(float(y))
        X_data.append(float(x1))
        X_data.append(float(x2))
        X_data.append(float(x3))
        X_data.append(float(x4))
        X_data.append(float(x5))
        X_data.append(1)


import numpy as np
import re
# Convert data to NumPy arrays
X = np.array(X_data).reshape(100,6)
Y = np.array(Y_data)

# Calculate X^T
XT = np.transpose(X)

# Calculate X^T X
XTX = np.dot(XT, X)

# Calculate X^T Y
XTY = np.dot(XT, Y)

# Calculate the inverse of X^T X
XTX_inverse = np.linalg.inv(XTX)

# Calculate W
W = np.dot(XTX_inverse, XTY)
b=float(f'{W[-1]:.2f}')
W = np.delete(W,-1)

for i in range(0,5):
    W[i]=float(f'{W[i]:.2f}')

print(W)

print(b)

#end of code.
```

```
The output is :
W=[ 0.18 -0.92  0.84  0.21  0.75]
b=-0.62

#end of output.
```

In the general case where x is a vector in $\mathbb{R}^n$ and we are given m data points, the closed-form solution to the linear regression problem is as follows:

The linear regression model can be represented as $y = Xw + b$, where:

- X is an mxn matrix, where each row represents a data point.
- w is an $n{\times}1$ weight vector.
- b is a scalar bias term.

The closed-form solution for w is given by:

$$w = (X^TX)^{-1}X^Ty$$

We deduced above as :
For $x \in \mathbb{R}^n$ and m data points, the equations are:

$$\begin{aligned} y_1 &= \begin{bmatrix} w_1 & \dots & w_n \end{bmatrix} \begin{bmatrix} x_{11} \\ \vdots \\ x_{1n} \end{bmatrix} + b \\ &\vdots \\ y_m &= \begin{bmatrix} w_1 & \dots & w_n \end{bmatrix} \begin{bmatrix} x_{m1} \\ \vdots \\ x_{mn} \end{bmatrix} + b \end{aligned}$$

can be written as:

$$\begin{bmatrix} y_1 \\ \vdots \\ y_m \end{bmatrix} = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{m1} & \cdots & x_{mn} \end{bmatrix} \begin{bmatrix} w_0 \\ w_1 \\ \vdots \\ w_n \end{bmatrix}$$

The square error as a function of weights $E(W)$ can be defined as:

$$E(W) = ||\text{actual values} - XW||^2$$

Thus we have,

$$EW = (\text{actual values} - XW)^T(\text{actual values} - XW)$$

let matrix of actual values be y

$$\begin{aligned} EW &= y^Ty - W^TX^Ty - y^TXW + yX^TW^T \\ &= y^Ty - 2W^TX^Ty + XW^TXW \end{aligned}$$

where this development uses the fact that the transpose of a scalar is the scalar i.e.$W^T X^T y$ =$y^T XW$
To find the W that minimizes the error, we need to take the derivative with respect to w. This gives us the following equation: $\frac{\partial^2 EW}{\partial W^2} = -2X^T y + 2X^T XW = 0$
$\Rightarrow X^T y = X^T XW$
$\Rightarrow W = X^T y(X^T X)^{-1}$
The uniqueness of the solution depends on the rank of the matrix $X^T X$. If $X^T X$ is full rank (i.e., has full column rank), then the solution is unique.

However, if the number of linearly independent data points is less than n, meaning that $X^T X$ is not full rank, there are infinitely many solutions, and we'll need to introduce regularization techniques like Ridge (L2 regularization) or Lasso (L1 regularization) to make the solution unique.
To address the issue of linearly dependent data points, you can add regularization terms to the linear regression problem:

- Ridge Regression (L2 Regularization): Minimize $\|y - Xw\|_2^2 + \lambda\|w\|_2^2$ where λ is the regularization parameter. This introduces bias towards smaller weight values.

- Lasso Regression (L1 Regularization): Minimize $\|y - Xw\|_2^2 + \lambda\|w\|_1$ where λ is the regularization parameter. This introduces sparsity in the weight vector.

These regularization techniques can help in obtaining a unique solution even when the data is linearly dependent.

Question 5:

You are each given a black box function which returns $f(x)$ and $\nabla f(x)$. Use the following iteration to try and find a minimum:

$$x_{k+1} = x_k - \eta \nabla f(x_k)$$

Starting at $x_0 = [0, 0]$, how many iterations will it take until you reach an ε-approximate point if (a) $\varepsilon = 0.01$, (b) $\varepsilon = 0.001$, for $\eta = 0.8$, $\eta = 0.5$, $\eta = 0.33$, and $\eta = 0.1$?
Plot the trajectories of your algorithm on the $x - y$ plane for each stepsize η. Based on your answers can you say anything about the function?

Answer 5:

The python code for the Number of iterations are as per the table:

Epsilon(ϵ)	Eta(η)	Iterations
0.01	0.8	>100
0.01	0.5	4
0.01	0.33	7
0.01	0.1	27
0.001	0.8	>100
0.001	0.5	4
0.001	0.33	9
0.001	0.1	37

Based on the number of iterations required to reach an ϵ-approximate point for different combinations of ϵ and η, we can make some observations about the function being optimized:

- Sensitivity to ϵ: When ϵ is set to 0.01, the number of iterations varies significantly with different values of η. For example, $\eta = 0.5$ and $\eta = 0.33$ require relatively fewer iterations (4 and 7, respectively), indicating that the function has regions where the gradient descent converges quickly with moderate step sizes. When ϵ is reduced to 0.001, the number of iterations generally increases, suggesting that the function might have regions with steep gradients or multiple local minima that require more precise optimization.

- Sensitivity to η: For $\epsilon = 0.01$, a smaller learning rate (η) results in significantly more iterations (27), indicating that a very small step size leads to slow convergence. This suggests that the function might have regions where a smaller η is not ideal for optimization.
  On the other hand, $\eta = 0.8$ and $\eta = 0.5$ for $\epsilon = 0.01$, as well as $\eta = 0.8$ for $\epsilon = 0.001$, require more than 100 iterations. This could indicate that a large η may cause divergence, suggesting sensitivity to the choice of η.

- Complexity: The function appears to be more complex or challenging to optimize when ϵ is smaller (0.001). Smaller ϵ values require finer convergence and likely imply the presence of steep valleys, sharp turns, or multiple local minima in the function landscape.

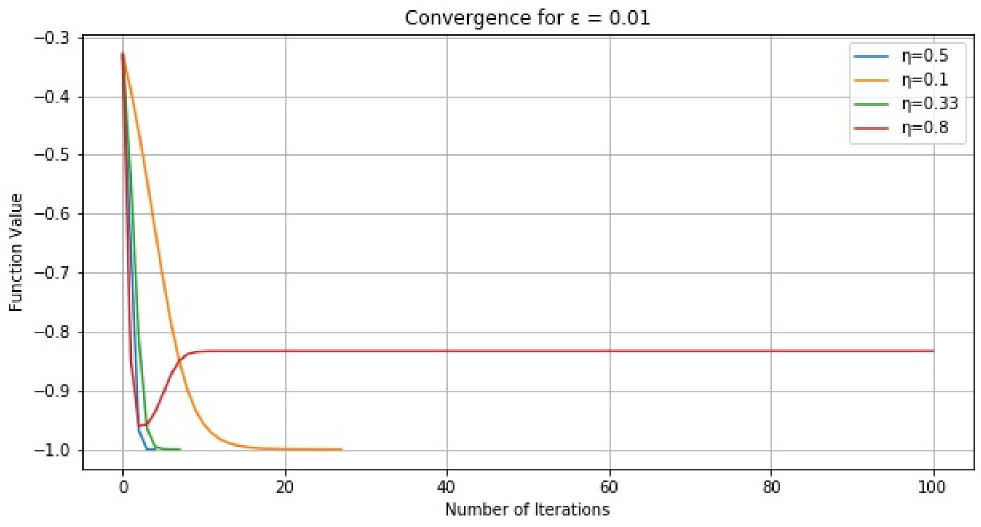
Some combinations of ϵ and η (e.g., $\epsilon = 0.001$ and $\eta = 0.1$) require a relatively high number of iterations (37), indicating that these regions of the function may be particularly challenging to optimize.

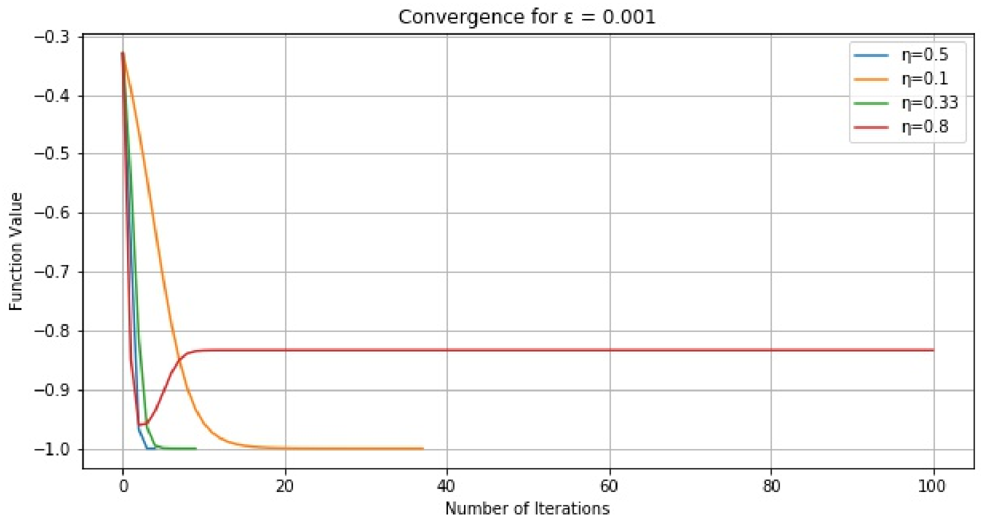
In summary, the behavior of the function, as observed through the gradient descent optimization, suggests that it may be a complex function with regions of steep gradients and possibly multiple local minima. The choice of ϵ and η appears to significantly impact the convergence speed, indicating that different parts of the function landscape may respond differently to the optimization process.

The algotith and the trajectories of the algorithm on the $x - y$ plane for each stepsize η is as below.

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

def get_function_value_and_gradient(x):
    str1=str(x[0])
    str2=str(x[1])
    result = subprocess.run(["./getGradient.exe", "22888", str1, str2], stdout=subprocess.PIPE).stdout
    .decode("utf-8")
    output = result.strip().split(',')
    function_value=float(output[0])
    output[1]=output[1].replace("[" , ' ')
    output[2]=output[2].replace("]" , ' ')
    m=float(output[1])
    n=float(output[2])
    gradient = [m,n]  # Gradient of f(x, y)
    return function_value, gradient

x0 = np.array([0, 0])  # Initial point
epsilons = [0.01,0.001]  # Different epsilon values to test
etas = [0.5,0.1,0.33,0.8]
max_iterartions=100

for epsilon in epsilons:
    plt.figure(figsize=(10, 5))
    plt.title(f'Convergence for e = {epsilon}')

    for eta in etas:
        x = x0
        function_values = []

        # Perform gradient descent iterations
        k = 0
        while True:
            # Call the black box function to get function value and gradient
            function_value, gradient = get_function_value_and_gradient(x)

            # Update x using gradient descent formula
            x_new = x - eta * np.array(gradient)
            #print (x_new)

            function_values.append(function_value)
            # Check the convergence condition
            if np.linalg.norm(gradient) <= epsilon:
                break
            if k==max_iterartions:
                break

            x = x_new
            k += 1

        # Plot trajectories on the x-y plane for each combination
        plt.plot(range(len(function_values)), function_values, label=f'n={eta}')

    # Customize the plot
    plt.xlabel('Number of Iterations')
    plt.ylabel('Function Value')
    plt.legend()
    plt.grid(True)

    # Show the plot
    if(epsilon==0.01):
        plt.savefig("output1.jpg")
        plt.show()
    if(epsilon==0.001):
        plt.savefig("output2.jpg")
        plt.show()
#end of code
```

The plot is as follows:


Convergence for ε = 0.01
η=0.5
η=0.1
η=0.33
η=0.8
Function Value
Number of Iterations


Convergence for ε = 0.001
η=0.5
η=0.1
η=0.33
η=0.8
Function Value
Number of Iterations