Introduction to Optimization

Week 3 Assignment Solution

1a. Explain (simply) why √ π is the minimum of this function

x^2+y^2 will always be greater than or equal to zero (positive) for all values of x and y.

Therefore:

$x^2+y^2+\pi$ = (at least) π.

The function:

$f(x, y) = x^2+y^2+\pi$,

will have its minimum value when x^2+y^2 is minimized, which happens when x=0 and y=0.

Hence, the minimum value of $f(x,y)$ is π, which occurs at the point (0,0).

So, π is the minimum value of the function $f(x,y)$.

1b. Compute the gradients (partial derivatives) of *f* with respect to *x* and *y*.

$f(x) = \sqrt{x^2 + y^2 + \pi}$

= let $g = x^2 + y^2 + \pi$

= f(x) = $\sqrt{g}$

With chain rule, *f* with respect to *x* and *y* becomes:

$=\frac{df}{dx} = \frac{d\sqrt{g}}{dg} \cdot \frac{dg}{dx}$

$=\frac{df}{dy} = \frac{d\sqrt{g}}{dg} \cdot \frac{dg}{dy}$

Differentiate $\sqrt{g}$ with respect to g

$=\frac{d\sqrt{g}}{dg} = \frac{1g^{\frac{1-2}{2}}}{2} = \frac{1g^{\frac{-1}{2}}}{2} = \frac{1}{2\sqrt{g}}$

Partially differentiate $g = x^2 + y^2 + \pi$ with respect to *x*

$=\frac{dg}{dx} = 2x$

Partially differentiate $g = x^2 + y^2 + \pi$ with respect to *y*

$\frac{dg}{dx}$ = $2y$

Now differentiate $\frac{df}{dx} = \frac{d\sqrt{g}}{dg} \cdot \frac{dg}{dx}$

$= \frac{1}{2\sqrt{g}} \cdot 2x$

$= \frac{1}{2\sqrt{x^2+y^2+\pi}} \cdot 2x = \frac{x}{\sqrt{x^2+y^2+\pi}}$

Then differentiate

$\frac{df}{dy} = \frac{d\sqrt{g}}{dg} \cdot \frac{dg}{dy}$

$= \frac{1}{2\sqrt{g}} \cdot 2y$

$= \frac{1}{2\sqrt{x^2+y^2+\pi}} \cdot 2y = \frac{y}{\sqrt{x^2+y^2+\pi}}$

Therefore, the gradients (partial derivatives) of *f with respect to x and y* are:

= $\boldsymbol{\frac{x}{\sqrt{x^2+y^2+\pi}}}$ **and** $\boldsymbol{\frac{y}{\sqrt{x^2+y^2+\pi}}}$

1c.

Using the hint that a function with two variables f has a local maximum or minimum at (a, b) if ∂f/∂x(a, b) = 0 and ∂f/∂y (a, b) = 0),

$\frac{df}{dx}$ and $\frac{df}{dy}$ must be = 0

i.e

$\boldsymbol{\frac{x}{\sqrt{x^2+y^2+\pi}}}$ **= 0, then *x = 0 since*** $\boldsymbol{\sqrt{x^2+y^2+\pi}}$ **will always be positive**

$\boldsymbol{\frac{y}{\sqrt{x^2+y^2+\pi}}}$ **= 0 then *y = 0***

Therefore $f(x,y)$ has a critical point at (x,y)=(0,0)

i.e *f*(0,0)= $\boldsymbol{\sqrt{0^2+0^2+\pi}}$ = $\boldsymbol{\sqrt{\pi}}$

We already know that $\boldsymbol{x^2+y^2 \geq 0}$

Adding $\boldsymbol{\pi}$ to both side,

=$\boldsymbol{x^2+y^2+\pi \geq 0+\pi}$

= $\boldsymbol{x^2+y^2+\pi \geq \pi}$

Taking the square root

$=\sqrt{\boldsymbol{x^2+y^2+\pi}} \geq \sqrt{\boldsymbol{\pi}}$

Therefore, the minimal value of f(x, y) is indeed √ $\boldsymbol{\pi}$.

2a.

Since $X = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ . & . \\ . & . \\ 1 & x_n \end{pmatrix}$, y= $\begin{pmatrix} y_1 \\ y_2 \\ . \\ . \\ y_n \end{pmatrix}$ and ω = $\binom{\omega_0}{\omega_1}$

the 1-norm of a vector u = $\|\mathbf{u}\|_1 = \frac{1}{m}\sum_{i=1}^{m}|u_i|$

Using these 4, the cost function $L(x;\omega)$ rewritten using a matrix form is:

$\boldsymbol{L(x;\omega)=\frac{1}{m}\|y-X\omega\|_1}$

2b.

Using the cost function:

$L(x;\omega)=\frac{1}{m}\sum_{i=1}^{m}|y_i-(\omega_0+\omega_1 x_i)|$

Gradient with respect to $\boldsymbol{\omega_0}$:

Let $yi-(\omega0+\omega1xi)$ = a

$L(x;\omega) = \frac{1}{m}\sum_{i=1}^{m}|a|$

$= \frac{d|a|}{d\boldsymbol{\omega_0}} = \frac{a}{|a|} \cdot \frac{da}{d\boldsymbol{\omega_0}}$

= Since $ei=yi-(\omega_0+\omega_1 x_i)$, we have

$=\frac{da}{d\boldsymbol{\omega_0}} = -1$

$\frac{d|a|}{d\boldsymbol{\omega_0}} = \frac{a}{|a|} \cdot -1 = -\frac{a}{|a|}$

Therefore, $\frac{dL}{d\boldsymbol{\omega_0}} = \frac{1}{m}\sum_{i=1}^{m}\frac{-a}{|a|}$

$= -\boldsymbol{\frac{1}{m}\sum_{i=1}^{m}(yi-(\omega_0+\omega_1 x_i)}$

Gradient with respect to ω_1:

Let $y_i - (\omega_0 + \omega_1 x_i) = a$

$L(x;\omega) = \frac{1}{m}\sum_{i=1}^{m} |a|$

$= \frac{d|a|}{d\boldsymbol{\omega_1}} = \frac{a}{|a|} \cdot \frac{da}{d\boldsymbol{\omega_i}}$

= Since $ei = yi - (\omega_0 + \omega_1 x_i)$, we have

$= \frac{da}{d\boldsymbol{\omega_i}} = -x_i$

$\frac{d|a|}{d\boldsymbol{\omega_i}} = \frac{a}{|a|} \cdot -x_i = -\frac{a}{|a|} x_i$

Therefore, $\frac{dL}{d\boldsymbol{\omega_i}} = \frac{1}{m}\sum_{i=1}^{m} \frac{-a}{|a|} x_i$

$= -\boldsymbol{\frac{1}{m}\sum_{i=1}^{m}(yi - (\omega_0 + \omega_1 x_i)) x_1}$

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

Your Question3 here

We will see if we can find the results of question 1 using the gradient descent. a. Use the results of question 1.b) to find the couple (x, y) that minimizes f(x, y), this time, with the gradient descent :

xn+1 = xn − μ.∂f/∂x(xn, yn)

yn+1 = yn − μ.∂f/∂y(xn, yn)

Use μ = 0.01, (x0, y0) = (7, 12) and define your number of iteration.

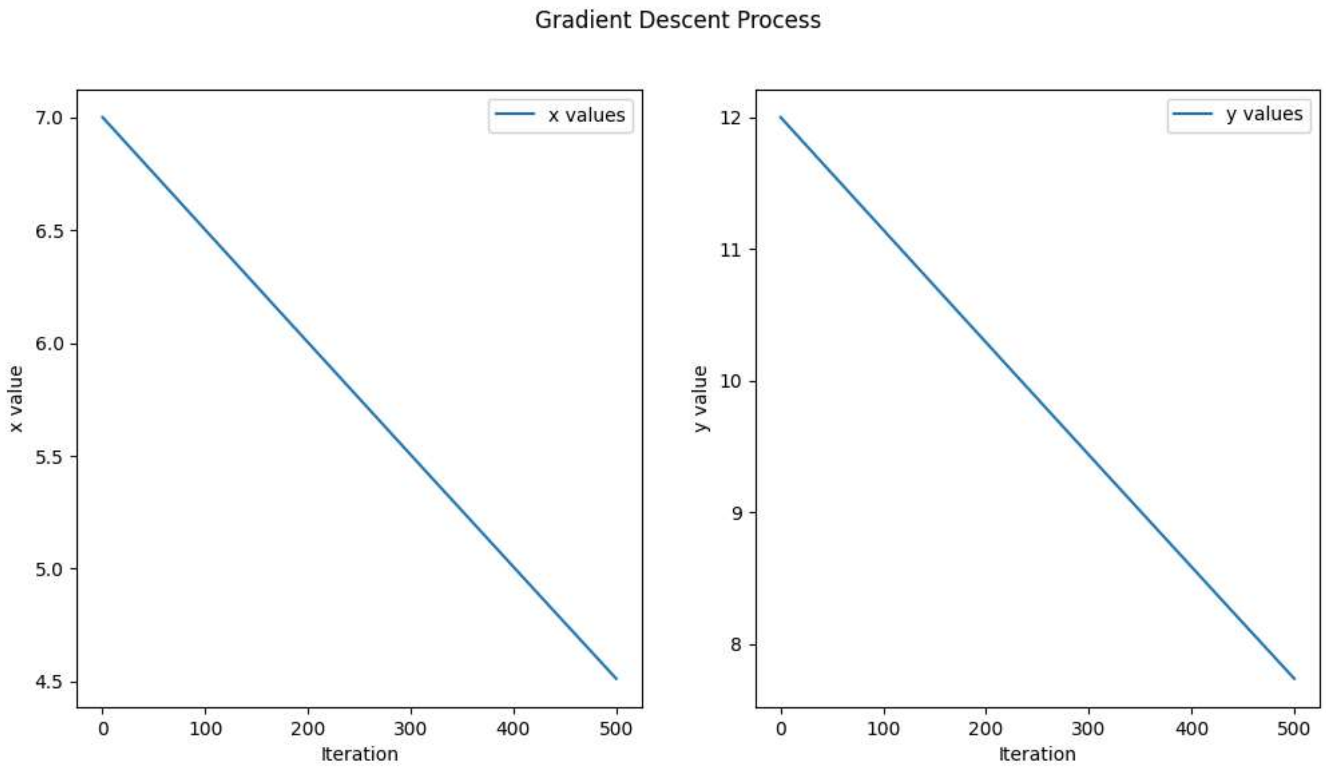
b. Plot the values of x and y during the iterative process.

c. Compute f(xfinal, yfinal) and compare with the result of question 1.

```
# Function f(x, y)
def f(x, y):
    return np.sqrt(x**2 + y**2 + np.pi)

# Gradients of f
def grad_f_x(x, y):
    return x / np.sqrt(x**2 + y**2 + np.pi)

def grad_f_y(x, y):
    return y / np.sqrt(x**2 + y**2 + np.pi)
# Gradient descent parameters
mu = 0.01
x0, y0 = 7, 12
iterations = 500
```

```
# Initialize variables
x, y = x0, y0
x_vals, y_vals = [x], [y]

# Gradient descent loop
for i in range(iterations):
    x_new = x - mu * grad_f_x(x, y)
    y_new = y - mu * grad_f_y(x, y)
    x, y = x_new, y_new
    x_vals.append(x)
    y_vals.append(y)
```

```
# Final values
x_final, y_final = x, y

# Plotting the values of x and y during the iterative process
plt.figure(figsize=(12, 6))
plt.subplot(1, 2, 1)
plt.plot(x_vals, label='x values')
plt.xlabel('Iteration')
plt.ylabel('x value')
plt.legend()
plt.subplot(1, 2, 2)
plt.plot(y_vals, label='y values')
plt.xlabel('Iteration')
plt.ylabel('y value')
plt.legend()
plt.suptitle('Gradient Descent Process')
plt.show()
```



Generate print hello world using rot13 Close

Generate is available for a limited time for unsubscribed users. Upgrade to Colab Pro

```
# Compute f(x_final, y_final)
f_final = f(x_final, y_final)
f_initial = f(x0, y0)

# Print the final results
print(f"Initial value of f(x0, y0): {f_initial}")
print(f"Final value of f(x_final, y_final): {f_final}")
print(f"Final (x, y): ({x_final}, {y_final})")
```

```
Initial value of f(x0, y0): 14.005055967527934
Final value of f(x_final, y_final): 9.128067105247087
Final (x, y): (4.5118271776279135, 7.73456087593357)
```

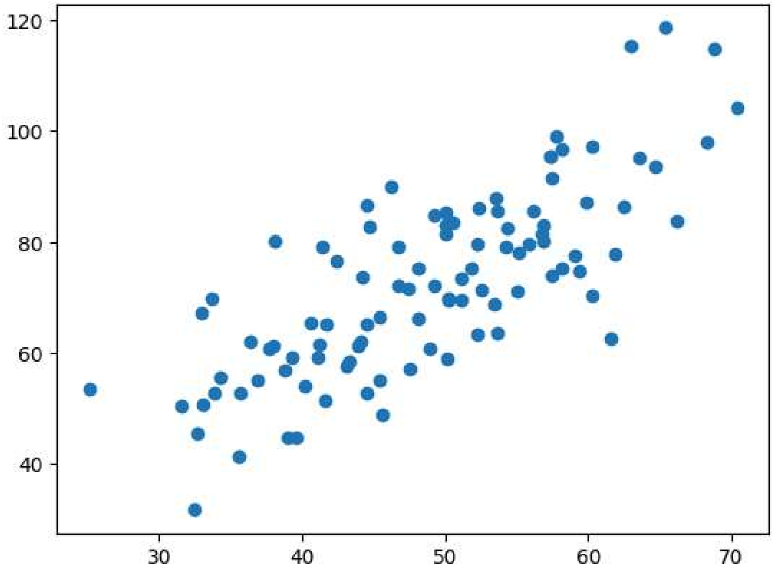
For Question4

```
data = np.loadtxt('Assignment-data_week3.csv', dtype=float, delimiter=',', converters={0:float , 1:float })
```

```
x = data[:,0]
y = data[:,1]
```

```
plt.scatter(x, y)
```

```
<matplotlib.collections.PathCollection at 0x7a76cc2877f0>
```




```
# CONSTRUCT THE DESIGN MAT
X = np.vstack((np.ones_like(x), x)).T
```

```
def MAE(X, y, w_0, w_1):
    """
       Input
        X : design matrix
        y : the responses
        w_0,w_1 : initial guess

        Return
        loss (mean absolute error) : the loss evaluated at (t)

       """
    m = len(y)
    w = [w_0, w_1]
    loss = np.mean(np.abs(y - y_hat))

    return loss
```

```
def computeGradients(x, y, w_0, w_1):
       """
       Input
        x : the treatment or input
        y : the responses
        w_0,w_1 : initial guess

        Return
        grad_0, grad_1: gradients of the loss evaluated at (t)

       """
       # Define y_hat

       y_hat = w_0 + w_1 * x
       error = y - y_hat

       #IMPLEMENT THE gradients with respect to w_0 and w_1

       grad_w1 = -np.mean(np.sign(error))
       grad_w0 =  -np.mean(np.sign(error) * x)
       return [grad_w0, grad_w1]
```

```
# initialize the weight
w_0 = 0
w_1 = 0

# Define the learning parameters
eta =0.01

num_iters =500
```

```
###
y_hat = w_0 + w_1 * x

lossVals = []
for i in range(num_iters):
    loss = MAE(X, y, w_0, w_1)
    lossVals.append(loss)


    # IMPLEMENT THE UPDATE using computeGradients(x, y, w_0, w_1)
    grad_w0, grad_w1 = computeGradients(x, y, w_0, w_1)

    w_0 -= eta * grad_w0
    w_1 -= eta * grad_w1


    #Save the values of the loss in lossVals
    print(f'Final w_0: {w_0}')
    print(f'Final w_1: {w_1}')
```

```
Final w_0: 0.4895834146155935
Final w_1: 0.01
Final w_0: 0.979166829231187
Final w_1: 0.02
Final w_0: 1.4687502438467805
Final w_1: 0.03
Final w_0: 1.958333658462374
Final w_1: 0.04
Final w_0: 2.4479170730779676
Final w_1: 0.05
Final w_0: 2.937500487693561
Final w_1: 0.060000000000000005
Final w_0: 3.4270839023091546
Final w_1: 0.07
Final w_0: 3.916667316924748
Final w_1: 0.08
Final w_0: 4.406250731540341
Final w_1: 0.09
Final w_0: 4.895834146155934
Final w_1: 0.09999999999999999
Final w_0: 5.385417560771527
Final w_1: 0.10999999999999999
Final w_0: 5.87500097538712
Final w_1: 0.11999999999999998
Final w_0: 6.3645843900027135
Final w_1: 0.12999999999999998
Final w_0: 6.8541678046183065
Final w_1: 0.13999999999999999
Final w_0: 7.3437512192339
Final w_1: 0.15
Final w_0: 7.833334633849493
Final w_1: 0.16
Final w_0: 8.322918048465086
Final w_1: 0.17
Final w_0: 8.812501463080679
Final w_1: 0.18000000000000002
Final w_0: 9.302084877696272
Final w_1: 0.19000000000000003
Final w_0: 9.791668292311865
Final w_1: 0.20000000000000004
Final w_0: 10.281251706927458
Final w_1: 0.21000000000000005
Final w_0: 10.770835121543051
Final w_1: 0.22000000000000006
Final w_0: 11.260418536158644
Final w_1: 0.23000000000000007
Final w_0: 11.750001950774237
Final w_1: 0.24000000000000007
Final w_0: 12.23958536538983
Final w_1: 0.25000000000000006
Final w_0: 12.729168780005423
```

```
Final w_1: 0.26000000000000006
Final w_0: 13.218752194621016
Final w_1: 0.2700000000000001
Final w_0: 13.70833560923661
Final w_1: 0.2800000000000001
Final w_0: 14.197919023852203
```

```
#==============================
# visualizing the loss function
#===============================

plt.plot(lossVals)
plt.xlabel('Iteration')
plt.ylabel('Mean Absolute Error')
plt.title('Error Evolution During Gradient Descent')
plt.show()

# Plot the regression line
plt.scatter(x, y, label='Data')
plt.plot(x, w_0 + w_1 * x, color='red', label='Fitted Line')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Linear Regression Fit')
plt.legend()
plt.show()
```