ECE 662 - Deep Learning with Python
Home Work 1

Richard Groves

September 7, 2023

1 Problem-1
Analysis of Gradient Descent and Ascent on a Given Cost Function

1.1 Background

In the Python example that illustrated gradient descent, a cost function was defined as $C(x, y) = \frac{1}{2}\left(\cos^2(x) + \sin^2(y)\right)$. The gradient descent path, along with all potential local minimum points within the region $-\frac{5\pi}{2} \leq x \leq \frac{5\pi}{2}, -2\pi \leq y \leq 2\pi$, were depicted on a plot.

1.2 Part-A
Verification of Local Minimum Points

To confirm that all the identified points were local minimums and not local maximums, the Hessian matrix of C(x,y) was analyzed. This matrix, which contains the second partial derivatives of the function, assisted in determining the nature of the critical points (local minimum or local maximum).

1.2.1 Python Code

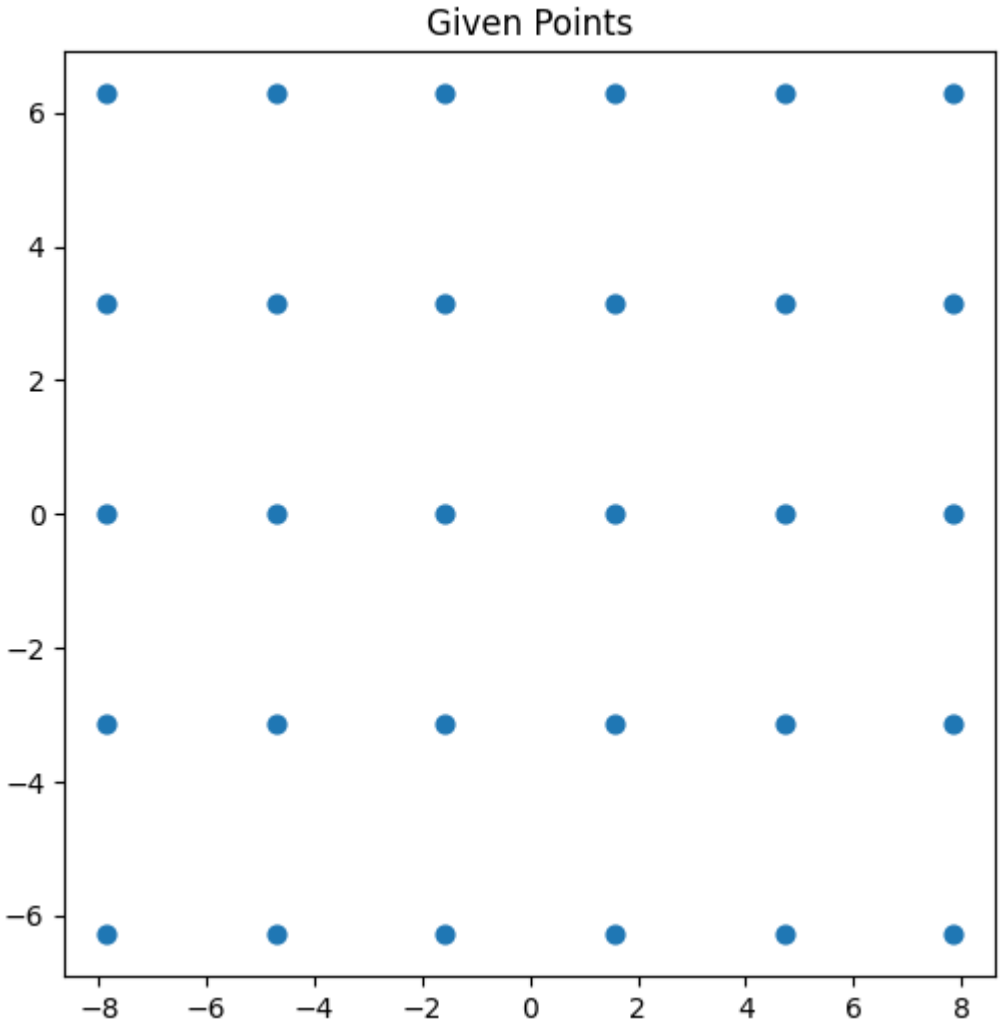
```
import numpy as np
import matplotlib.pyplot as plt
import math
from sympy import symbols, cos, sin, hessian, pprint
pi = math.pi

x, y = symbols('x y')
C = 0.5*(cos(x)**2 + sin(y)**2)
H = hessian(C, (x, y))

print('-----------hessian-----------------')
```

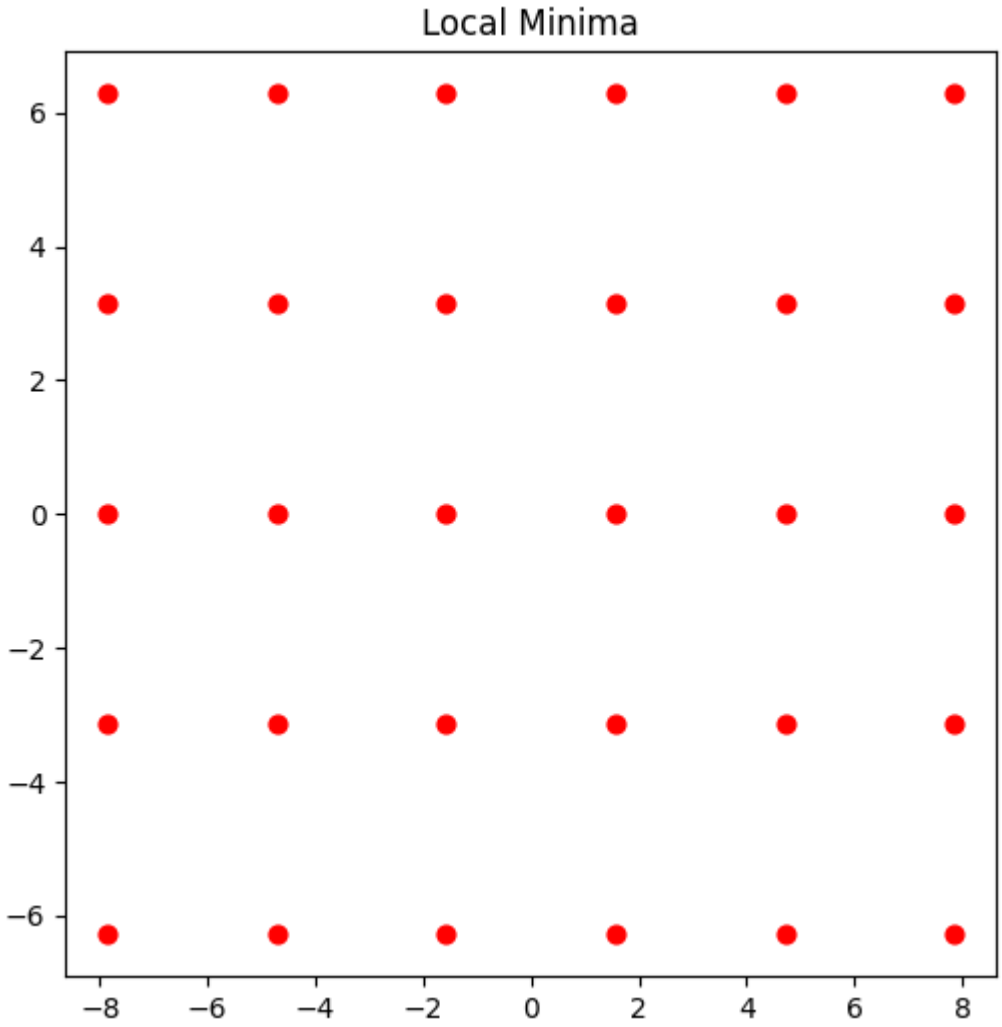
```
pprint(H)
print('')

xx, yy = np.meshgrid(np.arange(-5*pi/2, 7*pi/2, pi), np.arange(-2*pi, 3*pi, pi))
critical_points = list(zip(xx.ravel(), yy.ravel()))

print('------------critical_points-------------')
pprint(critical_points)
print('')

local_minima = []

for point in critical_points:
    point_dict = {x: point[0], y: point[1]}
    H_at_point = H.subs(point_dict)
    eigenvalues = [float(val.evalf()) for val in H_at_point.eigenvals()]
    if all(val > 0 for val in eigenvalues):
        print(f"The point {point} is a local minimum.")
        local_minima.append(point)

fig = plt.figure(num=1, figsize=[6, 6], edgecolor='red')
plt.scatter(xx, yy, s=40)
plt.title('Given Points')
plt.show(block=False)

fig = plt.figure(num=2, figsize=[6, 6], edgecolor='red')
local_minima_x, local_minima_y = zip(*local_minima)
plt.scatter(local_minima_x, local_minima_y, color='red', s=40)
plt.title('Local Minima')
plt.show(block=False)
```

1.2.2 Python Output

```
-----------hessian-----------------
{\small\fbox{23A1}}        2         2                            {\small\fbox{23A4}}
{\small\fbox{23A2}} 1.0{\small\fbox{22C5}} sin (x) - cos (x)             0           {\small
{\small\fbox{23A2}}                                                 {\small\fbox{23A5}}
{\small\fbox{23A2}}                              2             2    {\small\fbox{23A5}}
{\small\fbox{23A3}}           0            - sin (y) + 1.0{\small\fbox{22C5}} cos (y){\small
------------critical_points-------------
[(-7.853981633974483, -6.283185307179586), (-4.71238898038469, -6.283185307179
586), (-1.5707963267948966, -6.283185307179586), (1.5707963267948966, -6.28318
5307179586), (4.71238898038469, -6.283185307179586), (7.853981633974483, -6.28
3185307179586), (-7.853981633974483, -3.141592653589793), (-4.71238898038469,
-3.141592653589793), (-1.5707963267948966, -3.141592653589793), (1.57079632679
48966, -3.141592653589793), (4.71238898038469, -3.141592653589793), (7.8539816
```

```
33974483, -3.141592653589793), (-7.853981633974483, 0.0), (-4.71238898038469,
0.0), (-1.5707963267948966, 0.0), (1.5707963267948966, 0.0), (4.71238898038469
, 0.0), (7.853981633974483, 0.0), (-7.853981633974483, 3.141592653589793), (-4
.71238898038469, 3.141592653589793), (-1.5707963267948966, 3.141592653589793),
 (1.5707963267948966, 3.141592653589793), (4.71238898038469, 3.141592653589793
), (7.853981633974483, 3.141592653589793), (-7.853981633974483, 6.283185307179
586), (-4.71238898038469, 6.283185307179586), (-1.5707963267948966, 6.28318530
7179586), (1.5707963267948966, 6.283185307179586), (4.71238898038469, 6.283185
307179586), (7.853981633974483, 6.283185307179586)]
The point (-7.853981633974483, -6.283185307179586) is a local minimum.
The point (-4.71238898038469, -6.283185307179586) is a local minimum.
The point (-1.5707963267948966, -6.283185307179586) is a local minimum.
The point (1.5707963267948966, -6.283185307179586) is a local minimum.
The point (4.71238898038469, -6.283185307179586) is a local minimum.
The point (7.853981633974483, -6.283185307179586) is a local minimum.
The point (-7.853981633974483, -3.141592653589793) is a local minimum.
The point (-4.71238898038469, -3.141592653589793) is a local minimum.
The point (-1.5707963267948966, -3.141592653589793) is a local minimum.
The point (1.5707963267948966, -3.141592653589793) is a local minimum.
The point (4.71238898038469, -3.141592653589793) is a local minimum.
The point (7.853981633974483, -3.141592653589793) is a local minimum.
The point (-7.853981633974483, 0.0) is a local minimum.
The point (-4.71238898038469, 0.0) is a local minimum.
The point (-1.5707963267948966, 0.0) is a local minimum.
The point (1.5707963267948966, 0.0) is a local minimum.
The point (4.71238898038469, 0.0) is a local minimum.
The point (7.853981633974483, 0.0) is a local minimum.
The point (-7.853981633974483, 3.141592653589793) is a local minimum.
The point (-4.71238898038469, 3.141592653589793) is a local minimum.
The point (-1.5707963267948966, 3.141592653589793) is a local minimum.
The point (1.5707963267948966, 3.141592653589793) is a local minimum.
The point (4.71238898038469, 3.141592653589793) is a local minimum.
The point (7.853981633974483, 3.141592653589793) is a local minimum.
The point (-7.853981633974483, 6.283185307179586) is a local minimum.
The point (-4.71238898038469, 6.283185307179586) is a local minimum.
The point (-1.5707963267948966, 6.283185307179586) is a local minimum.
The point (1.5707963267948966, 6.283185307179586) is a local minimum.
The point (4.71238898038469, 6.283185307179586) is a local minimum.
The point (7.853981633974483, 6.283185307179586) is a local minimum.
```


Given Points
6
4
2
0
−2
−4
−6
−8
−6
−4
−2
0
2
4
6
8



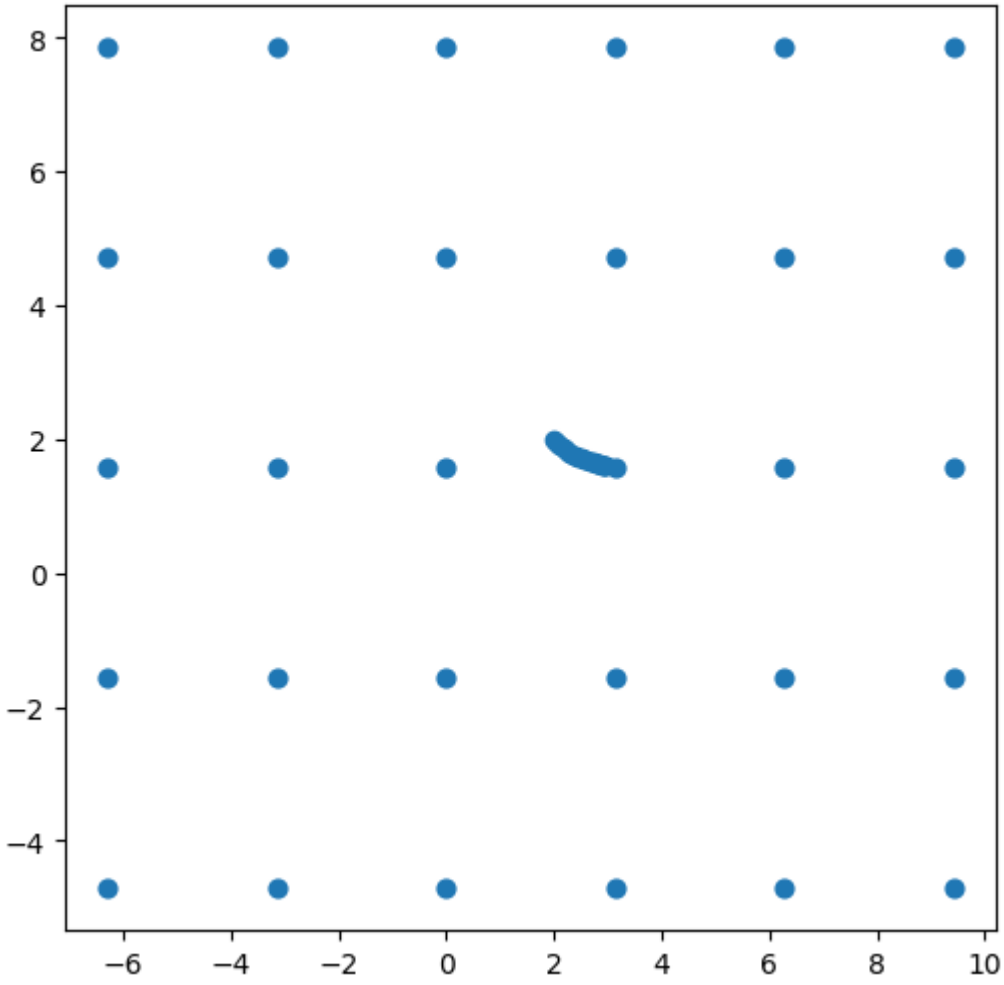
1.3 Part-B
Implementing Gradient Ascent and Visualizing Local Maximum Points

This part of the assignment involved modifying an exisiting Python script. The exisiting script plotted the local minimum points for a given cost function. It also plotted the gradient descent. The objecting was to modify the script to show the local maximum points, and the gradient ascent.

1.3.1 Python Code

```
import matplotlib as mpl
import numpy as np
import matplotlib.pyplot as plt
from numpy import linalg as LA
import math
from math import cos, sin
pi = math.pi
def grad(v):
    return np.array([cos(v[0])*sin(-v[0]),sin(v[1])*cos(v[1])])
def gradient_ascent(gradient, start, learn_rate, n_iter=25, tolerance=1e-3):

    ascent = np.copy(start)
    for _ in range(n_iter):
        diff = learn_rate * gradient(start)
        if np.all(np.abs(diff) <= tolerance):
            break
        start += diff
        ascent = np.vstack((ascent, start))

    return ascent
# C(x,y) = 0.5*(np.cos(x)**2 + np.sin(y)**2)

ascent = gradient_ascent(gradient=grad,start=np.array([2.0,2.0]), learn_rate=0.1)
xx_new, yy_new = np.meshgrid(np.arange(-5*pi/2 + pi/2, 7*pi/2, pi), np.arange(-2*pi + pi/2,
fig = plt.figure(num=1,figsize=[6, 6],edgecolor='red')
plt.scatter(xx_new, yy_new,s=40, cmap='winter')
plt.plot(ascent[:,0],ascent[:,1],'o')
plt.show(block=False)
```

1.3.2 Python Output



2 Problem-2
Neural Network Weight Visualization

2.1 Background

The objective was to implement a three-level neural network and visualize the weights associated with the network. The goal was to verify the successful execution of the provided code on different platforms, including our personal computers and the Borah server.

2.2 Part-A
Local Computer Execution

2.2.1 Python Output

```
<__main__.MyClass object at 0x00000200171D8210>
type(data) = <class '__main__.MyClass'>
type(data_dict = <class 'dict'>
list(dict_data) = ['w2', 'b2', 'w3', 'b3']



Plot the W2 weights

The weights into h0 have image

-

The weights into h1 have image


The weights into h2 have image


The weights into h3 have image


The weights into h4 have image


The weights into h5 have image


The weights into h6 have image


The weights into h7 have image


The weights into h8 have image


The weights into h9 have image
```

The weights into h10 have image

The weights into h11 have image

The weights into h12 have image

The weights into h13 have image

The weights into h14 have image

The weights into h15 have image

The weights into h16 have image

The weights into h17 have image

The weights into h18 have image

The weights into h19 have image

The weights into h20 have image

The weights into h21 have image

The weights into h22 have image

```
The weights into h23 have image


The weights into h24 have image


The weights into h25 have image


The weights into h26 have image


The weights into h27 have image


The weights into h28 have image


The weights into h29 have image


Plot the W3 weights.
The weights into output 0 are



The weights into output 1 are



The weights into output 2 are



The weights into output 3 are
```

```
The weights into output 4 are
```

```
The weights into output 5 are
```

```
The weights into output 6 are
```

```
The weights into output 7 are
```

```
The weights into output 8 are
```

```
The weights into output 9 are
```

2.3 Part-A Borah Execution

2.3.1 Observations

The python output from the Borah terminal did not include plot images.

2.3.2 Python Output

```
<__main__.MyClass object at 0x2aaab21eb790>
type(data) = <class '__main__.MyClass'>
type(data_dict = <class 'dict'>
list(dict_data) = ['w2', 'b2', 'w3', 'b3']
Plot the W2 weights
The weights into h0 have image
The weights into h1 have image
The weights into h2 have image
The weights into h3 have image
The weights into h4 have image
The weights into h5 have image
The weights into h6 have image
The weights into h7 have image
The weights into h8 have image
The weights into h9 have image
The weights into h10 have image
The weights into h11 have image
The weights into h12 have image
The weights into h13 have image
The weights into h14 have image
The weights into h15 have image
The weights into h16 have image
The weights into h17 have image
The weights into h18 have image
The weights into h19 have image
The weights into h20 have image
The weights into h21 have image
The weights into h22 have image
The weights into h23 have image
The weights into h24 have image
The weights into h25 have image
The weights into h26 have image
The weights into h27 have image
The weights into h28 have image
The weights into h29 have image
Plot the W3 weights.
The weights into output 0 are
The weights into output 1 are
The weights into output 2 are
The weights into output 3 are
The weights into output 4 are
The weights into output 5 are
The weights into output 6 are
The weights into output 7 are
```

```
The weights into output 8 are
The weights into output 9 are
```