Student name: Duong Doan Tung
Student ID: 21010294
Course: Applied Mathematics for Artificial Intelligence

Google colab notebook contains all the Python solution code for this assignment.
Hyper link to google colab notebook: Assignment 3.ipynb
Problem 1:

Problem 1:

Consider the following scalar-valued function

$$f(x, y, z) = x^2 y + \sin(z + 6y),$$

where $x, y, z \in \mathbb{R}$.

(A) Compute partial derivatives with respect to x, y, and z.

(B) We can consider f to take a vector $\boldsymbol{\theta} \in \mathbb{R}^3$ as input where $\boldsymbol{\theta} = [x, y, z]^T$. Show the gradient $\nabla_{\boldsymbol{\theta}} f$ as a vector and evaluate it at $\boldsymbol{\theta} = [3, \frac{\pi}{2}, 0]^T$.

(A)

$$f'_x = 2xy$$

$$f'_y = x^2 + 6cos(z + 6y)$$

$$f'_z = cos(z + 6y)$$

(B)

$$\nabla_\theta f = \left[2xy, x^2 + 6cos(z + 6y), cos(z + 6y)\ \right]^T$$

Where $\theta = \left[3, \frac{\pi}{2}, 0\right]^T \Rightarrow \begin{cases} x = 3 \\ y = \pi/2 \\ z = 0 \end{cases}$

Therefore, $\nabla_\theta f = [3\pi, 3, -1]^T$

Code implementation: Problem 1

Problem 2:

The purpose of this problem is to demonstrate Clairaut's Theorem, which states that in general, the order in which you partial differentiate does not matter. Consider the following scalar-valued function,

$$f(x, y, z) = x \sin(xy),$$

where $x, y \in \mathbb{R}$.

(A) Compute $\frac{\partial}{\partial x}\frac{\partial}{\partial y} f(x, y)$. This means we first compute the partial derivative of f with respect to y, then compute the partial derivative of the resulting function with respect to x. This is sometimes denoted $\partial_{xy} f$.

(B) Compute $\frac{\partial}{\partial y}\frac{\partial}{\partial x} f(x, y)$.

You should have gotten the same answer for parts (A) and (B), which demonstrates Clairaut's Theorem. This holds more generally for n variables as well, that is, if $f(x_1, x_2, \ldots, x_n)$ is a function of n variables. This result can be useful when differentiating functions, since it's possible that it's much more convenient computationally-wise to differentiate in an order that you choose.

(A)

$$\frac{\partial}{\partial y} f = x^2 cos(xy) => \frac{\partial}{\partial x}\frac{\partial}{\partial y} f = 2xcos(xy) - x^2 ysin(xy)$$

(B)

$$\frac{\partial}{\partial x} f = sin(xy) + xycos(xy) => \frac{\partial}{\partial y}\frac{\partial\ f}{\partial x} = xcos(xy) + xcos(xy) - x^2 ysin(xy)$$

=> Clairaut's Theorem is true for this function.
Code implementation: Problem 2

Problem 3:

In gradient descent, we attempt to minimize some function $f(\boldsymbol{x})$ by altering parameters $\boldsymbol{x} \in \mathbb{R}^n$ according to the following formula:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \lambda(\nabla_{\boldsymbol{x}} f(\boldsymbol{x}_t))^T$$

for some small $\lambda \geq 0$ known as the *learning rate* or *step size*. We adjust $\boldsymbol{x}$ so as to move in a direction proportional to the negative gradient. We will discuss gradient descent in more detail later in the course.

Consider the simple function $f(\boldsymbol{x}) = \boldsymbol{x}^T A\boldsymbol{x}$ for a constant matrix $A \in \mathbb{R}^{n \times n}$.

(A) Implement a function `f(A, x)` that takes as input an $n \times n$ numpy array `A` and a 1D array `x` of length n and returns the output for the above definition.

(B) Implement a function `grad_f(A, x)` that takes the same two arguments as above but returns $\nabla_{\boldsymbol{x}} f(\boldsymbol{x})$ evaluated at `x`.

(C) Now implement a third and final function `grad_descent(A, x, lr, num_iters)` that takes the additional arguments `lr`, representing the learning rate λ above, and `num_iters` indicating the total number of iterations of gradient descent to perform. The function should log, via either printing or plotting, the values of $\boldsymbol{x}_t$ and $f(\boldsymbol{x}_t)$ at each iteration of gradient descent.

(D) We will perform gradient descent on f with $A = \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}$. Set each element of the initial $\boldsymbol{x}_0$ to any value with magnitude between 10 and 100 of your choosing. Run gradient descent for 50 iterations with learning rates of 1, 0.25, 0.1, and 0.01. What do you notice? Does $\boldsymbol{x}_t$ always converge to the same value? Does our gradient descent algorithm work every time?

```python
#Problem 3
print()

# a)
def f(x):
    return x.T @ A@x
# b)
def grad_f(A,x):
    return 2*A@x
# c)
def grad_descent(A,x,l,num_inters):
    for i in range(num_inters):
        x = x - l*grad_f(A,x)
        print("\rLearning rate: ", l, "Iteration: ", i+1, "x: ", x,end = "")
    return x

# d)
A = np.array([[1,0],[0,4]])
X = np.random.randint(10,100,2)
L = [1,0.25,0.1,0.01]
for l in L:
    grad_descent(A,X,l,50)
    print('\n-----------------------------------')
```

✓ 0.2s

```
Learning rate:  1 Iteration:  50 x:  [        72 1861597058]0]
-----------------------------------
Learning rate:  0.25 Iteration:  50 x:  [6.39488462e-14 3.40000000e+01]1]
-----------------------------------
Learning rate:  0.1 Iteration:  50 x:  [1.02761834e-03 3.82805968e-34]
-----------------------------------
Learning rate:  0.01 Iteration:  50 x:  [26.22021697  0.52586018]
-----------------------------------
```

Jupyter Server: Local

Code implementation: Problem 3

Problem 4:

Consider the following vector function from $\mathbb{R}^3$ to $\mathbb{R}^3$:

$$\boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} \sin(x_1 x_2 x_3) \\ \cos(x_2 + x_3) \\ \exp\{-\frac{1}{2}(x_3^2)\} \end{bmatrix}$$

(A) What is the Jacobian matrix of $\boldsymbol{f}(\boldsymbol{x})$?

(B) Write the determinant of this Jacobian matrix as a function of $\boldsymbol{x}$.

(C) Is the Jacobian a full rank matrix for all of $\boldsymbol{x} \in \mathbb{R}^3$? Explain your reasoning.

A)

$$J = \begin{bmatrix} \frac{\partial f(x)}{\partial x_1} & \frac{\partial f(x)}{\partial x_2} & \frac{\partial f(x)}{\partial x_3} \end{bmatrix} = \begin{bmatrix} x_2 x_3 \cos(x_1 x_2 x_3) & x_1 x_3 \cos(x_1 x_2 x_3) & x_1 x_2 \cos(x_1 x_2 x_3) \\ 0 & -\sin(x_2 + x_3) & -\sin(x_2 + x_3) \\ 0 & 0 & -x_3 e^{\frac{-1}{2}({x_3}^2)} \end{bmatrix}$$

B) determinant of the Jacobian:

$$|J| = (x_2 x_3 \cos(x_1 x_2 x_3)) * (-\sin(x_2 + x_3)) * (-x_3 e^{\frac{-1}{2}({x_3}^2)})$$

$$= x_2 {x_3}^2 (e^{\frac{-1}{2}({x_3}^2)}) \cos(x_1 x_2 x_3) \sin(x_2 + x_3)$$

C) This Jacobian is full rank for all of $x \in R^3$ because the determinant is not equal to 0 for all $x \in R^3$.

Code implementation: Problem 4

Problem 5:

Compute the gradients for the following expressions. (You can use identities, but show your work.)

(A) $\nabla_x \operatorname{trace}(\boldsymbol{x}\boldsymbol{x}^T + \sigma^2 \boldsymbol{I})$ Assume $\boldsymbol{x} \in \mathbb{R}^n$ and $\sigma \in \mathbb{R}$.

(B) $\nabla_x \frac{1}{2}(\boldsymbol{x}-\mu)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{x}-\mu)$ Assume $\boldsymbol{x}, \mu \in \mathbb{R}^n$ and invertible symmetric $\boldsymbol{\Sigma} \in \mathbb{R}^{n \times n}$.

(C) $\nabla_x (\boldsymbol{c}-\boldsymbol{A}\boldsymbol{x})^T (\boldsymbol{c}-\boldsymbol{A}\boldsymbol{x})$ Assume $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{c} \in \mathbb{R}^m$ and $\boldsymbol{A} \in \mathbb{R}^{m \times n}$.

(D) $\nabla_x (\boldsymbol{c}+\boldsymbol{A}\boldsymbol{x})^T (\boldsymbol{c}-\boldsymbol{B}\boldsymbol{x})$ Assume $\boldsymbol{x} \in \mathbb{R}^n$, $\boldsymbol{c} \in \mathbb{R}^m$ and $\boldsymbol{A}, \boldsymbol{B} \in \mathbb{R}^{m \times n}$.

a)

$$\operatorname{trace}\left(\mathrm{x}\mathrm{x}^T + \sigma^2 I\right) = \operatorname{trace}\left(\mathrm{x}\mathrm{x}^T\right) + n * \sigma^2$$

$$\mathrm{x}\mathrm{x}^T = \begin{bmatrix} {x_1}^2 & \cdots & x_1 x_n \\ \vdots & \ddots & \vdots \\ x_n x_1 & \cdots & {x_n}^2 \end{bmatrix}$$

$$\textit{trace}(\mathrm{x}\mathrm{x}^T) = {x_1}^2 + {x_2}^2 + \cdots + {x_n}^2$$

$$f(x) = \operatorname{trace}\left(\mathrm{x}\mathrm{x}^T + \sigma^2 I\right) = {x_1}^2 + {x_2}^2 + \cdots + {x_n}^2 + n * \sigma^2$$

$$\nabla_x \operatorname{trace}\left(\mathrm{x}\mathrm{x}^T + \sigma^2 I\right) = \begin{bmatrix} \frac{\partial f(x)}{x_1} & \frac{\partial f(x)}{x_2} & \frac{\partial f(x)}{x_3} & \cdots & \frac{\partial f(x)}{x_n} \end{bmatrix}$$

$$= 2 \begin{bmatrix} x_1 & x_2 & x_3 & \cdots & x_n \end{bmatrix} = 2x^T$$

b) $\nabla_x \frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)$

We have: $\Sigma^{-1} = \begin{bmatrix} {\Sigma_1}^{-1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & {\Sigma_n}^{-1} \end{bmatrix}$

$$x - \mu = \begin{bmatrix} x_1 - \mu_1 \\ x_2 - \mu_2 \\ x_3 - \mu_3 \\ \vdots \\ x_n - \mu_n \end{bmatrix}$$

Transpose: $(x-\mu)^T = \begin{bmatrix} x_1 - \mu_1 & x_2 - \mu_2 & x_3 - \mu_3 & \cdots & x_n - \mu_n \end{bmatrix}$

$$= \begin{bmatrix} {\Sigma_1}^{-1}(x_1-\mu_1) & {\Sigma_2}^{-1}(x_2-\mu_2) & {\Sigma_3}^{-1}(x_3-\mu_3) & \cdots & {\Sigma_n}^{-1}(x_n-\mu_n) \end{bmatrix} \begin{bmatrix} x_1-\mu_1 \\ x_2-\mu_2 \\ x_3-\mu_3 \\ \vdots \\ x_n-\mu_n \end{bmatrix}$$

$$= \left[{\Sigma_1}^{-1} \left(x_1-\mu_1\right)^2 + {\Sigma_2}^{-1}(x_2-\mu_2)^2 + \cdots + {\Sigma_n}^{-1}(x_n-\mu_n)^2 \right]$$

$$\nabla_x \frac{1}{2} (x-\mu)^T \Sigma^{-1} (x-\mu) = \begin{bmatrix} \frac{x_1-\mu_1}{\Sigma_1} & \frac{x_2-\mu_2}{\Sigma_2} & \frac{x_3-\mu_3}{\Sigma_3} & \cdots & \frac{x_n-\mu_n}{\Sigma_n} \end{bmatrix}$$

c) $\nabla_x (c-Ax)^T (c-Ax)$

$$Ax = \begin{bmatrix} A_{11}x_1 + A_{12}x_2 + \cdots + A_{1n}x_n \\ \vdots \\ A_{m1}x_1 + A_{m2}x_2 + \cdots + A_{\mathrm{mn}}x_n \end{bmatrix}$$

$$c-Ax = \begin{bmatrix} c_1 - A_{11}x_1 - A_{12}x_2 - \cdots - A_{1n}x_n \\ \vdots \\ c_m - A_{m1}x_1 - A_{m2}x_2 - \cdots - A_{\mathrm{mn}}x_n \end{bmatrix}$$

$$(c-Ax)^T (c-Ax)$$

$$= \left[(c_1 - A_{11}x_1 - A_{12}x_2 - \cdots - A_{1n}x_n)^2 + \cdots + (c_m - A_{m1}x_1 - A_{m2}x_2 - \cdots - A_{\mathrm{mn}}x_n)^2 \right]$$

$$= \left[(A_{11}x_1 + A_{12}x_2 + \cdots + A_{1n}x_n - c_1)^2 + \cdots + (A_{m1}x_1 + A_{m2}x_2 + \cdots + A_{\mathrm{mn}}x_n - c_m)^2 \right]$$

$$\nabla_x (c-Ax)^T (c-Ax)$$

$$= \begin{bmatrix} 2A_{11}(A_{11}x_1 + A_{12}x_2 + \cdots + A_{1n}x_n - c_1) + \cdots + 2A_{m1}(A_{m1}x_1 + A_{m2}x_2 + \cdots + A_{\mathrm{mn}}x_n - c_m) \\ \vdots \\ 2A_{1n}(A_{11}x_1 + A_{12}x_2 + \cdots + A_{1n}x_n - c_1) + \cdots + 2A_{\mathrm{mn}}(A_{m1}x_1 + A_{m2}x_2 + \cdots + A_{\mathrm{mn}}x_n - c_m) \end{bmatrix}$$

$$= 2 \begin{bmatrix} A_{11} & \cdots & A_{m1} \\ \vdots & \ddots & \vdots \\ A_{1n} & \cdots & A_{\mathrm{mn}} \end{bmatrix} \begin{bmatrix} A_{11}x_1 + A_{12}x_2 + \cdots + A_{1n}x_n - c_1 \\ \vdots \\ A_{m1}x_1 + A_{m2}x_2 + \cdots + A_{\mathrm{mn}}x_n - c_m \end{bmatrix}$$

d)

$$c + Ax = \begin{bmatrix} c_1 + A_{11}x_1 + A_{12}x_2 + \cdots + A_{1n}x_n \\ \vdots \\ c_m + A_{m1}x_1 + A_{m2}x_2 + \cdots + A_{\mathrm{mn}}x_n \end{bmatrix}$$

$$c - Bx = \begin{bmatrix} c_1 - B_{11}x_1 - B_{12}x_2 - \cdots - B_{1n}x_n \\ \vdots \\ c_m - B_{m1}x_1 - B_{m2}x_2 - \cdots - B_{\mathrm{mn}}x_n \end{bmatrix}$$

$$(c + Ax)^T (c - Bx)$$

$$= [(c_1 + A_{11}x_1 + A_{12}x_2 + \cdots + A_{1n}x_n)(c_1 - B_{11}x_1 - B_{12}x_2 - \cdots - B_{1n}x_n) + \cdots + (c_m + A_{m1}x_1 + A_{m2}x_2 + \cdot$$

$$\nabla_x (c + Ax)^T (c - Bx)$$

$$= \begin{bmatrix} A_{11}(c_1 - (\sum_{i=1}^{n} B_{1i}x_i)) - B_{11}(c_1 + (\sum_{i=1}^{n} A_{1i}x_i)) + \cdots + A_{m1}(c_m - (\sum_{i=1}^{n} B_{\mathrm{mi}}x_i)) - B_{m1}(c_m + (\sum_{i=1}^{n} A_{\mathrm{r}} \\ \vdots \\ A_{1n}(c_1 - (\sum_{i=1}^{n} B_{1i}x_i)) - B_{1n}(c_1 + (\sum_{i=1}^{n} A_{1i}x_i)) + \cdots + A_{\mathrm{mn}}(c_m - (\sum_{i=1}^{n} B_{\mathrm{mi}}x_i)) - B_{\mathrm{mn}}(c_m + (\sum_{i=1}^{n} A_{\mathrm{r}} \end{bmatrix}$$

$$= \begin{bmatrix} A_{11} & -B_{11} & \cdots & A_{m1} & -B_{m1} \\ \vdots & & \ddots & & \vdots \\ A_{1n} & -B_{1n} & \cdots & A_{\mathrm{mn}} & -B_{\mathrm{mn}} \end{bmatrix} \begin{bmatrix} c_1 - B_{11}x_1 - B_{12}x_2 - \cdots - B_{1n}x_n \\ c_1 + A_{11}x_1 + A_{12}x_2 + \cdots + A_{1n}x_n \\ \vdots \\ c_m - B_{m1}x_1 - B_{m2}x_2 - \cdots - B_{\mathrm{mn}}x_n \\ c_m + A_{m1}x_1 + A_{m2}x_2 + \cdots + A_{\mathrm{mn}}x_n \end{bmatrix}$$

Code implementation: Problem 5

Problem 6:

(A) The sigmoid function $f : \mathbb{R} \to \mathbb{R}$ (also called the *logistic function*) is defined to be:

$$f(z) = \frac{1}{1 + e^{-z}} \tag{1}$$

Compute the derivative of the sigmoid function, i.e., $f'(z)$. And verify that $f'(z) = f(z)\,(1 - f(z))$

(B) The cost function of a very popular machine learning model logistic regression has the following form:

$$c(\boldsymbol{\theta}, \boldsymbol{x}, y) = -y \log(\frac{1}{1 + e^{-\theta^\top x}}) - (1 - y) \log(1 - \frac{1}{1 + e^{-\theta^\top x}}) \tag{2}$$

with $\boldsymbol{\theta} \in \mathbb{R}^d, \boldsymbol{x} \in \mathbb{R}^d, y \in \mathbb{R}$. Compute the partial derivative with regards to θ, i.e. $\frac{\partial c(\theta, x, y)}{\partial \theta}$. And verify that $\frac{\partial c(\theta, x, y)}{\partial \theta} = \left(f(\boldsymbol{\theta}^\top \boldsymbol{x}) - y\right) \boldsymbol{x}^\top$.

a)

$$\frac{\partial f(z)}{\partial z} = \frac{e^{-z}}{(1 + e^{-z})^2} = \frac{1}{1 + e^{-z}} \left(1 - \frac{1}{1 + e^{-z}}\right) = f(z)(1 - f\,(z))$$

b)

..To be added..

Code implementation: Problem 6