AI VIET NAM – COURSE 2022

Calculus - Exercise 2

(Derivative and its Applications)

Ngày 8 tháng 6 năm 2023

1. **Trình bày chi tiết đạo hàm các activation sau và thực hiện code bằng python ((a) Tính đạo hàm và cài đặt; (b) dùng matplotlib để plot hàm số và hàm đạo hàm; chỉ dùng list)**

   1.1 **Exponential** $f(x) = e^x$

   1.2 **PReLU**: $\alpha = 0.25$, $f(x) = \begin{cases} \alpha x, & \text{if } x \leq 0. \\ x, & \text{if x} > 0. \end{cases}$

   1.3 **ELU**: $\alpha = 0.25$, $f(x) = \begin{cases} \alpha(e^x - 1), & \text{if } x \leq 0. \\ x, & \text{if } x > 0. \end{cases}$

   1.4 **Softplus**: $f(x) = \log(1 + e^x)$

   1.5 **Softsign**: $f(x) = \dfrac{x}{|x| + 1}$

   - **NOTE: Các bạn thực hiện theo các yêu cầu sau**
     - Trình bày chi tiết cách đạo hàm (Có thể viết bằng latex sau đó gửi file, hoặc viết bằng markdown trên google colab)
     - Sau đó các bạn viết code cho function $f(x)$ và $f'(x)$, tạo ra một list các điểm x. Sử dụng hàm đã được viết sẵn **plot_function_and_derivative(x, function, function_sau_khi_đạo_hàm)** để thu được hình tương tự như hình 1. Các bạn có thể sử dụng code mẫu tại đây LINK.

- **Example**
  - **Trình bày đạo hàm Sigmoid** $f(x) = \dfrac{1}{1+e^{-x}}$
    - $f'(x) = \dfrac{-1}{(1+e^{-x})^2}(-e^{-x}) = \dfrac{e^{-x}}{(1+e^{-x})^2} = \dfrac{1+e^{-x}-1}{(1+e^{-x})^2} = \dfrac{1+e^{-x}}{(1+e^{-x})^2} - \dfrac{1}{(1+e^{-x})^2} = \dfrac{1}{1+e^{-x}} - \dfrac{1}{(1+e^{-x})^2} = \dfrac{1}{1+e^{-x}}(1 - \dfrac{1}{1+e^{-x}}) = f(x)(1-f(x))$
  - **Thực hiện code và vẽ**:
    - code $f(x)$:

```
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
```

    - code list of $f(x)$:

```
def sigmoid_list(x_data):
    fx = [sigmoid(x) for x in x_data]
    return fx
```

    - code $f'(x)$:

```
def sigmoid_derivative(x):
    y = sigmoid(x)
    return y*(1-y)
```

    - code list of $f'(x)$:

```
def d_sigmoid_list(x_data):
    dfx = [sigmoid_derivative(x) for x in x_data]
    return dfx
```

    - Tạo một array list các giá trị x trong range [-10, 10] mỗi step là 1 đơn vị (Cái này tùy các bạn lựa chọn):

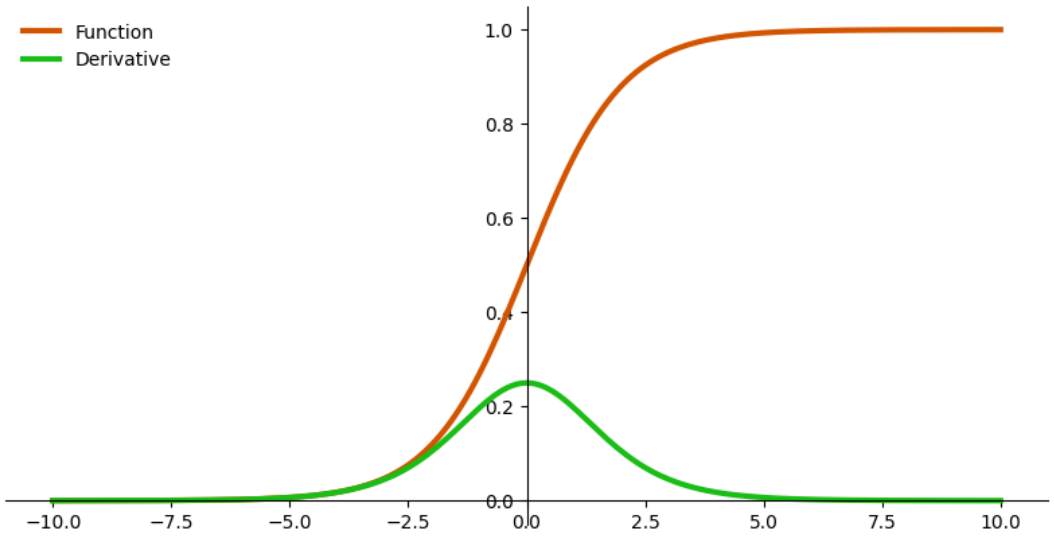
```
x_data = list(range(-100, 100+1, 1))
x_data = [x/10 for x in x_data]
```

    - Thực hiện vẽ với hàm cho trước **plot_function_and_derivative** truyền vào danh sách x, danh sách kết quả $f(x)$ và danh sách kết quả $f'(x)$:

```
plot_function_and_derivative(x_data, fx, dfx)
```

```
import matplotlib.pyplot as plt
import math

def plot_function_and_derivative(x, f_x, df_x):
    # setting the axes at the centre
    fig, ax = plt.subplots(figsize=(10, 5))
    ax.spines['left'].set_position('center')
    ax.spines['bottom'].set_position('zero')
    ax.spines['right'].set_color('none')
    ax.spines['top'].set_color('none')
    ax.xaxis.set_ticks_position('bottom')
    ax.yaxis.set_ticks_position('left')

    # plot the function
    ax.plot(x, f_x, color="#d35400", linewidth=3, label='Function')
    ax.plot(x, df_x, color="#1abd15", linewidth=3, label='Derivative')

    ax.legend(loc="upper left", frameon=False)
```

```
    plt.show()

# Sigmoid Example
def sigmoid(x):
    return 1 / (1 + math.exp(-x))

def sigmoid_list(x_data):
    fx = [sigmoid(x) for x in x_data]
    return fx

def sigmoid_derivative(x):
    y = sigmoid(x)
    return y*(1-y)

def d_sigmoid_list(x_data):
    dfx = [sigmoid_derivative(x) for x in x_data]
    return dfx

x_data = list(range(-100, 100+1, 1))
x_data = [x/10 for x in x_data]
fx = sigmoid_list(x_data)
dfx = d_sigmoid_list(x_data)
plot_function_and_derivative(x_data, fx, dfx)
```



Hình 1: Ví dụ vẽ hình sigmoid và đạo hàm sigmoid

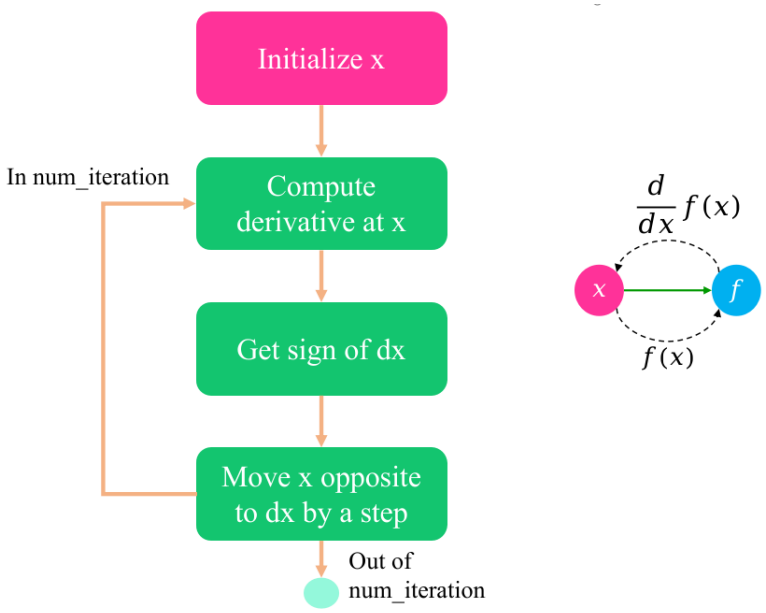
```
Multiple choices:****************************************

1. What are the outputs of the following bunch of codes by using sigmoid
    activation:

x_list = list([-1, 0, 1])
x_sigmoid_list = sigmoid_list(x_list)
print([round(num, 2) for num in x_sigmoid_list])

a) [0.27, 0.5, 0.73]

b) [0.35, 0.5, 0.75]

c) [0.45, 0.5, 0.71]
```

```
d) [0.27, 0.5, 0.77]


2. What are the outputs of the following bunch of codes by using sigmoid
    activation:

x_list = list([-1, 0, 1])
x_gradient_list = d_sigmoid_list(sigmoid_list(x_list))
print([round(num, 2) for num in x_gradient_list])

a) [0.25, 0.24, 0.22]

b) [0.25, 0.25, 0.25]

c) [0.25, 0.24, 0.28]

d) [[0.25, 0.74, 0.22]]

3. What are the outputs of the following bunch of codes by using  Exponential
    activation:

x_list = list([-1, 0, 1])
x_exponential_list = exponential_list(x_list)
print([round(num, 2) for num in x_exponential_list])

a) [0.47, 1.0, 2.72]

b) [0.37, 1.0, 2.72]

c) [0.37, 2.0, 2.72]

d) [0.37, 1.0, 3.72]


4. What are the outputs of the following bunch of codes by using  Exponential
    activation:

x_list = list([-1, 0, 1])
x_gradient_list = d_exponential_list(exponential_list(x_list))
print([round(num, 2) for num in x_gradient_list])

a) [2.44, 2.72, 15.15]

b) [3.44, 2.72, 15.15]

c) [1.44, 2.72, 15.15]

d) [1.44, 2.72, 20.15]


5. What are the outputs of the following bunch of codes by using  PReLU
    activation:

x_list = list([-1, 0, 1])
x_prelu_list = prelu_list(x_list)
print([round(num, 2) for num in x_prelu_list])

a) [-1.25, 0.0, 1.0]
```

```
b) [-0.25, 1.0, 1.0]

c) [-0.25, 0.0, 2.0]

d) [-0.25, 0.0, 1.0]

6. What are the outputs of the following bunch of codes by using  PReLU
    activation:

x_list = list([-1, 0, 1])
x_gradient_list = d_prelu_list(prelu_list(x_list))
print([round(num, 2) for num in x_gradient_list])

a) [0.25, 1.0, 1.0]

b) [1.25, 1.0, 1.0]

c) [0.25, 2.0, 1.0]

d) [0.25, 1.0, 3.0]

7. What are the outputs of the following bunch of codes by using  ELU activation:

x_list = list([-1, 0, 1])
x_elu_list = elu_list(x_list)
print([round(num, 2) for num in x_elu_list])

a) [0.16, 0.0, 1.0]

b) [-0.16, 0.0, -1.0]

c) [0.16, 1.0, 1.0]

d) [-0.16, 0.0, 1.0]


8. What are the outputs of the following bunch of codes by using  ELU activation:

x_list = list([-1, 0, 1])
x_gradient_list = d_elu_list(elu_list(x_list))
print([round(num, 2) for num in x_gradient_list])

a) [0.21, 1.0, 1.0]

b) [0.21, -1.0, 1.0]

c) [0.21, 1.0, -1.0]

d) [-0.21, 1.0, 1.0]


9. What are the outputs of the following bunch of codes by using  Softplus
    activation:

x_list = list([-1, 0, 1])
x_softplus_list = softplus_list(x_list)
print([round(num, 2) for num in x_gradi

a) [1.31, 0.69, 1.31]
```

```
b) [0.31, 0.69, 1.31]

c) [0.31, 0.69, 2.31]

d) [0.31, 1.69, 1.31]

10. What are the outputs of the following bunch of codes by using  Softplus
    activation:

x_list = list([-1, 0, 1])
x_gradient_list = d_softplus_list(softplus_list(x_list))
print([round(num, 2) for num in x_gradient_list])

a) [0.58, 0.67, 0.79]

b) [1.58, 0.67, 0.79]

c) [0.58, 1.67, 0.79]

d) [0.58, 0.67, 1.79]


11. What are the outputs of the following bunch of codes by using  Softsign
    activation:

x_list = list([-1, 0, 1])
x_softsign_list = softsign_list(x_list)
print([round(num, 2) for num in x_softsign_list])

a) [-0.5, 0.0, 0.5]

b) [-0.5, 2.0, 0.5]

c) [-0.5, 0.0, 3.5]

d) [-0.5, 1.0, 0.5]

12. What are the outputs of the following bunch of codes by using  Softsign
    activation:

x_list = list([-1, 0, 1])
x_gradient_list = d_softsign_list(softsign_list(x_list))
print([round(num, 2) for num in x_gradient_list])

a) [1.44, 1.0, 0.44]

b) [0.44, 1.0, 0.44]

c) [0.44, 1.0, 2.44]

d) [0.44, 2.0, 0.44]
```

2. **Simple Optimization: Thực hiện thuật toán optimization đơn giản sau để tìm vị trí tại x mà $f(x)$ là minimum**



Hình 2: Simple Optimization Algorithm

NOTE: Các bạn thực hiện theo các yêu cầu sau

2.1 Viết function *find_ minimum(f, x, num_ iteration, step)* và dựa theo thuật toán ở hình 2 để tìm xấp xỉ x mà $f(x)$ (ví dụ $f(x) = 3x^4 - 4x^2 - 6x - 3$) là minimum.

- **Input**: Nhận 4 input
  - **f**: function $f(x)$
  - **x**: giá trị khởi tạo x đầu tiên
  - **num_iteration**: Số lần lặp thuật toán để tìm x
  - **step**: độ lớn để cho một lần cập nhật x (độ lớn quãng đường đi ngược hướng với giá trị đạo hàm tại x (dx))
- **Output** giá trị xấp xỉ của x tại đó giá trị hàm $f(x)$ (hàm được truyền vào từ input) là minimum
- **Các bạn thực hiện theo các step sau**:
  - **Step1**: Thực hiện vòng lặp với num_iteration số lần lặp
  - **Step2**: Trong mỗi lần lặp tìm giá trị đạo hàm (dx) tại x của hàm $f(x)$ bằng phương pháp đạo hàm trung tâm (central difference)
  - **Step3**: Xét dấu của giá trị đạo hàm (dx) để xác định độ lớn giá trị cập nhật.
  - **Step4**: Nếu dx là số dương thì cập nhật $x = x - step$, nếu dx là số âm thì cập nhật $x = x + step$, nếu $dx = 0$ thì không thực hiện việc cập nhật
  - **Step5**: Thực hiện Step2, Step3 và Step4 để cập nhật x cho đến khi đủ num_iteration số lần lặp thì thoát loop và trả về x mới nhất

```python
# Example 2.1
import random
def f(x):
    return 3*x**4 - 4*x**2 - 6*x - 3

x = random.uniform(-10, 10)
print("initial x: ", x)
```

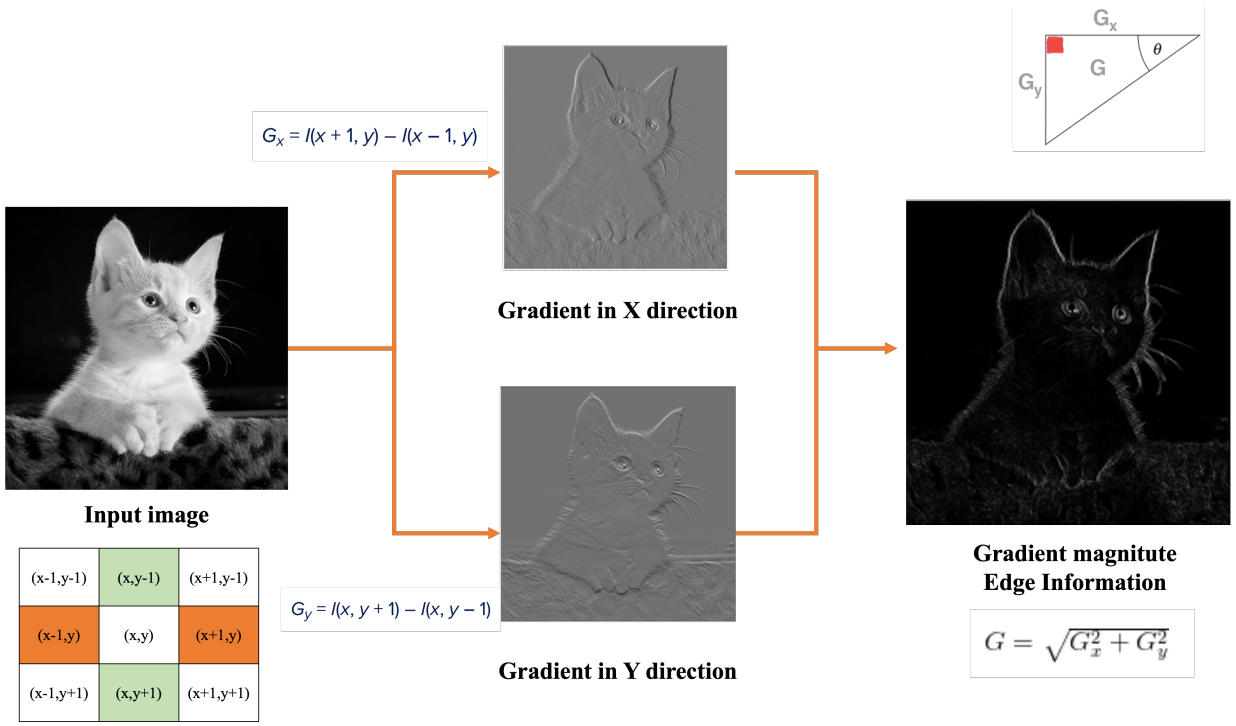
```
>> initial x:  8.033107966229196

x = find_minimum(f=f, x=x, num_iteration=100, step=0.1)
print(x)
>> 1.033107966229207
```

2.2 Multiple choice questions: Sử dụng hàm *find_minimum(f, x, num_iteration, step)* hoàn thành ở 2.1 để hoàn thành bài tập sau. Cho trước hàm $f(x) = 3x^4 - 4x^2 - 6x - 3$

```
13.  What is the result x after excuting the following bunch of codes to find
     the value x where $f(x)$ gets minumum value:
x = 3.0
x = find_minimum(f=f, x=x, num_iteration=1, step=0.1)
print(round(x,2))

a) x = 2.9

b) x = 3.9

c) x = 4.0

d) x = 4.9

14.  What is the result x after excuting the following bunch of codes to find
     the value x where $f(x)$ gets minumum value:

x = 3.0
x = find_minimum(f=f, x=x, num_iteration=5, step=0.1)
print(round(x,2))

a) x = 2.5

b) x = 3.5

c) x = 4.5

d) x = 5.5

15.  What is the result x after excuting the following bunch of codes to find
     the value x where $f(x)$ gets minumum value:

x = 3.0
x = find_minimum(f=f, x=x, num_iteration=100, step=0.2)
print(round(x,2))

a) x = 1.0

b) x = 2.0

c) x = 3.0

d) x = 4.0
```

3. **Sử dụng đạo hàm rời rạc trên ảnh để phát hiện thông tin cạnh/edge (optional), như hình 3**. Cho trước một ảnh đầu vào LINK có kích thước bất kỳ. Hãy viết chương trình giảm size ảnh xuống (400,400), sau đó hiện thức các hàm sau:

- *computeXDerivative(image))*:



Hình 3: Sử dụng đạo hàm rời rạc theo phương x và y để phát hiện thông tin cạnh trong ảnh

```python
#Caculate gradient in X-direction
def computeXDerivative(image):
  w = len(image[0])
  h = len(image)
  x_derivative = [[0]*w for _ in range(h)]

  # your code here *************

  return x_derivative
```

- *computeYDerivative(image))*:

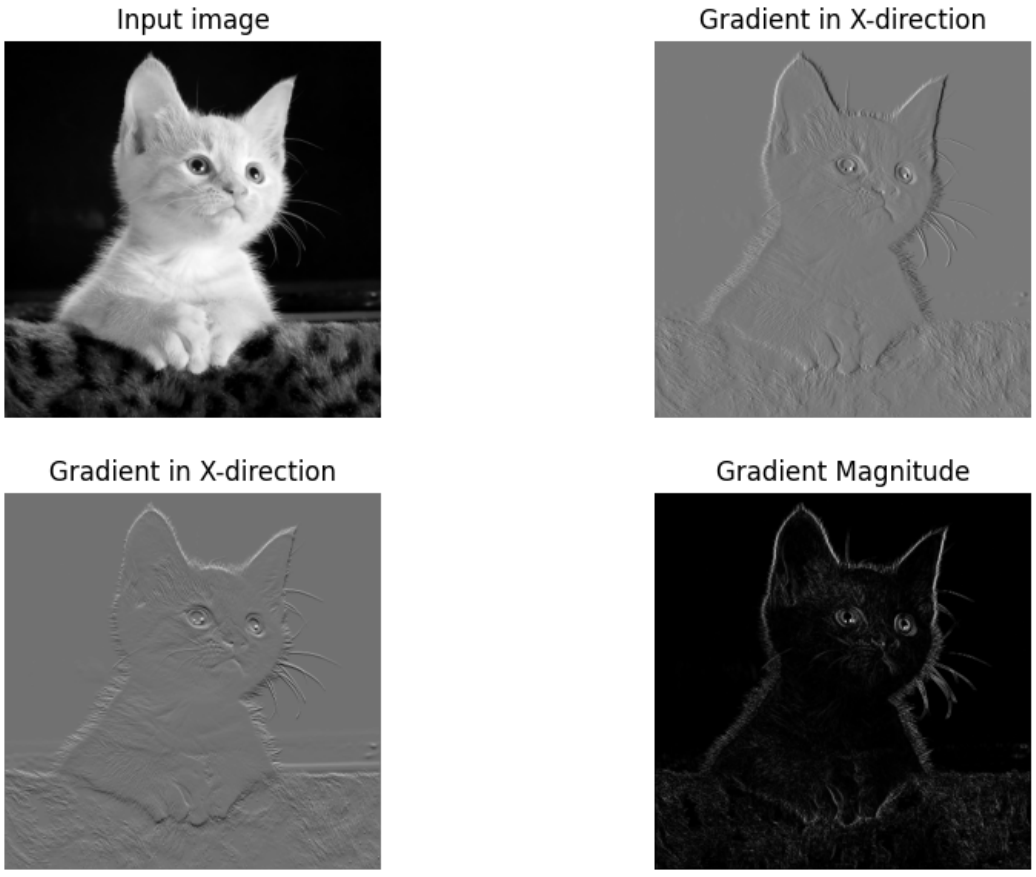
```python
#Caculate gradient in Y-direction
def computeYDerivative(image):
  w = len(image[0])
  h = len(image)
  y_derivative = [[0]*w for _ in range(h)]

   # your code here *************

  return y_derivative
```

- *computeMagniguteXY(image))*:

```python
# Calculate the gradient magnitude
def computeMagniguteXY(image):
  w = len(image[0])
  h = len(image)
  gradient_magnitude = [[0]*w for _ in range(h)]

   # your code here *************

  return gradient_magnitude
```

Sử dụng đoạn code bên dưới để visualize thông tin gradient theo phương X, Y và magnitude (như hình 4)

```
import cv2
from google.colab.patches import cv2_imshow
import math
from matplotlib import pyplot as plt


cat_image = cv2.imread('/content/cat.jpeg', 0)
cat_image = cv2.resize(cat_image, (400,400), interpolation = cv2.INTER_AREA)

cat_image_list = cat_image.tolist()
x_derivative = computeXDerivative(cat_image_list)
y_derivative = computeYDerivative(cat_image_list)
gradient_magnitude = computeMagniguteXY(cat_image_list)

# create figure
fig = plt.figure(figsize=(10, 7))

# showing image
fig.add_subplot(2, 2, 1)
plt.imshow(cat_image, cmap="gray")
plt.axis('off')
plt.title("Input image")

fig.add_subplot(2, 2, 2)
plt.imshow(x_derivative, cmap="gray")
plt.axis('off')
plt.title("Gradient in X-direction")

fig.add_subplot(2, 2, 3)
plt.imshow(y_derivative, cmap="gray")
plt.axis('off')
plt.title("Gradient in X-direction")

fig.add_subplot(2, 2, 4)
plt.imshow(gradient_magnitude, cmap="gray")
plt.axis('off')
plt.title("Gradient Magnitude")
```



Hình 4: Kết quả visualize đạo hàm the phương X, Y và Magnitude