AI VIET NAM – COURSE 2022

# Calculus 1 Exercise

Ngày 18 tháng 7 năm 2022

1. **Trình bày chi tiết đạo hàm các activation sau và thực hiện code bằng python**

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

   1.6 **LeakyReLU (OPTIONAL)**: $\alpha = 0.01, f(x) = \begin{cases} \alpha x, & \text{if } x \leq 0. \\ x, & \text{if } x > 0. \end{cases}$

   1.7 **Selu (OPTIONAL)**: $\lambda = 1.05, \alpha = 1.67, f(x) = \lambda \begin{cases} \alpha e^x - \alpha, & \text{if } x \leq 0 \\ x, & \text{if } x > 0. \end{cases}$

   1.8 **Hard Sigmoid (OPTIONAL)**: $f(x) = \begin{cases} 0, & \text{if } x < -2.5 \\ 0.2x + 0.5, & \text{if } -2.5 \leq x \leq 2.5 \\ 1 & \text{if } 2.5 < x \end{cases}$

   1.9 **GELU (OPTIONAL)**: $f(x) = 0.5x(1 + tanh(\sqrt{\dfrac{2}{\pi}} * (x + 0.0044715x^3)))$

   1.10 **Swish (OPTIONAL)**: $x * sigmoid(x) = x * \dfrac{1}{1 + e^{-x}}$

   - **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 $f'(x)$:

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

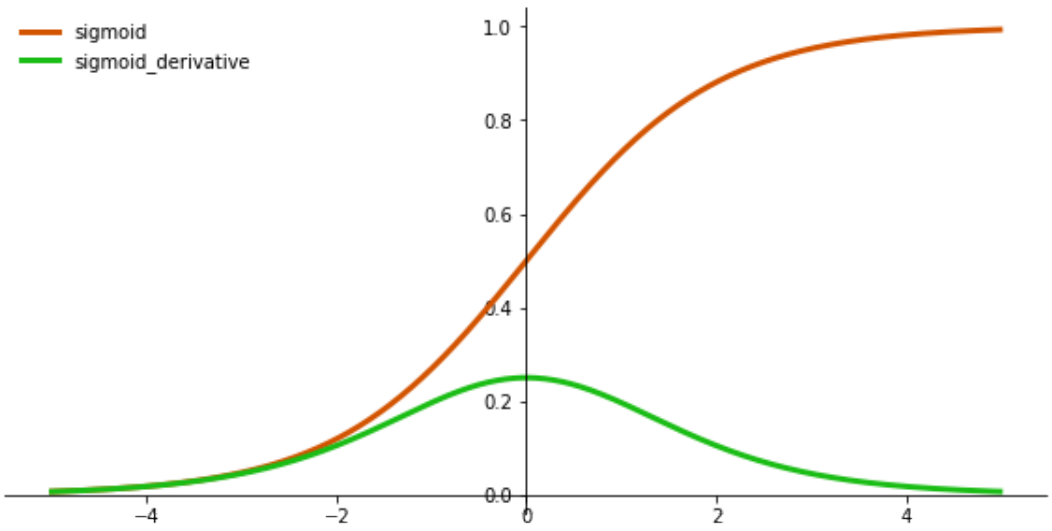
    - Tạo một array list các giá trị x trong range [-5, 5] mỗi step là 0.01 đơn vị (Cái này tùy các bạn lựa chọn):

```
x = np.arange(-5, 5, 0.01)
```

    - Thực hiện vẽ với hàm cho trước **plot_function_and_derivative** truyền vào array x vừa tạo, hàm $f(x)$ và $f'(x)$:

```
plot_function_and_derivative(x, sigmoid, sigmoid_derivative)
```

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

def plot_function_and_derivative(x, func, derivative):

    # 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,func(x), color="#d35400", linewidth=3, label=func.__name__)
    ax.plot(x,derivative(x), color="#1abd15", linewidth=3, label=derivative.__name__)

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

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

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

x = np.arange(-5, 5, 0.01)
plot_function_and_derivative(x, sigmoid, sigmoid_derivative)
```




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

2. **Xấp xỉ $\sqrt{2}$ (căn bậc hai của 2) bằng Newton's method**:

   **Newton's method**: Cho một function $f(x)$, bài toán yêu cầu tìm nghiệm của $f(x)$, chính là đi tìm $x$ để $f(x) = 0$. Đối với các phương trình đơn giản như $f(x)$ là một phương trình đường thẳng, phương trình bậc 2 hoặc 3, sẽ có công thức giúp ta tìm được nghiệm. Tuy nhiên đối với các phương trình bậc cao hơn hầu như sẽ không có công thức cụ thể để tìm nghiệm. Từ đó **Newton's method** ra đời giúp cho việc xấp xỉ nghiệm của một phương trình $f(x) = 0$ đơn giản hơn.

$$\textbf{Newton's Method} : x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

   **Các bước thực hiện Newton's Method**:

   - **Step 1**: Tìm đạo hàm f'(x) của function f(x)
   - **Step 2**: Khởi tạo giá trị đầu tiên $x_0$ (là giá trị bất kỳ).
   - **Step 3**: Tìm $f(x_0)$, $f'(x_0)$ bằng cách input $x_0$ vào $f(x)$ và $f'(x)$ (đạo hàm của f tại $x_0$)
   - **Step 4**: Tính $x_1$ bằng cách thay $x_0$ vào công thức **Newton's method**. $x_1 = x_0 - \frac{f(x_0)}{f'(x_0)}$
   - **Step 5**: Lặp lại K lần các bước trên để thu được $x_K$, K càng lớn thì $x_K$ càng tiến gần về nghiệm thực để $f(x) = 0$.

   **Yêu cầu bài làm**: **NOTE các bạn chú ý câu 2.1, 2.2 và 2.3 sẽ khởi tạo $x_0$ nên là số dương và càng gần N (cho câu 2.1) hoặc gần 2 (cho câu 2.2 và 2.3).**

   2.1 Viết function *compute_square_root* và dựa theo công thức sau để tìm xấp xỉ $\sqrt{N}$. Công thức này được thực hiện dựa trên Newton's method và là công thức tổng quát để tìm $\sqrt{N}$.

$$x_{i+1} = \frac{x_i + \frac{N}{x_i}}{2}$$

   - **Input**: N (số nguyên N>=2, được dùng để tìm $\sqrt{N}$), num_loops (số nguyên lớn hơn 1) là số lần lặp để tìm số xấp xỉ $\sqrt{N}$.

- **Output** giá trị xấp xỉ của $\sqrt{N}$. (num_loops càng lớn giá trị cần tìm càng xấp xỉ gần bằng). Dựa trên công thức trên $x_{num\_loop}$ sẽ là giá trị trả về

2.2 Tìm $\sqrt{2}$ bằng Newton's method, biết rằng $f(x) = x^2 - 2$ và tìm được nghiệm của $x$ để $f(x) = 0$, thì $x$ chính là giá trị xấp xỉ của $\sqrt{2}$. Viết hàm *compute_square_root2*

$$\textbf{Newton's Method} : x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

.

- **Input**: num_loops (số nguyên lớn hơn 1) là số lần lặp để tìm nghiệm xấp xỉ $\sqrt{2}$.
- **Output**: giá trị xấp xỉ của $\sqrt{2}$. (num_loops càng lớn giá trị cần tìm càng xấp xỉ gần bằng). Dựa trên Newton's method $x_{num\_loop}$ sẽ là giá trị trả về
- **NOTE: Các bạn dùng phương thức đạo hàm trung tâm để tính được $f'(x_i)$. Gía trị của đạo hàm f(x) tại $x_i$**

2.3 Tìm $\sqrt{2}$ bằng Newton's method, biết rằng $f(x) = x^2 - 2$ và tìm được nghiệm của $x$ để $f(x) = 0$, thì $x$ chính là giá trị xấp xỉ của $\sqrt{2}$. Viết hàm *compute_square_root3*

$$\textbf{Newton's Method} : x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

.

- **Input**: num_loops (số nguyên lớn hơn 1) là số lần lặp để tìm nghiệm xấp xỉ $\sqrt{2}$.
- **Output**: giá trị xấp xỉ của $\sqrt{2}$. (num_loops càng lớn giá trị cần tìm càng xấp xỉ gần bằng). Dựa trên Newton's method $x_{num\_loop}$ sẽ là giá trị trả về
- **NOTE: Các bạn dùng tensorflow framework (tf.GradientTape()) để tìm $f'(x_i)$. Gía trị của đạo hàm f(x) tại $x_i$**

```
# Example 2.1
# approximate sqrt(9) with  num_loops=3
print(compute_square_root(N=9, num_loops=3))
>> 112.52624883340555

# approximate sqrt(9) with  num_loops=15
print(compute_square_root(N=9, num_loops=15))
>> 3.0

# approximate sqrt(4) with  num_loops=10
print(compute_square_root(N=4, num_loops=10))
>> 2.0001428443098597

# Example 2.2
# approximate sqrt(2) with  num_loops=15,
# use Newtons Method with Central different
print(compute_square_root2(num_loops=15))
>> 1.4142135623730951

# Example 2.3
# approximate sqrt(2) with  num_loops=15,
# use Newtons Method tensorflow  (tf.GradientTape())
print(compute_square_root3(num_loops=15))
>> 1.414213562373095
```

3. **Khai triển Maclaurin với ba function lượng giác sau và thực hiện lập trình với kết quả đã khai triển để kiểm tra tính xấp xỉ**:

- Nếu $f$ có derivative của tất cả order tại $x = a$, **Taylor series** cho function $f$ tại $a$ là

$$\sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n = f(a) + f'(a)(x-a) + \frac{f''(a)}{2!}(x-a)^2 + \ldots + \frac{f^{(n)}(a)}{n!}(x-a)^n + \ldots$$

- **Taylor series** cho function $f$ tại 0 là **Maclaurin series**

$$\sum_{n=0}^{\infty} \frac{f^{(n)}(0)}{n!}(x)^n = f(0) + f'(0)(x) + \frac{f''(0)}{2!}(x)^2 + \ldots + \frac{f^{(n)}(0)}{n!}(x)^n + \ldots$$

Hình 2: Triển khai Taylor và Maclaurin cho một function tại $a$ (Taylor), hoặc 0 (Maclaurin)

3.1 **Khai triển Maclaurin với hàm sin**: $f(x) = sin(x)$ và biết rằng

- $sin'(x) = cos(x)$
- $cos'(x) = -sin(x)$
- $sin(0) = 0,\ cos(0) = 1$

3.2 **Khai triển Maclaurin với hàm sinh**: $f(x) = sinh(x)$ và biết rằng

- $sinh'(x) = cosh(x)$
- $cosh'(x) = sinh(x)$
- $sinh(0) = 0,\ cosh(0) = 1$

3.3 **Khai triển Maclaurin với hàm cosh**: $f(x) = cosh(x)$ (sử dụng thông tin về các hàm hạo cũng như công thức đạo hàm ở 3.2)

**Các bước làm bài**:

- **Step 1**: Các bạn đạo hàm $f(x)$ từ bậc 1 đến bậc 5 (tìm $f'(x)$, $f''(x)$, $f'''(x)$, $f^{(4)}(x)$, $f^{(5)}(x)$)
- **Step 2**: Các bàn tính giá trị đạo hàm tại $x = 0$ từ bậc 1 đến bậc 5. (Tính $f'(0)$, $f''(0)$, $f'''(0)$, $f^{(4)}(0)$, $f^{(5)}(0)$). Và tính $f(0)$.
- **Step 3**: Thay các giá trị vừa tính được vào công thức Maclaurin Series. Tìm quy luật lặp lại của các các giá trị để rút ra được công thức tổng quát cho xấp xỉ với bậc **n**.
- **Step 4**: Từ công thức tổng quát vừa tìm được ở trên, thực hiện code bằng python viết function nhận hai input **n** (bậc tối đa của đạo hàm muốn xấp xỉ $f(x)$) và **x** (giá trị bất kỳ). Output là kết quả xấp xỉ của $f(x)$ tại **x** tương ứng.
- **NOTE**:Từ Step1 đến Step3 các bạn có thể viết bằng latex sau đó gửi file, hoặc viết bằng markdown trên google colab. (Ví dụ hình 3)
- **NOTE**: Để test kết quả output Step4, các bạn chọn **x** bất kỳ sau đó thế và $f(x)$ thu được kết quả thật. Sau đó input **x** hàm vừa viết ở Step4 cùng với **n** (số nguyên càng lớn thì xấp xỉ càng đúng số thực) thu được kết quả xấp xỉ. Kiểm tra kết quả thật và xấp xỉ có gần nhau (nếu gần là đã thực hiện đúng). Các giá trị $x$ theo đơn vị radian.

**Example**: $f(x) = e^x$, tìm Maclaurin Series cho $f(x)$.

**Step1:** $f(x) = e^x$

- $f'(x) = e^x$
- $f''(x) = e^x$
- $f'''(x) = e^x$
- $f^{(4)}(x) = e^x$
- $f^{(5)}(x) = e^x$

**Step2:** $f(0) = e^0 = 1$

- $f'(0) = e^0 = 1$
- $f''(0) = e^0 = 1$
- $f'''(0) = e^0 = 1$
- $f^{(4)}(0) = e^0 = 1$
- $f^{(5)}(0) = e^0 = 1$

$$e^x \approx \sum_{k=0}^{\infty} \frac{x^k}{k!}$$

Công thức tổng quát khi quan sát tại step3

**Step3:** $f(x) = e^x$

- $p_0(x) = f(0) = 1$
- $p_1(x) = f(0) + f'(0)(x) = 1 + x$
- $p_2(x) = f(0) + f'(0)(x) + \frac{f''(0)}{2!} = 1 + x + \frac{1}{2!}x^2$
- $p_3(x) = f(0) + f'(0)(x) + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3$
- $p_4(x) = f(0) + f'(0)(x) + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 + \frac{f^{(4)}(0)}{4!}x^4 = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \frac{1}{4!}x^4$
- $p_5(x) = f(0) + f'(0)(x) + \frac{f''(0)}{2!}x^2 + \frac{f'''(0)}{3!}x^3 + \frac{f^{(4)}(0)}{4!}x^4 + \frac{f^{(5)}(0)}{5!}x^5 = 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + \frac{1}{4!}x^4 + \frac{1}{5!}x^5$

Hình 3: Ví dụ triển khai Maclaurin với hàm exponential

```
import numpy as np

# Example 3.1
approximate_sin(np.pi/3, 5)
>> 0.8660254034934827
np.sin(np.pi/3)
>> 0.8660254037844385

# Example 3.2
approximate_sinh(np.pi/3, 5)
>> 1.2493670502299645
np.sinh(np.pi/3)
>> 1.249367050523975

# Example 3.3
approximate_cosh(np.pi/3, 5)
>> 1.6002868540495738
np.cosh(np.pi/3)
>> 1.6002868577023863
```