Pytorch

In [ ]:
```
import torch
print(torch.__version__)  # Xem phiên bản PyTorch có sẵn
```

```
2.5.1+cu124
```

In [ ]:
```
import torch

torch.cuda.is_available()
```

Out[ ]:
```
True
```

Sử dụng GPU và Cuda

In [ ]:
```
torch.cuda.current_device()
```

Out[ ]:
```
0
```

In [ ]:
```
torch.cuda.get_device_name(0)
```

Out[ ]:
```
'Tesla T4'
```

In [ ]:
```
# trả về mức sử dụng bộ nhớ gpu hiện taijtheo tensors tính bằng byte cho thiết bị
torch.cuda.memory_allocated()
```

Out[ ]:
```
0
```

In [ ]:
```
# trả về bộ nhớ gpu hiện tại được quản lý bởi bộ phân bổ bộ nhớ đệm hiện tại
torch.cuda.memory_cached()
```

```
<ipython-input-7-107e6c788df7>:2: FutureWarning: `torch.cuda.memory_cached` has been renamed to `torch.cuda.memory_reserved`
  torch.cuda.memory_cached()
```

Out[ ]:
```
0
```

Dataset with Pytorch

In [ ]:
```
# loading data Iris
import pandas as pd
from sklearn.datasets import load_iris

iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)

print(df.head())  # Hiển thị 5 dòng đầu tiên
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)
0                5.1               3.5                1.4               0.2
1                4.9               3.0                1.4               0.2
2                4.7               3.2                1.3               0.2
3                4.6               3.1                1.5               0.2
4                5.0               3.6                1.4               0.2
```

In [ ]:
```
import torch
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

df = pd.read_csv('Iris.csv')
df.head()
```

Out[ ]:

| | Id | SepalLengthCm | SepalWidthCm | PetalLengthCm | PetalWidthCm | Species |
|---|---|---|---|---|---|---|
| **0** | 1 | 5.1 | 3.5 | 1.4 | 0.2 | Iris-setosa |
| **1** | 2 | 4.9 | 3.0 | 1.4 | 0.2 | Iris-setosa |
| **2** | 3 | 4.7 | 3.2 | 1.3 | 0.2 | Iris-setosa |
| **3** | 4 | 4.6 | 3.1 | 1.5 | 0.2 | Iris-setosa |
| **4** | 5 | 5.0 | 3.6 | 1.4 | 0.2 | Iris-setosa |

In [ ]:
```
# xem bao nhiêu hàng nhiu cột
df.shape
```

Out[ ]:
```
(150, 6)
```

In [ ]:
```
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split

le = LabelEncoder()
X = df.drop(['Id','Species'], axis=1).values
y = le.fit_transform(df['Species'].values)

# chia dữ liệu với test size = 0.2
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta

X_train = torch.FloatTensor(X_train)
X_test = torch.FloatTensor(X_test)
y_train = torch.LongTensor(y_train).reshape(-1, 1)
y_test = torch.LongTensor(y_test).reshape(-1, 1)
```

In [ ]:
```
print(f"train size {len(X_train)}")
```

```
train size 120
```

In [ ]:
```
labels, counts = np.unique(y_train, return_counts=True)
print(labels, counts)
```

```
[0 1 2] [40 41 39]
```

In [2]:
```
# Tính đạo hàm bằng pytorch

# Cho y = 2x^4 + x^3 + 3x^2 + 5x +1
# tính y'

import torch
```

In [3]:
```
# Tạo một tensor với requirments_grad được đặt thành true

x = torch.tensor(2.0, requires_grad=True)
print(x)
print(x.grad)
```

```
tensor(2., requires_grad=True)
None
```

In [4]:
```
# định nghĩa hàm
y = 2*x**4 + x**3 + 3*x**2 + 5*x + 1
print(y)
```

```
tensor(63., grad_fn=<AddBackward0>)
```

In [ ]:
```
y.grad_fn
```

Out[ ]:
```
<AddBackward0 at 0x79f7a0a26e60>
```

In [ ]:
```
# Thực hiện truyền ngược và tính toán các gradient
y.backward()
```

In [ ]:
```
# Kết quả đạo hàm
print(x.grad)
```

```
tensor(93.)
```

BT

1. Tính y' của y = 5x^6 +3x^3 + 2x^1 +x +2x +5x^4 +1
2. Tìm độ dốc của đa thức trên tại điểm nào

In [9]:
```
# Tính y' của y = 5x^6 +3x^3 + 2x^1 +x +2x +5^4 +1
import torch
x = torch.tensor(1.0, requires_grad=True)
y = 5*x**6 + 3*x**3 + 2*x**1 + x + 2*x + 5*x**4 + 1

# Tìm độ dốc của đa thức trên tại điểm nào
print(f"Giá trị của y tại x=1: {y.item()}")

# đáp án x = 1, y =19
```

```
Giá trị của y tại x=1: 19.0
```

BTVN 1:

1. Tạo một tensor x có giá trị ban đầu là 2.0. định nghĩa hàm số:

- Tính gradient
- y = x^3 + 2x^2 +5x +1
- Hãy tính dy/dx tại giá trị của x
- Dùng phương phóa Gradient Descent với learning rate alpha = 0.1 để cập nhật giá trị x trong 10 vòng

BTVN 2: Tạo một tập dữ liệu giả lập với x là số giờ học(ngẫu nhiên từ 1 tới 10) và y là số điểm được tính theo công thức y = 3x +5 + noise Với noise là một giá trị ngẫu nhiên nhỏ

1. Khởi tạo tham số w và b ngẫu nhiên với requires_grad=True
2. Tính MSE
3. Tính gradient
4. Cập nhật tham số w và b bằng grandient Descent với Learning rate alpha = 0.01
5. Lặp lại quá trình trên trong 100 vòng lặp và quan sát sự hội tụ mô hình

In [24]:
```python
# BTVN 1:
import torch

# Khởi tạo x và yêu cầu tính gradient
x = torch.tensor(2.0, requires_grad=True)

# Learning rate và số vòng lặp
alpha = 0.1
num_iterations = 10

# Quá trình Gradient Descent
for i in range(num_iterations):
    y = x**3 + 2*x**2 + 5*x + 1  # Tính giá trị hàm số
    y.backward()  # Tính đạo hàm dy/dx
    gradient = x.grad.item()  # Lưu lại giá trị đạo hàm

    with torch.no_grad():  # Cập nhật x mà không tính gradient
        x -= alpha * x.grad

    x.grad.zero_()  # Reset gradient để tránh cộng dồn

    print(f"Vòng {i+1}: x = {x.item():.4f}, Gradient = {gradient:.4f}")

print("Cập nhật hoàn tất!")
```

```
Vòng 1: x = -0.5000, Gradient = 25.0000
Vòng 2: x = -0.8750, Gradient = 3.7500
Vòng 3: x = -1.2547, Gradient = 3.7969
Vòng 4: x = -1.7251, Gradient = 4.7040
Vòng 5: x = -2.4278, Gradient = 7.0274
Vòng 6: x = -3.7250, Gradient = 12.9717
Vòng 7: x = -6.8977, Gradient = 31.7268
Vòng 8: x = -18.9120, Gradient = 120.1433
Vòng 9: x = -119.1464, Gradient = 1002.3439
Vòng 10: x = -4330.7476, Gradient = 42116.0117
Cập nhật hoàn tất!
```

In [26]:

```python
# BTVN2
import torch

# Tạo dữ liệu giả lập
torch.manual_seed(42)  # Để kết quả ngẫu nhiên có thể tái lập
x = torch.randint(1, 11, (20,), dtype=torch.float32)  # 20 giá trị x từ 1 đến 10
noise = torch.randn(20)  # Nhiễu ngẫu nhiên
y = 3 * x + 5 + noise  # Công thức y = 3x + 5 + noise

# Khởi tạo tham số w, b ngẫu nhiên
w = torch.randn(1, requires_grad=True)
b = torch.randn(1, requires_grad=True)

# Learning rate và số vòng lặp
alpha = 0.01
num_iterations = 100

# Gradient Descent
for i in range(num_iterations):
    # Dự đoán y
    y_pred = w * x + b

    # Tính Mean Squared Error (MSE)
    loss = ((y_pred - y) ** 2).mean()

    # Tính gradient
    loss.backward()

    # Cập nhật w, b
    with torch.no_grad():
        w -= alpha * w.grad
        b -= alpha * b.grad

    # Reset gradient
    w.grad.zero_()
    b.grad.zero_()

    # In loss mỗi 10 vòng
    if (i + 1) % 10 == 0:
        print(f"Vòng {i+1}: MSE = {loss.item():.4f}, w = {w.item():.4f}, b = {b.ite

print("Huấn luyện hoàn tất!")
print(f"Trọng số cuối cùng: w = {w.item():.4f}, b = {b.item():.4f}")
```

```
Vòng 10: MSE = 5.0599, w = 3.6198, b = 1.3066
Vòng 20: MSE = 4.7781, w = 3.5928, b = 1.4718
Vòng 30: MSE = 4.5164, w = 3.5666, b = 1.6310
Vòng 40: MSE = 4.2734, w = 3.5414, b = 1.7843
Vòng 50: MSE = 4.0478, w = 3.5170, b = 1.9321
Vòng 60: MSE = 3.8383, w = 3.4936, b = 2.0745
Vòng 70: MSE = 3.6438, w = 3.4710, b = 2.2118
Vòng 80: MSE = 3.4632, w = 3.4492, b = 2.3440
Vòng 90: MSE = 3.2955, w = 3.4282, b = 2.4714
Vòng 100: MSE = 3.1397, w = 3.4080, b = 2.5942
Huấn luyện hoàn tất!
Trọng số cuối cùng: w = 3.4080, b = 2.5942
```

In [10]:
```python
# pytorch with tensor
import torch
import numpy as np

torch.__version__
```

Out[10]: '2.5.1+cu124'

In [11]:
```python
# Chuyển đổi mảng numpy sang tensor pytorch
arr = np.array([1,2,3,4,5])
print(arr)
print(arr.dtype)
print(type(arr))
```

```
[1 2 3 4 5]
int64
<class 'numpy.ndarray'>
```

In [12]:
```python
x = torch.from_numpy(arr)
print(x)
print(x.dtype)
print(type(x))
```

```
tensor([1, 2, 3, 4, 5])
torch.int64
<class 'torch.Tensor'>
```

In [13]:
```python
arr2 = np.arange(0,12).reshape(4,3)
print(arr2)
```

```
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
```

In [14]:
```python
x2 = torch.from_numpy(arr2)
print(x2)
print(x2.dtype)
print(x2.type())
```

```
tensor([[ 0,  1,  2],
        [ 3,  4,  5],
        [ 6,  7,  8],
        [ 9, 10, 11]])
torch.int64
torch.LongTensor
```

Copy and sharring

In [15]:
```
arr = np.arange(0,5)
x = torch.from_numpy(arr)
print(x)
print(arr)
```

```
tensor([0, 1, 2, 3, 4])
[0 1 2 3 4]
```

In [16]:
```
arr[0] = 99
print(x)
```

```
tensor([99,  1,  2,  3,  4])
```

In [17]:
```
# BTVN 3:
# Giai thích lý do tại trong 2 trường hợp ở trên một cái dùng a[0] = 99 lại thay đổ
# -> vì khi chạy torch.tensor là nó tạo một bản sao clone của arr nên chạy nó vẫn k
```

In [21]:
```
# BTVN 4:
# về nahf tạo tensor với empty , zeros, ones, random, reshape với view và view as

# emty
x = torch.empty(3, 3)
print("tensor voi emty")
print(x)

# zeros
x = torch.zeros(3, 3)  # Tạo tensor toàn số 0
print("\ntensor voi zeros")
print(x)

# ones
x = torch.ones(3, 3)  # Tạo tensor toàn số 1
print("\ntensor voi ones")
print(x)

# random
x = torch.rand(3, 3)  # Tạo tensor với giá trị ngẫu nhiên từ 0 đến 1
print("\ntensor voi random")
print(x)

# reshape
x = torch.arange(12)  # Tạo tensor từ 0 đến 11
y = x.reshape(3, 4)  # Chuyển thành ma trận (3,4)
print("\ntensor voi reshape")
print(y)

# reshape voi view
```

```python
x = torch.arange(12)
y = x.view(3, 4)  # Giống reshape nhưng dùng chung bộ nhớ với tensor gốc
print("\ntensor voi reshape voi view")
print(y)

# reshape voiws view as
x = torch.arange(12)
z = torch.zeros(3, 4)  # Tạo tensor mẫu
y = x.view_as(z)  # Chuyển x thành cùng kích thước với z
print("\ntensor voi reshape voi view as")
print(y)
```

```
tensor voi emty
tensor([[1.5638e-34, 0.0000e+00, 1.6250e-34],
        [0.0000e+00, 2.1622e-01, 5.2089e-01],
        [5.2166e-01, 9.3347e-02, 3.0279e-01]])

tensor voi zeros
tensor([[0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.]])

tensor voi ones
tensor([[1., 1., 1.],
        [1., 1., 1.],
        [1., 1., 1.]])

tensor voi random
tensor([[0.0837, 0.1452, 0.4051],
        [0.2222, 0.3024, 0.8100],
        [0.0528, 0.5751, 0.3528]])

tensor voi reshape
tensor([[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11]])

tensor voi reshape voi view
tensor([[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11]])

tensor voi reshape voi view as
tensor([[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11]])
```

Deadline: 15/3