Pytorch

```python
import torch
print(torch.__version__)  # Xem phiên bản PyTorch có sẵn
```

```
2.5.1+cu124
```

```python
import torch

torch.cuda.is_available()
```

```
True
```

Sử dụng GPU và Cuda

```python
torch.cuda.current_device()
```

```
0
```

```python
torch.cuda.get_device_name(0)
```

```
'Tesla T4'
```

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

```
0
```

```python
# trả về bộ nhớ gpu hiện tại được quản lý bởi bộ phận bố bộ nhớ đệm hiện tại
torch.cuda.memory_cached()
```

```
<ipython-input-6-107e6c788df7>:2: FutureWarning: `torch.cuda.memory_cached` has been renamed to `torch.cuda.memory_reserved`
  torch.cuda.memory_cached()

0
```

Dataset with Pytorch

```python
# 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

```python
import torch
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

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

	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

+ Code + Markdown

```python
# xem bao nhiêu hàng nhiu cột
df.shape
```

```
(150, 6)
```

```python
from sklearn.preprocessing import L
from sklearn.model_selection import

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

# 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_state=42)

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)
```

```
(function) def train_test_split(
    *arrays: Any,
    test_size: Float | None = None,
    train_size: Float | None = None,
    random_state: Int | None = None,
    shuffle: bool = True,
    stratify: Any | None = None
) -> list
```

```python
print(f"train size {len(X_train)}")
```

```
train size 120
```

```
labels, counts = np.unique(y_train, return_counts=True)
print(labels, counts)
```

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

```
# Tính đạo hàm bằng pytorch

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

import torch
```

"thành": Unknown word. cSpell
View Problem (Alt+... Quick Fix... (Ctrl...

```
# Tạo một tensor với requirments_grad được đặt thành true

x = torch.tensor(4.0, requires_grad=True)
```

```
# định nghĩa hàm
y = 2*x**4 + x**3 + 3*x**2 + 5*x + 1
```

```
tensor(645., grad_fn=<AddBackward0>)
```

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

```
# Kết quả đạo hàm
print(x.grad)
```

```
tensor(589.)
```

```
print(y)
```

```
tensor(645., grad_fn=<AddBackward0>)
```

```python
import torch

# Khởi tạo x
x = torch.tensor(2.0, requires_grad=True)

# Learning rate (Giảm xuống để tránh lỗi)
alpha = 0.001
num_iterations = 500  # Số vòng lặp

# Gradient Descent
for i in range(num_iterations):
    # Định nghĩa lại y để không bị tích lũy
    y = 5*x**6 + 3*x**3 + 2*x + x + 2*x + 5*x**4 + 1

    # Tính đạo hàm dy/dx
    y.backward()
    gradient = x.grad.item()  # Lấy giá trị gradient

    # Kiểm tra gradient trước khi cập nhật
    if abs(gradient) > 1e6:
        print(f"⚠️ Gradient quá lớn ({gradient}), dừng cập nhật!")
        break

    # Cập nhật giá trị của x
    with torch.no_grad():
        x -= alpha * x.grad

    # Xóa gradient để tránh tích lũy sai lệch
    x.grad.zero_()

    # In kết quả
    print(f"Vòng {i+1}: x = {x.item()}, Gradient = {gradient}")
```

```
Vòng 1: x = 0.8389999866485596, Gradient = 1161.0
Vòng 2: x = 0.8033810257911682, Gradient = 35.61893844604492
Vòng 3: x = 0.7721619606018066, Gradient = 31.21906280517578
Vòng 4: x = 0.7443530559539795, Gradient = 27.80888557434082
Vòng 5: x = 0.7192630171775818, Gradient = 25.090038299560547
Vòng 6: x = 0.6963897943496704, Gradient = 22.873218536376953
Vòng 7: x = 0.6753573417663574, Gradient = 21.032445907592773
Vòng 8: x = 0.6558767557144165, Gradient = 19.480600357055664
Vòng 9: x = 0.6377212405204773, Gradient = 18.155492782592773
Vòng 10: x = 0.6207096576690674, Gradient = 17.01155662536621
Vòng 11: x = 0.6046950221061707, Gradient = 16.01463508605957
Vòng 12: x = 0.5895563960075378, Gradient = 15.138622283935547
Vòng 13: x = 0.5751931667327881, Gradient = 14.363235473632812
Vòng 14: x = 0.5615206956863403, Gradient = 13.672462463378906
Vòng 15: x = 0.5484671592712402, Gradient = 13.053511619567871
Vòng 16: x = 0.5359711050987244, Gradient = 12.496033668518066
Vòng 17: x = 0.5239795446395874, Gradient = 11.991569519042969
Vòng 18: x = 0.512446403503418, Gradient = 11.533141136169434
Vòng 19: x = 0.5013314485549927, Gradient = 11.114934921264648
Vòng 20: x = 0.4905993640422821, Gradient = 10.732072830200195
Vòng 21: x = 0.48021891713142395, Gradient = 10.380435943603516
Vòng 22: x = 0.47016239166259766, Gradient = 10.05651569366455
Vòng 23: x = 0.4604050815105438, Gradient = 9.757311820983887
Vòng 24: x = 0.4509248435497284, Gradient = 9.480237007141113
Vòng 25: x = 0.4417017996311188, Gradient = 9.223055839538574
...
Vòng 497: x = -0.6457210183143616, Gradient = 6.556510925292969e-05
Vòng 498: x = -0.6457210779190063, Gradient = 6.389617919921875e-05
Vòng 499: x = -0.6457211375236511, Gradient = 6.127357482910156e-05
Vòng 500: x = -0.6457211971282959, Gradient = 5.91278076171875e-05
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

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

```
# cho  y = 5x^6 +3x^3 + 2x^1 +x +2x +5x^4 +1


# Tìm độ dốc của đa thức trên tại điểm nào



# đáp án x = 1, y =19
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
print(y)
```

```
tensor(19., grad_fn=<AddBackward0>)
```

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

```
x = torch.tensor(2.0, requires_grad=True)
y = x**3 + 2*x**2 + 5*x + 1
y.backward()
print(x.grad)
```

```
tensor(25.)
```

```python
alpha = 0.1

# Số vòng lặp
num_iterations = 10

# Quá trình Gradient Descent
for i in range(num_iterations):
    # Định nghĩa hàm y
    y = x**3 + 2*x**2 + 5*x + 1

    # Tính đạo hàm dy/dx
    y.backward()
    gradient = x.grad.item()  # Lấy giá trị gradient

    # Cập nhật giá trị của x theo Gradient Descent
    with torch.no_grad():  # Tắt tính toán gradient cho phần cập nhật
        x -= alpha * x.grad

    # Xóa gradient để tránh tích lũy sai lệch
    x.grad.zero_()

    # In kết quả sau mỗi vòng lặp
    print(f"Vòng {i+1}: x = {x.item()}, Gradient = {gradient}")
```

```
Vòng 1: x = -3.0, Gradient = 50.0
Vòng 2: x = -5.0, Gradient = 20.0
Vòng 3: x = -11.0, Gradient = 60.0
Vòng 4: x = -43.400001525878906, Gradient = 324.0
Vòng 5: x = -591.6080932617188, Gradient = 5482.08056640625
Vòng 6: x = -105355.5078125, Gradient = 1047638.9375
Vòng 7: x = -3329998336.0, Gradient = 33298927616.0
Vòng 8: x = -3.32666671295837e+18, Gradient = 3.326666767933951e+19
Vòng 9: x = -3.3200133669524894e+36, Gradient = 3.3200134303350194e+37
Vòng 10: x = -inf, Gradient = inf
```

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

```python
import random

# 1. Tạo tập dữ liệu giả lập (x từ 1 đến 10, y = 3x + 5 + noise)
torch.manual_seed(42)  # Để kết quả tái lập
x = torch.FloatTensor([random.uniform(1, 10) for _ in range(20)])  # 20 điểm dữ liệu
noise = torch.randn_like(x) * 0.5  # Nhiễu nhỏ
y = 3 * x + 5 + noise  # Hàm thực tế
```

Python

```python
# 2. Khởi tạo tham số w, b ngẫu nhiên
w = torch.randn(1, requires_grad=True)
b = torch.randn(1, requires_grad=True)
alpha = 0.01
```

Python

```python
# 5. Lặp lại Gradient Descent 100 lần
for epoch in range(100):
    # 3. Dự đoán y_hat
    y_hat = w * x + b

    # 3. Tính hàm mất mát MSE
    loss = ((y_hat - y) ** 2).mean()

    # 4. Tính gradient
    loss.backward()

    # Cập nhật w, b bằng Gradient Descent
    with torch.no_grad():  # Tắt tracking gradient để cập nhật giá trị
        w -= alpha * w.grad
        b -= alpha * b.grad

        # Reset gradient về 0
        w.grad.zero_()
        b.grad.zero_()

    # Hiển thị mỗi 10 vòng
    if epoch % 10 == 0:
        print(f'Epoch {epoch}: MSE = {loss.item()}, w = {w.item()}, b = {b.item()}')
```

```
Epoch 0: MSE = 917.3636474609375, w = 3.2155215740203857, b = -0.17368602752685547
Epoch 10: MSE = 4.477686882019043, w = 3.702928066253662, b = 0.053219910711050034
Epoch 20: MSE = 4.244551658630371, w = 3.6816742420196533, b = 0.2040921300649643
Epoch 30: MSE = 4.024136543273926, w = 3.66100811958313, b = 0.3507910668849945
Epoch 40: MSE = 3.8157432079315186, w = 3.640913963317871, b = 0.4934321641921997
Epoch 50: MSE = 3.618720293045044, w = 3.621375560760498, b = 0.632127583026886
Epoch 60: MSE = 3.432447910308838, w = 3.6023776531219482, b = 0.7669866681098938
Epoch 70: MSE = 3.256335496902466, w = 3.583904981613159, b = 0.8981153964996338
Epoch 80: MSE = 3.089834690093994, w = 3.565943479537964, b = 1.025617003440857
Epoch 90: MSE = 2.9324164390563965, w = 3.548478603363037, b = 1.1495918035507202
```

```python
print(f'Final: w = {w.item()}, b = {b.item()}')
```

```
Final: w = 3.5331737995147705, b = 1.2582343816757202
```

```python
# pytorch with tensor
import torch
import numpy as np

torch.__version__
```

```
'2.5.1+cu124'
```

```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'>
```

```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'>
```

```python
arr2 = np.arange(0,12).reshape(4,3)
print(arr2)
```

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

```
    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

+ Code

```
    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]
```

```
    arr[0] = 99
    print(x)
```
[44]

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

```
    print(arr)
```
[45]

```
[99  1  2  3  4]
```

```
    arr = np.arange(0,5)
    x = torch.tensor(arr)
    arr[0] = 99
    print(x)
    print(arr)
```
[46]

```
tensor([0, 1, 2, 3, 4])
[99  1  2  3  4]
```

```
# BTVN 3:
# Giải thích lý do tại trong 2 trường hợp ở trên một cái dùng a[0] = 99 lại thay đổi mà cái còn lại thì hong?
vì khi chạy torch.tensor là nó tạo một bản sao clone của arr nên chạy nó vẫn ko thay đổi
```

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

```
x = torch.empty(3, 3)
print(x)
```

```
tensor([[-7.6669e+17,  4.3334e-41,  2.0229e-32],
        [ 0.0000e+00, -4.2904e+17,  4.3334e-41],
        [ 1.5065e-38,  0.0000e+00,  1.5065e-38]])
```

```
x = torch.zeros(3, 3)  # Tạo tensor toàn số 0
print(x)
```

```
tensor([[0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.]])
```

```python
x = torch.zeros(3, 3)  # Tạo tensor toàn số 0
print(x)
```

```
tensor([[0., 0., 0.],
        [0., 0., 0.],
        [0., 0., 0.]])
```

```python
x = torch.ones(3, 3)  # Tạo tensor toàn số 1
print(x)
```

```
tensor([[1., 1., 1.],
        [1., 1., 1.],
        [1., 1., 1.]])
```

```python
x = torch.rand(3, 3)  # Tạo tensor với giá trị ngẫu nhiên từ 0 đến 1
print(x)
```

```
tensor([[0.6343, 0.3644, 0.7104],
        [0.9464, 0.7890, 0.2814],
        [0.7886, 0.5895, 0.7539]])
```

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

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

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

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

```python
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(y)
```

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