Tensores

O que é um tensor?

Um **tensor** é uma estrutura de dados usada para guardar números de forma ordenada. Ele pode ser desde um simples vetor até uma pilha de matrizes. Os tensores são o **coração de tudo** no PyTorch.

Um exemplo prático do uso de tensores é no treinamento de modelos de IA: a rede só consegue aprender se os dados de entrada estiverem no formato de um tensor. Por exemplo, se quisermos treinar um modelo com uma sequência de texto (como em modelos de linguagem natural), precisamos transformar esse texto em palavras, e depois converter cada palavra em um número. Esse processo se chama *tokenizaçã**o***, e o resultado final é um tensor contendo números que representam o texto.

No PyTorch, tanto vetores quanto matrizes (e até estruturas com mais dimensões) são chamados de **tensores**. Você pode criar qualquer um deles usando o seguinte comando:

```
import torch

x = torch.tensor([1, 2, 3])

print(x)
```

Nesse exemplo criamos um simples vetor contendo os números de 1 a 3 e se imprimirmos na tela o nosso vetor X o que aparece é justamente:

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

Se quisermos criar uma **matriz**, é igualmente simples. Basta lembrar que, para criar tanto vetores quanto matrizes, usamos sempre o comando `torch.tensor()`, como no exemplo abaixo:

```
import torch

matriz = torch.tensor([[2,2],
                       [4,4]])

print(matriz)
```

Operações com tensores

Como o PyTorch lida com operações com tensores?

Assim como na álgebra linear — com multiplicações de matrizes por vetores, somas, subtrações e multiplicação por escalares — no PyTorch também usamos essas operações com frequência. Afinal, a álgebra linear é um dos pilares da Inteligência Artificial. A boa notícia é que você não vai precisar passar o dia todo calculando derivadas ou enchendo folhas com sistemas de equações: o PyTorch faz esses cálculos automaticamente.
Ainda assim, é muito importante que você entenda como essas operações funcionam por trás dos bastidores. Isso vai te ajudar a compreender como as redes neurais realmente aprendem, como os pesos são ajustados e como os modelos evoluem durante o treinamento.

Operações básicas

Já aprendemos a criar tensores. Agora, vamos dar um passo adiante e aprender como realizar operações aritméticas com eles — como soma, subtração, multiplicação por escalar e até multiplicação entre tensores.

No *PyTorch*, isso é muito simples. Basta criar dois tensores e usar os operadores aritméticos diretamente. O *PyTorch* executa a operação automaticamente, como no exemplo abaixo, onde somamos dois tensores:

```
import torch

a = torch.tensor([1, 2, 3])

b = torch.tensor([2, 2, 2])

soma = a + b

print(soma)
```

Para que a soma entre tensores funcione corretamente, eles precisam ter o mesmo tamanho — ou seja, as mesmas dimensões.

No exemplo abaixo, temos dois tensores com três números cada. Isso permite que o PyTorch some os elementos posição por posição e nos retorne um novo tensor com o resultado:

```
tensor([3, 4, 5])
```

Tudo vai depender do que você está tentando fazer e de qual operador aritmético está usando, como mostrado no exemplo abaixo:

```python
import torch

# Criando dois tensores
a = torch.tensor([2,3,4])

b = torch.tensor([3,1,2])

# Operações básicas
soma = a + b

mult = a * b

sub = a - b

div = a / b

# Multiplicação por escalar (por um número)
escalar_A = a * 3

escalar_B = b * 3

# Imprimindo os resultados
print(f'A soma dos tensores resulta em: {soma}\n')

print(f'A multiplicação dos tensores resulta em: {mult}\n')

print(f'A subtração resulta em: {sub}\n')

print(f'A divisão dos tensores resulta em: {div}\n')

print(f'Multiplicando A pelo escalar 3 resultará em: {escalar_A}\n')

print(f'Multiplicando B pelo escalar 3 resultará em: {escalar_B}\n')
```

Você pode acessar esse código e executá-lo através do arquivo **“tensores.py”** presente neste módulo.