PyTorch

Felipe Noronha e Izabela Fonseca

16 de Novembro, IME-USP

Roteiro

- Introdução
- Módulos
- Exemplo NN
- Diferenças com TF
- Pytorch Geometric

O que é?

- Biblioteca de machine learning
- Desenvolvida, principalmente, pelo Facebook e GitHub
- Possui API em Python e C++
- Usa recursos de GPU

Principais módulos

torch.Tensor

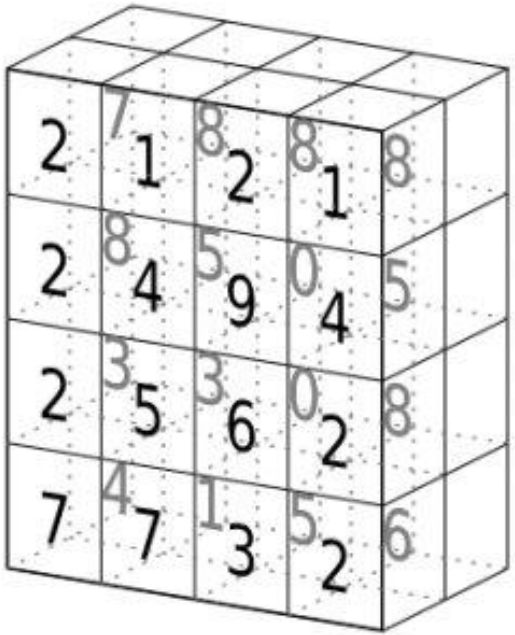
- Matriz multidimensional, generaliza escalar e vetor
- Implementados com o mesmo conjunto de operações que uma numpy array
- Principal classe usada usada nas computações, podendo ter vários tipos
- É versátil, podendo ser armazenado na GPU ou CPU

't'
'e'
'n'
's'
'o'
'r'

tensor of dimensions [6]
(vector of dimension 6)

3	1	4	1
5	9	2	6
5	3	5	8
9	7	9	3
2	3	8	4
6	2	6	4

tensor of dimensions [6,4]
(matrix 6 by 4)



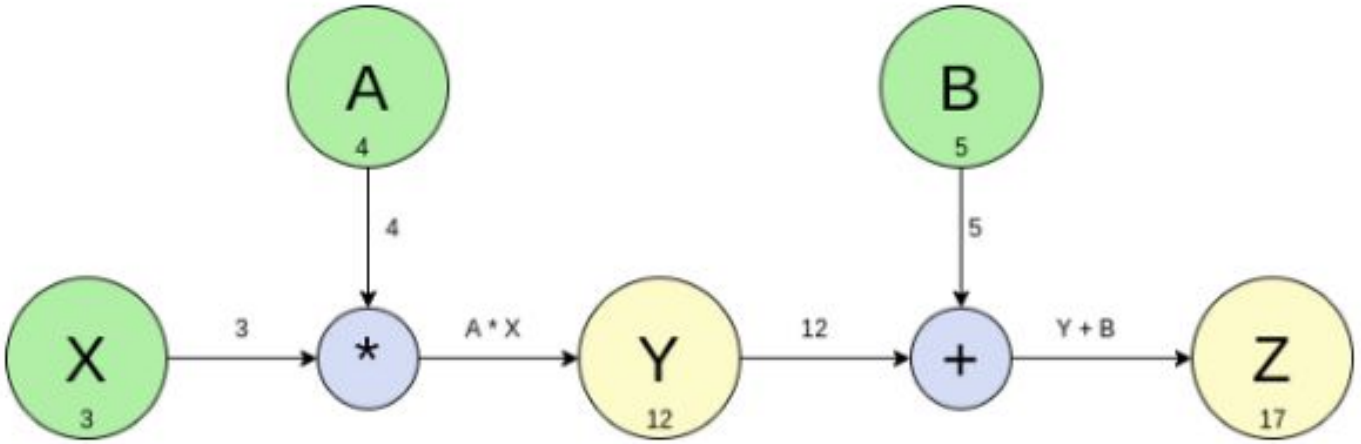
tensor of dimensions [4,4,2]

torch.autograd

- Módulo que implementa diferenciação automática
- Grafo de computação dinâmico
- Raízes são os nós de saída
- Folhas são os nós de entrada
- É construído no forward pass e usado para se obter os gradientes da backpropagation

```
x = torch.tensor(3.)
a = torch.tensor(4.)
b = torch.tensor(5.)


y = a * x
z = y + b
```



torch.nn

- Oferece os blocos de construção para as redes neurais
- Camadas (layers): pooling, convolução, normalização, transformação, etc
- Funções de ativação (activation functions): ReLu, Sigmoid, Softmax, etc
- Funções de perda (loss functions): CrossEntropy, MSE, L1, etc

Exemplo de implementação

PyTorch vs TensorFlow

PyTorch

TensorFlow

PyTorch

- Paralelização dos dados automática, podendo ter suas configurações alteradas manualmente

TensorFlow

- Paralelização de dados precisa ser configurada manualmente

PyTorch

- Consegue manipular os tensors à medida que desenvolve os códigos

TensorFlow

- Não permite a manipulação dos tensors após a sua criação, o que dificulta as correções do código

PyTorch

- Não possui solução automatizada para Deploy
- Não há uma solução nativa para monitoramento do modelo nem das bases de treino e aplicação

TensorFlow

- Utiliza o TensorFlow Extended (TFX) para deploy automatizado
- TFX disponibiliza uma análise do dataset de treino em relação ao dataset de aplicação

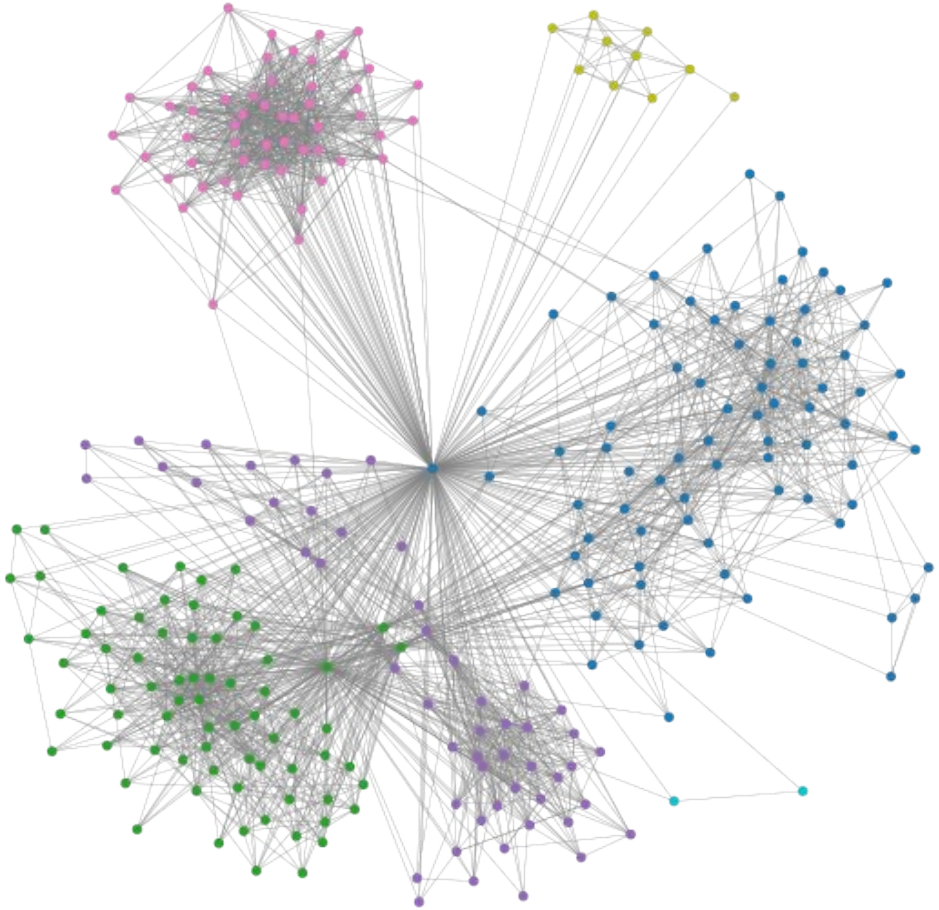
PyTorch

TensorFlow

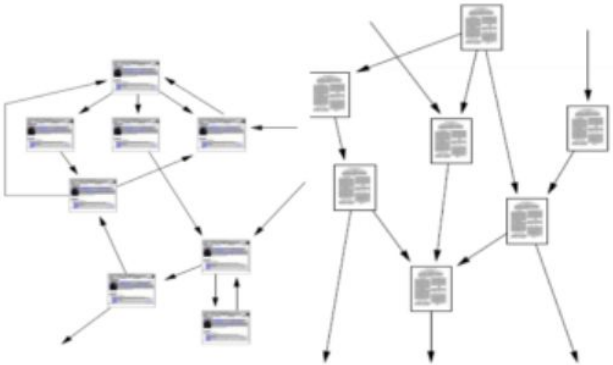
PyToch Geometric

Grafos

$$G = (V, L)$$

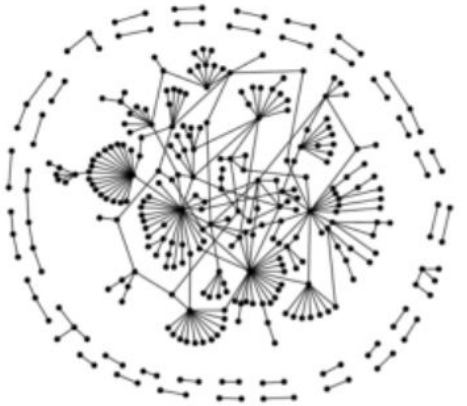


Exemplos



Redes de informação:
Web e citações

Redes Sociais



Redes Biomédicas

Redes Neurais para Grafos (GNN)

- Modelo apresentado pela primeira vez para a comunidade científica em 2009
- Aplicação em um problema de classificação
- Este modelo aprendeu a representar cada um dos vértices com um vetor de estado que continha informações sobre os vértices vizinhos
- Este vetor de estado é passado para a função de saída para permitir a obtenção do resultado

Redes Neurais para Grafos (GNN)

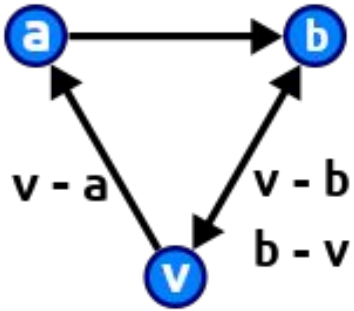
Feature do vértice v

Feature dos links conectados ao vértice v

Features dos vértices vizinhos de v

$$\mathbf{h}_v = f(\mathbf{x}_v, \mathbf{x}_{co[v]}, \mathbf{h}_{ne[v]}, \mathbf{x}_{ne[v]})$$

Vetor de estado que contém as informações dos vértices vizinhos (d)



Estados dos vértices vizinhos de v

Redes Neurais para Grafos (GNN)

Como é buscada uma solução única, aplica-se o teorema do ponto fixo de Banach para tornar tal função um processo iterativo, o que resulta na equação abaixo. Tal operação é chamada de **passagem de mensagem** ou **agregação de vizinhança**.

$$\mathbf{H}^{t+1} = F(\mathbf{H}^t, \mathbf{X})$$

Redes Neurais para Grafos (GNN)

$$\mathbf{o}_v = g(\mathbf{h}_v, \mathbf{x}_v)$$

- A saída **O_v** da GNN é computada passando-se os estados e as features para a função de saída ***g***.
- **g** e **f** podem ser interpretadas como uma rede neural feed-forward totalmente conectada.

Redes Neurais para Grafos (GNN)

$$loss = \sum_{i=1}^{p} (\mathbf{t}_i - \mathbf{o}_i)$$

A perda pode ser formulada como mostrada acima, e pode ser otimizada usando gradiente descendente.

PyTorch Geometric

- Possui as GNNs mais recentes já implementadas e prontas para uso
- É ainda mais rápida que a Deep Graph Library

Obrigada :)