PDPM
IIITDM
Jabalpur

Introduction to PyTorch

What is PyTorch?

- A library to process tensors in an optimized manner.
- It is an open-source library developed by Facebook AI research team.
- It is mostly used library used for machine learning/deep learning application development, as tensorflow and keras.
- Official Website: https://pytorch.org/

PyTorch Installation

- Go to the official website of PyTorch: https://pytorch.org/
- Select the configurations and get the installation command.
- Installation using Conda
    - conda create –n pytorch_env
    - conda activate pytorch_env
    - conda install jupyter notebook
    - conda install pytorch torchvision -c pytorch
- Installation using pip
    - Pip install torch

Working with PyTorch

- Import the package
- Import the Pytorch
    - import torch
- Checking the version of torch
    - torch.__version__
- Check CUDA availability
    - torch.cuda.is_available()

Basics of Tensor

- Tensors are scalar, vector or n-dimensional arrays.
- Defining a scalar in torch
    - a=torch.tensor(10)
- Checking the datatype
    - a.dtype
- Defining a vector
    - b=torch.tensor([1,2,3,4])
- Defining 2-dimensional array
    - c=torch.tensor([[1,2,3,4],[5,6,7,8],[9,8,7,7]])

Basics of Tensor

- Defining 3-dimensional array
  - c=torch.tensor([[[1,2,3,4],[5,6,7,8],[9,8,7,7]],[[1,2,3,4],[5,6,7,8],[9,8,7,7]]])
- Checking the Shape
  - a.shape
- Checking the size
  - a.size()

PyTorch with Numpy

- From Numpy to Tensor
  - import numpy as np
  - a1 = np.array([[1,2],[3,4]])
  - a2=torch.from_numpy(a1)
  - a2.dtype
- From Tensor to Numpy
  - a3= a2.numpy()
  - a3.dtype

PyTorch Arrays

- Initializing the array using PyTorch
  - x = torch.empty(2,3)
  - y=torch.rand(2,2)
  - x=torch.zeros(2,2)
  - x=torch.ones(2,2)
  - x.dtype

Python Basics

- Initializing the datatype using PyTorch
  - x=torch.ones(2,2, dtype=torch.int)
  - x.dtype
- Arithmetic Operations in PyTorch
  - x= torch.rand(2,2)
  - y=torch.rand(2,2)
  - z= x+y

Arithmetic Operations

- Addition
  - z = torch.add(x,y)
  - z

  OR

  - z = y.add_(x)
  - z

Arithmetic Operations

- Sutraction
  - z = x-y
  - z

OR

  - z=torch.sub(x,y)
  - z

OR

  - z =x.sub_(y)
  - z

Arithmetic Operations

- Multiplication
  - z = x*y
  - z

  OR

  - z=torch.mul(x,y)
  - z

  OR

  - z =x.mul_(y)
  - z

Arithmetic Operations

- Division
  - z = x/y
  - z

OR

  - z=torch.div(x,y)
  - z

OR

  - z =x.div_(y)
  - z

PyTorch Basics

- Array Reshaping
  - y=x.view(4)
  - z=x.view(-1, 4)

Working with CUDA

- if torch.cuda.is_available():
- device = torch.device("cuda")
- x= torch.ones(5,5, device=device)
- #or
- y=torch.ones(5,5)
- y=y.to(device)
- z=x + y
- # A GUP tensor is not convertible to numpy array
- z= z.to("cpu")
- z.numpy()

Working with Gradient

- w=torch.tensor(10.)
- x=torch.tensor(5., requires_grad=True)
- b=torch.tensor(2., requires_grad=True)
- y = w * x + b
- y
- y.backward()
- print("dy/dx:", x.grad)
- print("dy/db:", b.grad)
- print("dy/dw:", w.grad)

Thank You