LG 에너지 솔루션 실습 강의

Pytorch Tutorial

2023. 06. 16

고려대학교 산업경영공학과

Data Mining & Quality Analytics Lab.

임 새 린

Pytorch란?

Pytorch

❖ 파이썬 딥러닝 개발 프레임워크

VS

Pytorch란?

Pytorch

❖ 파이썬 딥러닝 개발 프레임워크

TensorFlow

VS

PyTorch

- 다양한 지원 범위(C언어, 하드웨어 등)
- 강력한 배포 기능(TensorFlow Serving, TensorFlow Lite 등)
- 자동화된 최적화

- 직관적인 인터페이스
- 동적 그래프를 활용한 디버깅 및 커스터마이징
- 활발한 커뮤니티 및 오픈소스

Pytorch란?

Pytorch

❖ 파이썬 딥러닝 개발 프레임워크

TensorFlow

VS

PyTorch

- 복잡하고 추상적인 표현
- 정적 그래프로 인한 유연성 부족
- 커뮤니티 분열 (TensorFlow & TensorFlow 2.0)

- 동적 그래프로 인한 속도와 성능 하락
- 배포 및 서비스화 부족
- 문서화된 튜토리얼이 부족

Pytorch란?

Pytorch

❖ 파이썬 딥러닝 개발 프레임워크

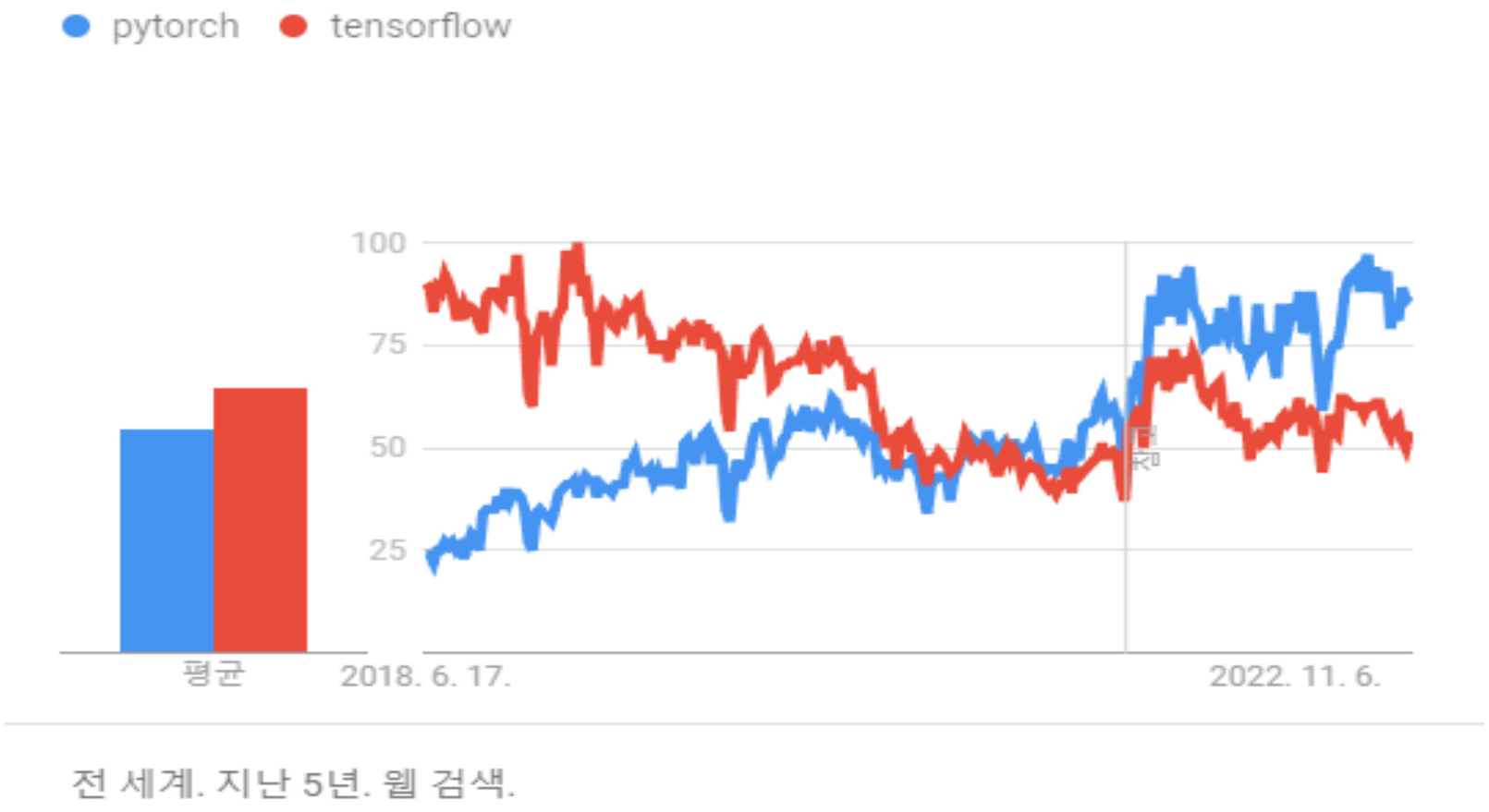
- 연구, 개발을 위해서는 직관성, 쉬운 디버깅, 쉬운 커스터마이징의 특징을 가지는 pytorch가 더 적합함

VS

Pytorch란?

Pytorch

❖ 파이썬 딥러닝 개발 프레임워크



Pytorch Tutorial

Tutorial

❖ 아나콘다 가상환경에 pytorch와 CUDA 설치하기

- 1. anaconda prompt 창에서 새로운 가상환경 생성 및 실행

가상환경 이름 **파이썬 버전**

```
(base) C:₩Users₩korea>conda create -n mytorch python=3.9
Collecting package metadata (current_repodata.json): done
Solving environment: done


==> WARNING: A newer version of conda exists. <==
  current version: 4.12.0
  latest version: 23.5.0
```

```
(base) C:₩Users₩korea>conda activate mytorch

(mytorch) C:₩Users₩korea>
```

Pytorch Tutorial

Tutorial

❖ 아나콘다 가상환경에 pytorch와 CUDA 설치하기

- 2. CUDA Toolkit 및 cuDNN 설치
- 파이썬 버전에 맞는 CUDA toolkit과 cuDNN 설치가 필요

(https://www.tensorflow.org/install/source_windows?hl=ko#tested_build_configurations)

Cudatoolkit 버전

```
(mytorch) C:₩Users₩korea>conda install -c conda-forge cudatoolkit=11.6
```

```
(mytorch) C:₩Users₩korea>conda install -c conda-forge cudnn
```

Pytorch Tutorial

Tutorial

❖ 아나콘다 가상환경에 pytorch와 CUDA 설치하기

- 3. CUDA Toolkit 버전에 알맞은 pytorch 설치 (https://pytorch.org/get-started/previous-versions/)
- 4. GPU를 사용할 수 있는지 확인

```
(mytorch) C:₩Users₩korea>conda install pytorch==1.12.1 torchvision==0.13.1 torchaudio==0.12.1 cudatoolkit=11.6 -c pytorch -c conda-forge
```

```
(mytorch) C:₩Users₩korea>python
Python 3.9.16 (main, May 17 2023, 17:49:16) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import torch
>>> torch.cuda.is_available()
True
>>>
```

Pytorch Tutorial

Tutorial

❖ 아나콘다 가상환경에 pytorch와 CUDA 설치하기

- 5. 가상환경에 주피터 노트북 설치 및 연결하기

```
(mytorch) C:₩Users₩korea>pip install jupyter notebook
```

```
(mytorch) C:₩Users₩korea>python -m ipykernel install --user --name mytorch --display-name mytorch
Installed kernelspec mytorch in C:₩Users₩korea₩AppData₩Roaming₩jupyter₩kernels₩mytorch
```

Pytorch Tutorial

Tutorial

❖ GPU가 없는 경우 가상환경에 pytorch 설치

- CUDA toolkit이나 cuDNN 설치 필요없이 바로 cpu버전의 pytorch 설치하고 가상환경과 주피터 노트북 연결
- https://pytorch.org/get-started/previous-versions/ ← 해당 사이트에 들어가서 "cpu" 검색, 원하는 버전이 나올 때까지 진행

Linux and Windows

```
# CUDA 10.2
conda install pytorch==1.12.1 torchvision==0.13.1 torchaudio==0.12.1 cudatoolkit=10.2 -c pytorch
# CUDA 11.3
conda install pytorch==1.12.1 torchvision==0.13.1 torchaudio==0.12.1 cudatoolkit=11.3 -c pytorch
# CUDA 11.6
conda install pytorch==1.12.1 torchvision==0.13.1 torchaudio==0.12.1 cudatoolkit=11.6 -c pytorch -c conda-fo
# CPU Only
conda install pytorch==1.12.1 torchvision==0.13.1 torchaudio==0.12.1 cpuonly -c pytorch
```

Pytorch Tutorial

Tutorial

❖ 파이토치 자료형

- 자료형 : 변수가 메모리공간에 저장되는 형태(Type)로 pytorch는 tensor라는 독자적인 자료형이 있음

실수 변수는 FloatTensor

Data type	dtype	CPU tensor	GPU tensor
32-bit floating point	`torch.float32` or `torch.float`	`torch.FloatTensor`	`torch.cuda.FloatTensor`
64-bit floating point	`torch.float64` or `torch.double`	`torch.DoubleTensor`	`torch.cuda.DoubleTensor`
32-bit integer (signed)	`torch.int32` or `torch.int`	`torch.IntTensor`	`torch.cuda.IntTensor`
64-bit integer (signed)	`torch.int64` or `torch.long`	`torch.LongTensor`	`torch.cuda.LongTensor`
Boolean	`torch.bool`	`torch.BoolTensor`	`torch.cuda.BoolTensor`

정수 변수는 LongTensor

Pytorch Tutorial

Tutorial

❖ 파이토치 자료형

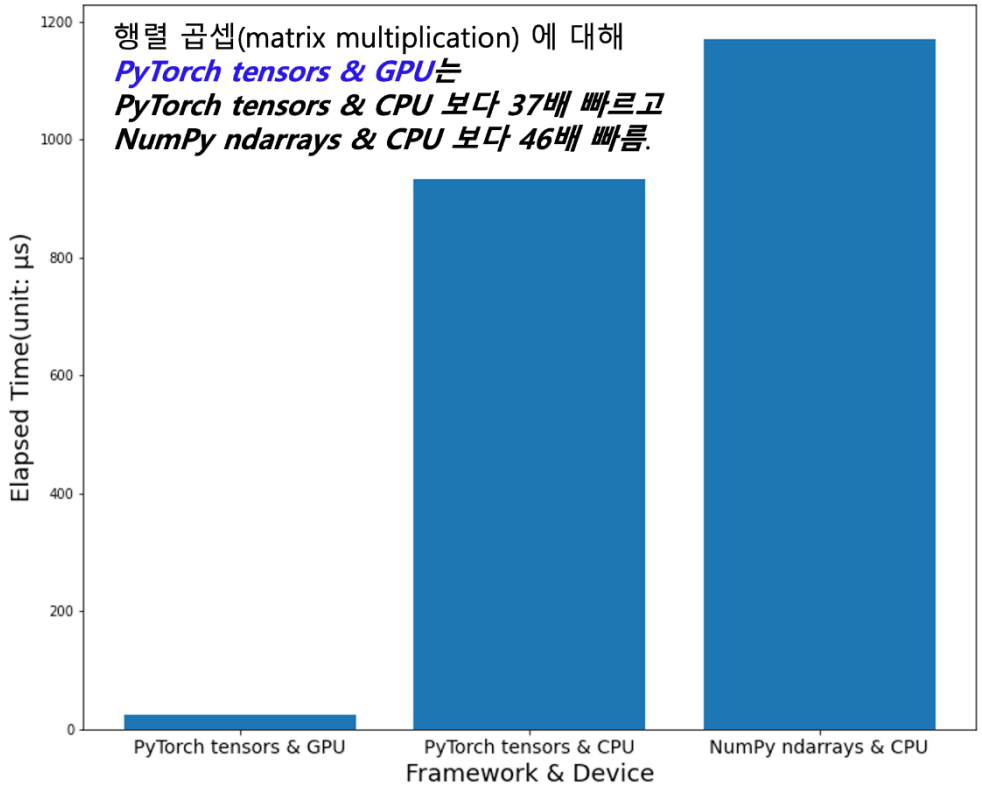
- 자료형 : 변수가 메모리공간에 저장되는 형태(Type)로 pytorch는 tensor라는 독자적인 자료형이 있음

- DatFrame 이 기본 자료형
- 3차원 이상의 데이터를 다루기 어려움

- Ndarray 가 기본 자료형
- GPU 연산이 불가능
- 행렬 연산에 최적화된 패키지가 아님

- Tensor 가 기본 자료형
- GPU 연산 가능
- 행렬 연산에 최적화
- Numpy와 memory를 공유하여 접근성과 효율성이 좋음

PyTorch tensors 대비 NumPy ndarrays 의 행렬 곱셈 성능 비교



Pytorch Tutorial

Tutorial

❖ 파이토치 자동 미분

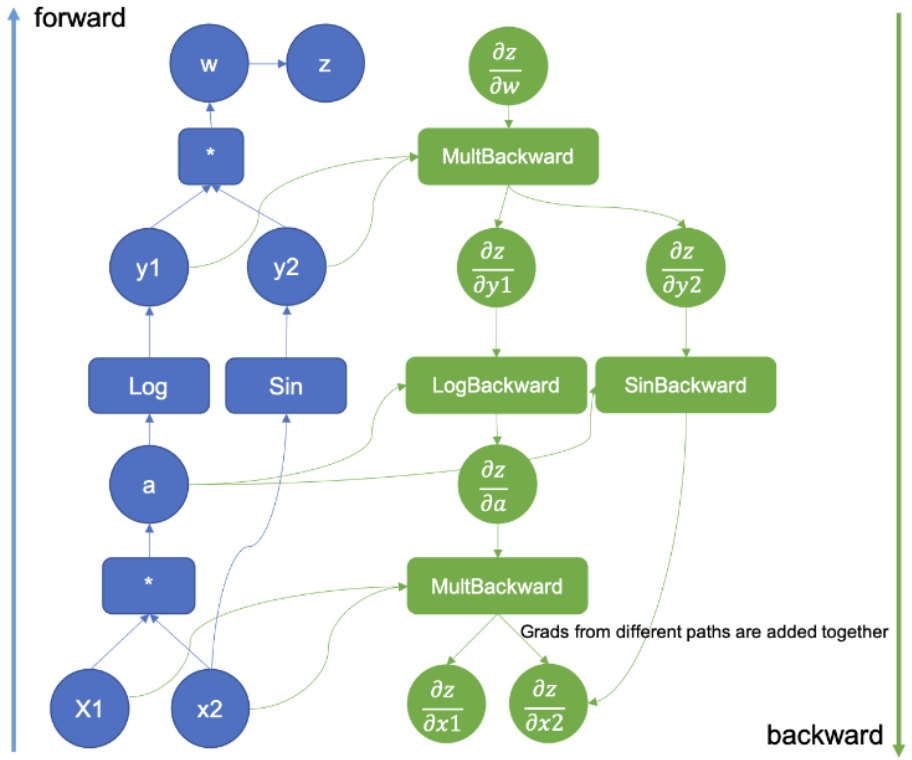
- 파이토치는 tensor의 연산이 진행될 때 grad_fn 함수에 어떤 연산이 이루어 졌는지 저장
- 연산 정보를 저장함으로써 역전파 과정에서 gradient를 쉽게 계산할 수 있음

```
import torch
x = torch.ones(2, 2, requires_grad=True)
y1 = x + 2
print(y1)
# tensor([[3., 3.],
#         [3., 3.]], grad_fn=<AddBackward0>)

y2 = x - 2
print(y2)
# tensor([[-1., -1.],
#         [-1., -1.]], grad_fn=<SubBackward0>)

y3 = x * 2
print(y3)
# tensor([[2., 2.],
#         [2., 2.]], grad_fn=<MulBackward0>)

y4 = x / 2
print(y4)
# tensor([[0.5000, 0.5000],
#         [0.5000, 0.5000]], grad_fn=<DivBackward0>)
```

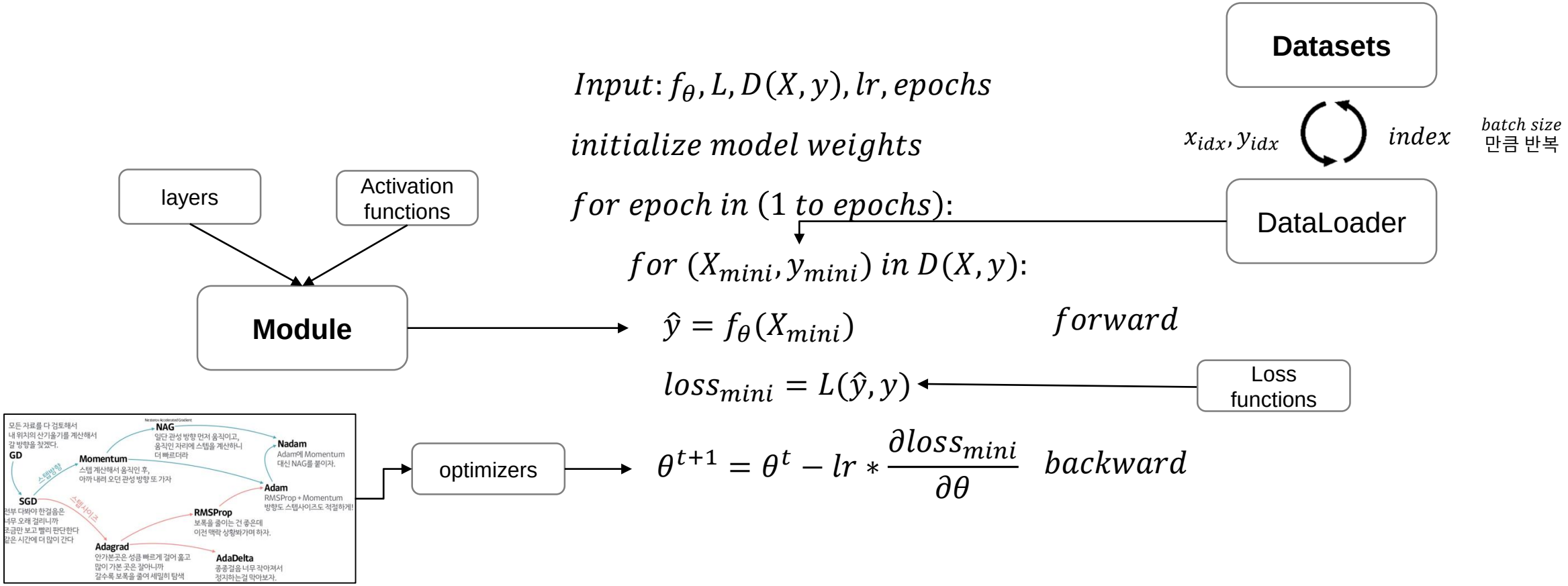


Pytorch Tutorial

Tutorial

❖ 오류 역전파 알고리즘

- 파이토치는 데이터셋, 데이터로더, 모델, 옵티마이저, 손실함수를 통해서 오류 역전파 알고리즘을 구현



감사합니다