

Deep Learning Programming

Lecture 1.2: Pytorch Introduction

Sangryul Jeon

School of Computer Science and Engineering

srjeonn@pusan.ac.kr

PART 3: Data Type & Basic Functions

Slides borrowed from Naver Connect Foundation

1. PyTorch의 데이터 탑

1.1 PyTorch의 데이터 탑

1.2 탑 캐스팅

1.3 학습정리

2. Tensor의 기초 함수 및 메서드

2.1 Tensor의 요소를 반환하거나 계산하는 함수

2.2 Tensor의 특성을 확인하는 메서드

2.3 학습정리

1.

PyTorch의 데이터 탑입

이번 챕터에서는 PyTorch의 데이터 탑입과 탑입 캐스팅에 대해 살펴봅니다.

데이터 타입

- PyTorch에서 데이터 타입(dtype)은 Tensor가 저장하는 값의 데이터 유형을 의미함

정수형 데이터 타입

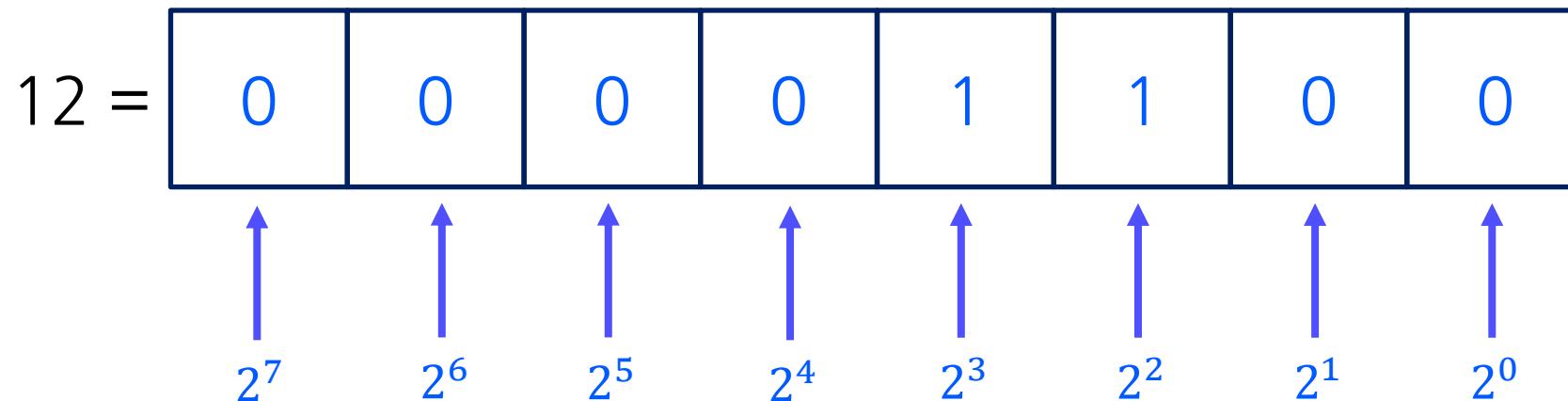
- 정수형
 - 정수형 데이터 타입은 소수 부분이 없는 숫자를 저장하는 데 사용되는 데이터 타입으로 8비트 부호 없는 정수, 8비트 부호 있는 정수, 16비트 부호 있는 정수, 32비트 부호 있는 정수, 64비트 부호 있는 정수 유형으로 구분함

정수형 데이터 타입

- 8비트 **부호 없는** 정수 **언어적 표현**
 - 8개의 이진 자리수(비트)를 사용하여 0부터 255까지의 정수를 표현할 수 있는 데이터 형식

정수형 데이터 타입

- 8비트 부호 없는 정수의 공간에서 표현



정수형 데이터 타입

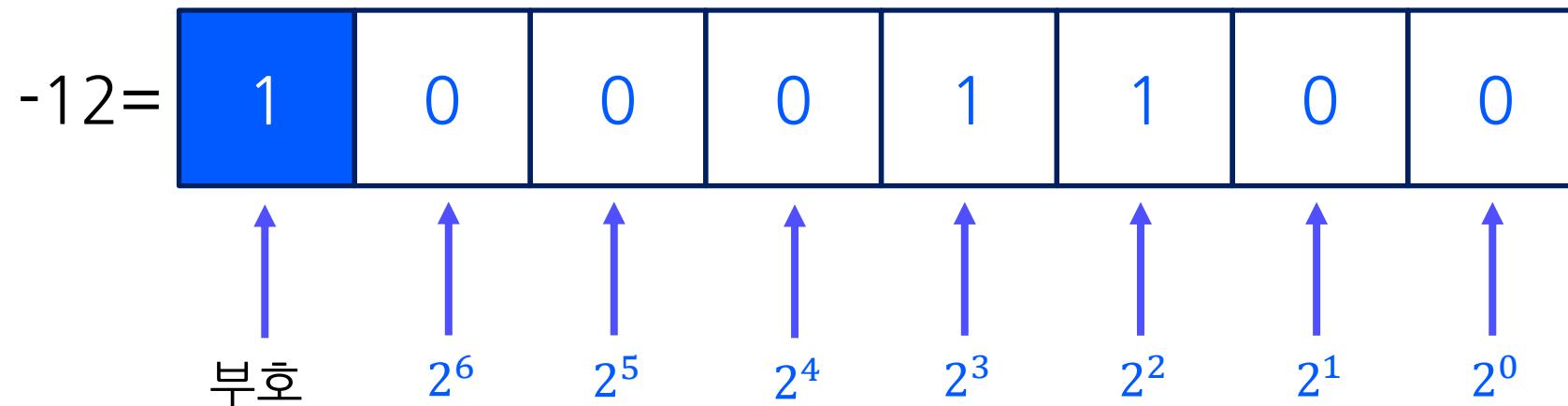
- 8비트 부호 없는 정수의 코드 표현
 - `dtype = torch.uint8`

정수형 데이터 타입

- 8비트 부호 있는 정수의 언어적 표현
 - 8개의 이진 자리수(비트)를 사용하여 -128부터 127까지의 정수를 표현할 수 있는 데이터 형식

정수형 데이터 타입

- 8비트 부호 있는 정수의 공간에서 표현



정수형 데이터 타입

- 8비트 부호 있는 정수의 코드 표현
 - `dtype = torch.int8`

정수형 데이터 타입

- 16비트 **부호 있는** 정수의 언어적 표현
 - 16개의 이진 자리수(비트)를 사용하여 -32,768부터 32,767까지의 정수를 표현할 수 있는 데이터 형식

정수형 데이터 타입

- 16비트 부호 있는 정수의 코드 표현
 - `dtype = torch.int16` 또는 `dtype = torch.short`

정수형 데이터 타입

- 32비트 부호 있는 정수의 언어적 표현
 - 32개의 이진 자리수(비트)를 사용하여 -2,147,483,648부터 2,147,483,647까지의 정수를 표현할 수 있는 데이터 형식
 - 대부분의 프로그래밍에서 표준적인 정수 크기로 사용

정수형 데이터 타입

- 32비트 부호 있는 정수의 코드 표현
 - `dtype = torch.int32` 또는 `dtype = torch.int`

정수형 데이터 타입

- 64비트 부호 있는 정수의 언어적 표현
 - 64개의 이진 자리수(비트)를 사용하여 -9,223,372,036,854,775,808부터 9,223,372,036,854,775,807까지의 정수를 표현할 수 있는 데이터 형식

정수형 데이터 타입

- 64비트 부호 있는 정수의 코드 표현
 - `dtype = torch.int64` 또는 `dtype = torch.long`

실수형 데이터 타입

- 실수형
 - 실수형 데이터 타입은 32비트 부동 소수점 수와 64비트 부동 소수점 수의 유형 등으로 구분함
 - [실수형 데이터 타입들은 신경망의 수치 계산](#)에서 사용됨

실수형 데이터 타입

- 16비트 고정 소수점 수의 언어적 표현
 - 16개의 이진 자리수(비트)를 사용하여 정수부와 소수부로 표현하는 데이터 형식

실수형 데이터 타입

- 16비트 고정 소수점 수의 공간에서 표현

The diagram shows the binary representation of the decimal number 102.5. It consists of a sequence of 16 boxes arranged horizontally. The first three boxes from the left are labeled "부호" (Sign), the next seven are "정수부" (Integer part), and the last six are "소수부" (Fractional part). The boxes are colored blue and white alternately. In the "정수부" section, the boxes are filled with the binary digits 0, 1, 1, 0, 0, 1, 1. In the "소수부" section, the boxes are filled with the binary digits 0, 0, 0, 0, 0, 1, 0, 1.

실수형 데이터 타입

- 16비트 고정 소수점 수의 공간에서 표현

The diagram shows the binary representation of the floating-point number 102.5. It consists of three main parts: the sign bit (부호), the fraction part (정수부), and the exponent part (소수부). The sign bit is 0, indicating a positive number. The fraction part starts with 102 and ends with .5, which is represented as 0.1 in binary. The exponent part is 7, represented as 0111 in binary.

102.005는 어떻게 저장해야 할까요?

실수형 데이터 타입

- 16비트 고정 소수점 수의 문제점
 - 102.005를 고정 소수점 방식으로 저장하기 위해서는 소수부를 각 자리마다 따로 저장해야 함
 - 다시 말해, 소수부의 각 자리마다 4bit가 필요함(0~9까지 저장이 가능하기 때문에)

$$9 = \boxed{1} \boxed{0} \boxed{0} \boxed{1}$$

- 컴퓨터가 메모리를 감당하기 어려움

실수형 데이터 타입

- 부동 소수점 수의 등장
 - 고정 소수점 수의 방식의 문제점을 해결하기 위해 부동 소수점 수가 등장함
 - 부동 소수점 수는 숫자를 정규화하여 가수부와 지수부로 나누어 표현하는 방식

$$102.5 = 1.025 \times 10^2$$

정규화

가수부

지수부

실수형 데이터 타입

- 32비트 부동 소수점 수의 언어적 표현
 - 32개의 이진 자리수(비트)를 사용하여 가수부와 지수부로 표현하는 데이터 형식

실수형 데이터 타입

- 32비트 부동 소수점 수의 공간에서 표현



부호
1bit

지수부
8bit

가수부
23bit

실수형 데이터 타입

- 32비트 부동 소수점 수의 [코드 표현](#)
 - `dtype = torch.float32` 또는 `dtype = torch.float`

실수형 데이터 타입

- 64비트 부동 소수점 수의 언어적 표현
 - 64개의 이진 자리수(비트)를 사용하여 지수부와 가수부로 표현하는 데이터 형식

실수형 데이터 타입

- 64비트 부동 소수점 수의 공간에서 표현



부호
1bit

지수부
11bit

가수부
52bit

실수형 데이터 타입

- 64비트 부동 소수점 수의 코드 표현
 - `dtype = torch.float64` 또는 `dtype = torch.double`

타입 캐스팅

- PyTorch에서 타입 캐스팅은 한 데이터 타입을 다른 데이터 타입으로 변환하는 것을 의미함

타입 캐스팅의 코드 표현

- `i = torch.tensor([2, 3, 4], dtype = torch.int8)`과 같이 Tensor를 생성했을 때,
- 32비트 부동 소수점 수로 변환하는 코드 표현
 - `j = i.float()`
- 64비트 부동 소수점 수로 변환하는 코드 표현
 - `k = i.double()`

PyTorch의 데이터 타입

- PyTorch에서 데이터 타입(dtype)은 Tensor가 저장하는 값의 데이터 유형을 의미한다.
- PyTorch에서 타입 캐스팅은 한 데이터 타입을 다른 데이터 타입으로 변환하는 것을 의미한다.

2.

Tensor의 기초 함수 및 메서드

이번 챕터에서는 Tensor의 요소를 반환하거나 계산하는 함수, 특성을 확인하는 메서드에 대해 살펴봅니다.

Tensor의 요소를 반환하거나 계산하는 함수

- `min()` 함수의 코드 표현
 - PyTorch의 `min()` 함수는 Tensor의 모든 요소들 중 최소값을 반환하는 함수

2.1 Tensor의 요소를 반환하거나 계산하는 함수

2. Tensor의 기초 함수 및 메서드

Tensor의 요소를 반환하거나 계산하는 함수

- `min()` 함수의 공간에서 표현

1.	2	3
3	4	5

Tensor의 요소를 반환하거나 계산하는 함수

- max() 함수의 언어적 표현
 - PyTorch의 max() 함수는 Tensor의 모든 요소들 중 최대값을 반환하는 함수

2.1 Tensor의 요소를 반환하거나 계산하는 함수

2. Tensor의 기초 함수 및 메서드

Tensor의 요소를 반환하거나 계산하는 함수

- max() 함수의 공간에서 표현

1	2	3
3	4	5.

Tensor의 요소를 반환하거나 계산하는 함수

- sum() 함수의 언어적 표현
 - PyTorch의 sum() 함수는 Tensor의 모든 요소들의 합을 계산하는 함수
- prod() 함수의 언어적 표현
 - PyTorch의 prod() 함수는 Tensor의 모든 요소들의 곱을 계산하는 함수
- mean() 함수의 언어적 표현
 - PyTorch의 mean() 함수는 Tensor의 모든 요소들의 평균을 계산하는 함수
- var() 함수의 언어적 표현
 - PyTorch의 var() 함수는 Tensor의 모든 요소들의 표본분산을 계산하는 함수
- std() 함수의 언어적 표현
 - PyTorch의 std() 함수는 Tensor의 모든 요소들의 표본표준편차를 계산하는 함수

Tensor의 요소를 반환하거나 계산하는 함수

- `I = torch.tensor([[1, 2, 3], [3, 4, 5]])`과 같이 Tensor를 생성했을 때,
- `min()` 함수의 [코드 표현](#)
 - `torch.min(I)`
- `max()` 함수의 [코드 표현](#)
 - `torch.max(I)`

Tensor의 요소를 반환하거나 계산하는 함수

- `l = torch.tensor([[1, 2, 3], [3, 4, 5]])`과 같이 Tensor를 생성했을 때,
 - `sum()` 함수의 [코드 표현](#)
 - `torch.sum(l)`
 - `prod()` 함수의 [코드 표현](#)
 - `torch.prod(l)`

Tensor의 요소를 반환하거나 계산하는 함수

- `l = torch.tensor([[1, 2, 3], [3, 4, 5]])`과 같이 Tensor를 생성했을 때,
 - `mean()` 함수의 [코드 표현](#)
 - `torch.mean(l)`
 - `var()` 함수의 [코드 표현](#)
 - `torch.var(l)`
 - `std()` 함수의 [코드 표현](#)
 - `torch.std(l)`

2.2 Tensor의 특성을 확인하는 메서드

2. Tensor의 기초 함수 및 메서드

Tensor의 특성을 확인하는 메서드의 코드 표현

- Tensor 'l'의 차원의 수를 확인
 - `l.dim()`
- Tensor 'l'의 크기(모양)를 확인
 - `l.size()` 또는 `l.shape` (메서드가 아닌 속성)
- Tensor 'l'에 있는 요소의 총 개수를 확인
 - `l.numel()`

Tensor의 기초 함수 및 메서드

- Tensor의 요소를 반환하거나 계산하는 함수의 코드 표현으로 `torch.min()`, `torch.max()`, `torch.sum()`, `torch.prod()`, `torch.mean()`, `torch.var()`, `torch.std()` 등이 있다.
- Tensor의 특성을 확인하는 메서드의 코드 표현으로 `dim()`, `size()`, `numel()` 등이 있다.

Thank you

Prof. Jeon, Sangryul

Computer Vision Lab.

Pusan National University, Korea

Tel: +82-51-510-2423

Web: <http://sr-jeon.github.io/>

E-mail: srjeonn@pusan.ac.kr