

Deep Learning Programming

Lecture 1.2: Pytorch Introduction

Sangryul Jeon

School of Computer Science and Engineering

srjeonn@pusan.ac.kr

PART 4: Creating Tensors

Slides borrowed from Naver Connect Foundation

1. Tensor의 생성

- 1.1 특정한 값으로 초기화된 Tensor 생성
- 1.2 특정한 값으로 초기화된 Tensor 변환
- 1.3 난수로 초기화된 Tensor 생성
- 1.4 지정된 범위 내에서 초기화된 Tensor 생성
- 1.5 초기화되지 않은 Tensor 생성
- 1.6 list, Numpy 데이터로부터 Tensor 생성
- 1.7 CPU Tensor 생성
- 1.8 Tensor의 복제
- 1.9 CUDA Tensor 생성과 변환
- 1.10 학습정리

1.

Tensor의 생성

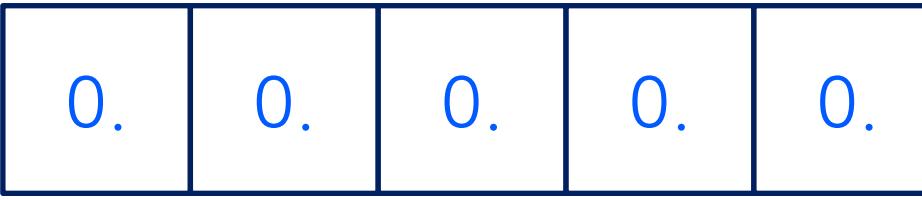
이번 챕터에서는 Tensor를 생성하는 여러가지 방법에 대해 살펴봅니다.

1.1 특정한 값으로 초기화된 Tensor 생성

1. Tensor의 생성

0으로 초기화된 Tensor를 생성하는 표현

- 길이가 5인 0으로 초기화된 1-D Tensor a를 생성 (언어적 표현)

- 

0. 0. 0. 0. 0.

dim-0

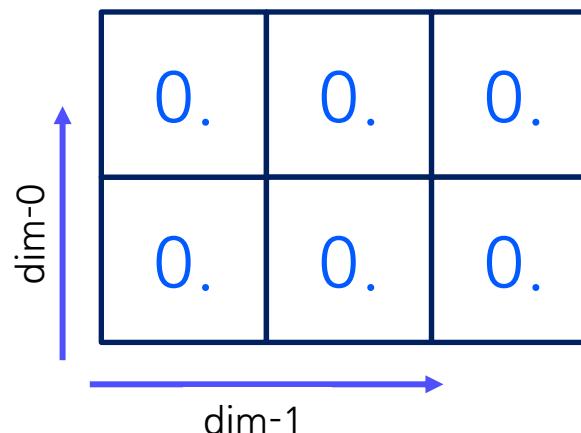
(공간에서 표현)
- `a = torch.zeros(5)` (코드 표현)

1.1 특정한 값으로 초기화된 Tensor 생성

1. Tensor의 생성

0으로 초기화된 Tensor를 생성하는 표현

- 크기가 2x3인 0으로 초기화된 2-D Tensor b를 생성



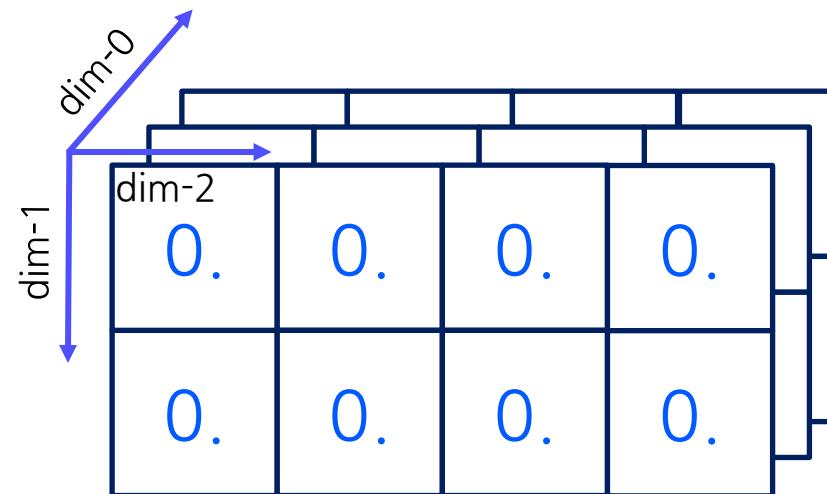
- `b = torch.zeros([2, 3])`

1.1 특정한 값으로 초기화된 Tensor 생성

1. Tensor의 생성

0으로 초기화된 Tensor를 생성하는 표현

- 크기가 3x2x4인 0으로 초기화된 3-D Tensor c를 생성



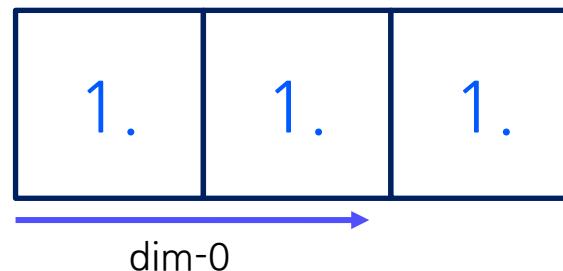
- `c = torch.zeros([3, 2, 4])`

1.1 특정한 값으로 초기화된 Tensor 생성

1. Tensor의 생성

1로 초기화된 Tensor를 생성하는 표현

- 길이가 3인 1로 초기화된 1-D Tensor d를 생성



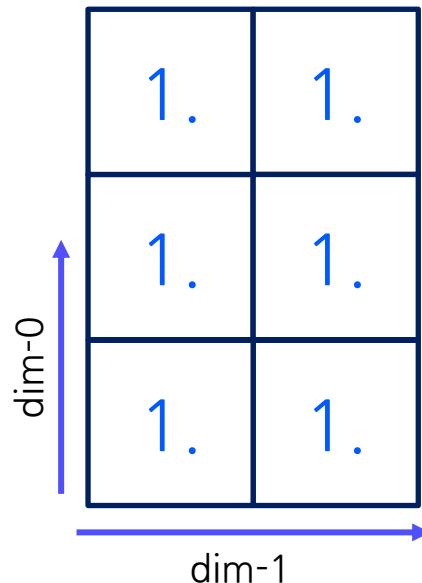
- `d = torch.ones(3)`

1.1 특정한 값으로 초기화된 Tensor 생성

1. Tensor의 생성

1로 초기화된 Tensor를 생성하는 표현

- 크기가 3x2인 1로 초기화된 2-D Tensor e를 생성



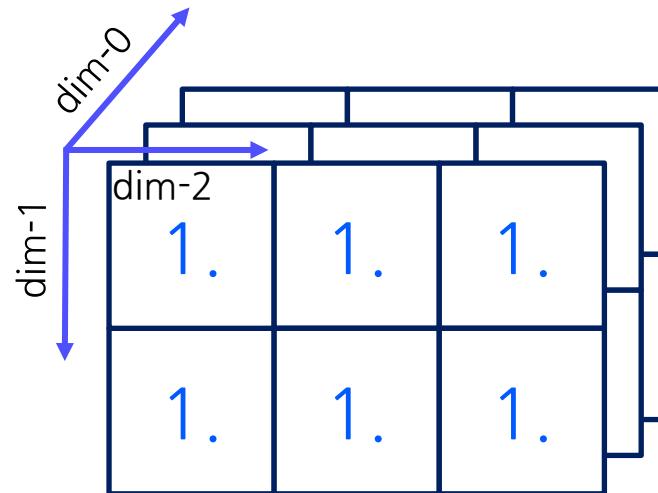
- `e = torch.ones([3, 2])`

1.1 특정한 값으로 초기화된 Tensor 생성

1. Tensor의 생성

1로 초기화된 Tensor를 생성하는 표현

- 크기가 3x2x3인 1로 초기화된 3-D Tensor f를 생성



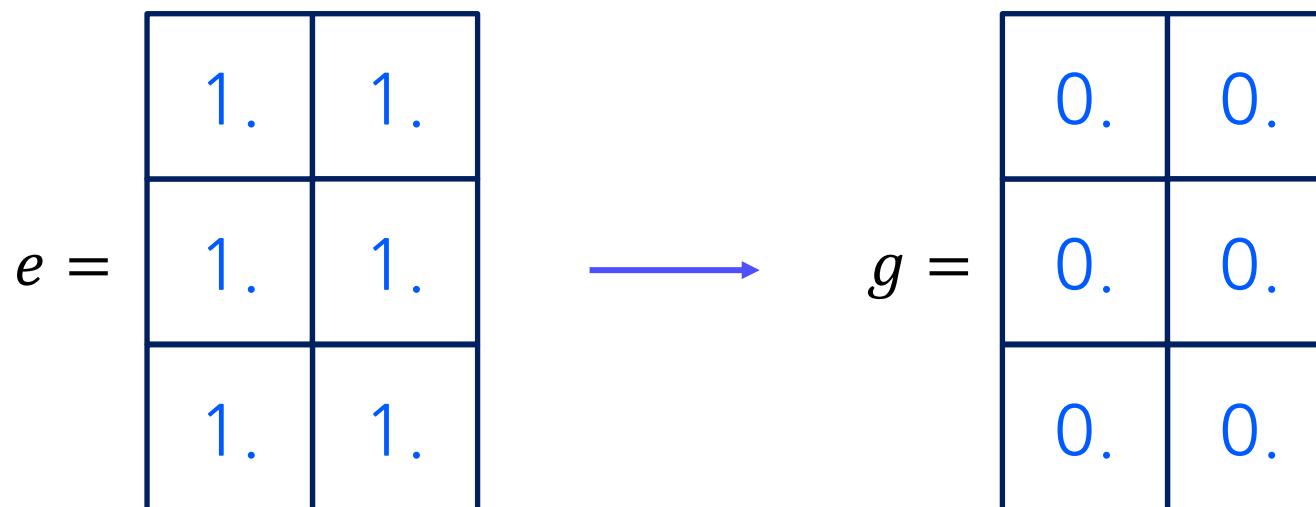
- `f = torch.ones([3, 2, 3])`

1.2 특정한 값으로 초기화된 Tensor 변환

1. Tensor의 생성

크기와 자료형이 같은 0으로 초기화된 Tensor로 변환하는 표현

- e와 크기와 자료형이 같은 0으로 초기화된 Tensor g로 변환



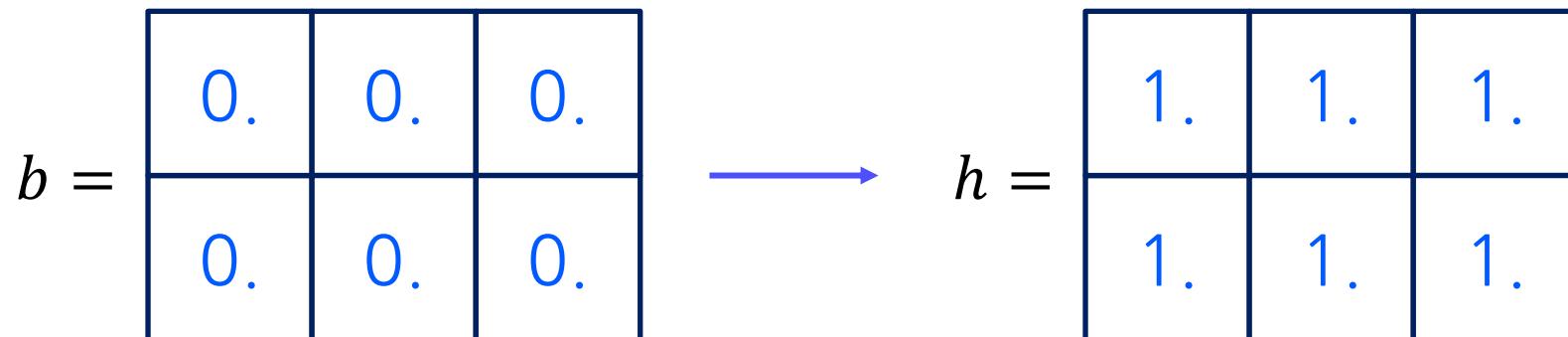
- $g = \text{torch.zeros_like}(e)$

1.2 특정한 값으로 초기화된 Tensor 변환

1. Tensor의 생성

크기와 자료형이 같은 1로 초기화된 Tensor로 변환하는 표현

- b와 크기와 자료형이 같은 1로 초기화된 Tensor h로 변환



- $h = \text{torch.ones_like}(b)$

1.3 난수로 초기화된 Tensor 생성

1. Tensor의 생성

[0, 1] 구간의 연속균등분포 난수 Tensor 생성

- 0~1 사이의 연속균등분포에서 추출한 난수로 채워진 특정 크기의 Tensor를 생성하는 [코드 표현](#)
 - `i = torch.rand(3)`
 - `j = torch.rand([2, 3])`

여기서 [0, 1] 구간의 연속균등분포란 무엇일까?

1.3 난수로 초기화된 Tensor 생성

1. Tensor의 생성

[0, 1] 구간의 연속균등분포란?

- [0, 1] 구간의 연속균등분포의 언어적 표현
 - 연속균등분포란 특정한 두 경계값 사이의 모든 값에 대해 동일한 확률을 가지는 확률분포
 - [0, 1] 구간의 연속균등분포에서 특정한 두 경계값은 0과 1

1.3 난수로 초기화된 Tensor 생성

1. Tensor의 생성

[0, 1] 구간의 연속균등분포란?

- [0, 1] 구간의 연속균등분포의 수식 표현
 - 연속균등분포의 확률밀도함수식

- $f(x) = \frac{1}{b-a} \quad (a \leq x \leq b)$

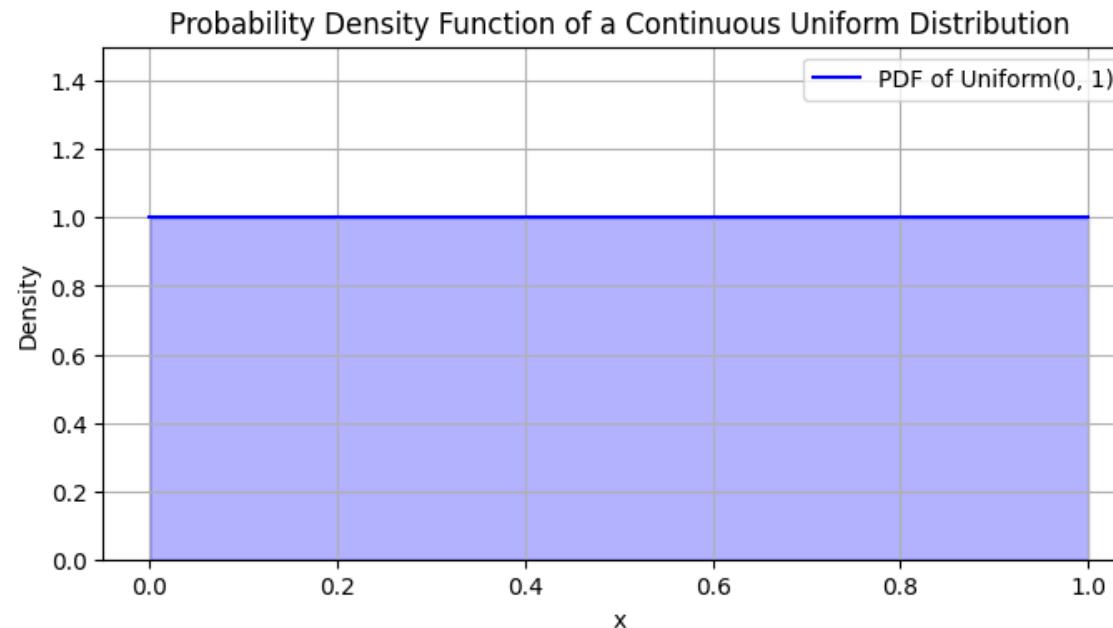
- 연속균등분포의 모수
 - . 평균(기대값): $\mu = \frac{a+b}{2}$
 - . 분산: $\sigma^2 = \frac{(a-b)^2}{12}$
 - . 표준편차: $\sigma = \sqrt{\frac{(a-b)^2}{12}} = \frac{a-b}{2\sqrt{3}}$

1.3 난수로 초기화된 Tensor 생성

1. Tensor의 생성

[0, 1] 구간의 연속균등분포란?

- [0, 1] 구간의 연속균등분포의 확률밀도함수 [그래프 표현](#)

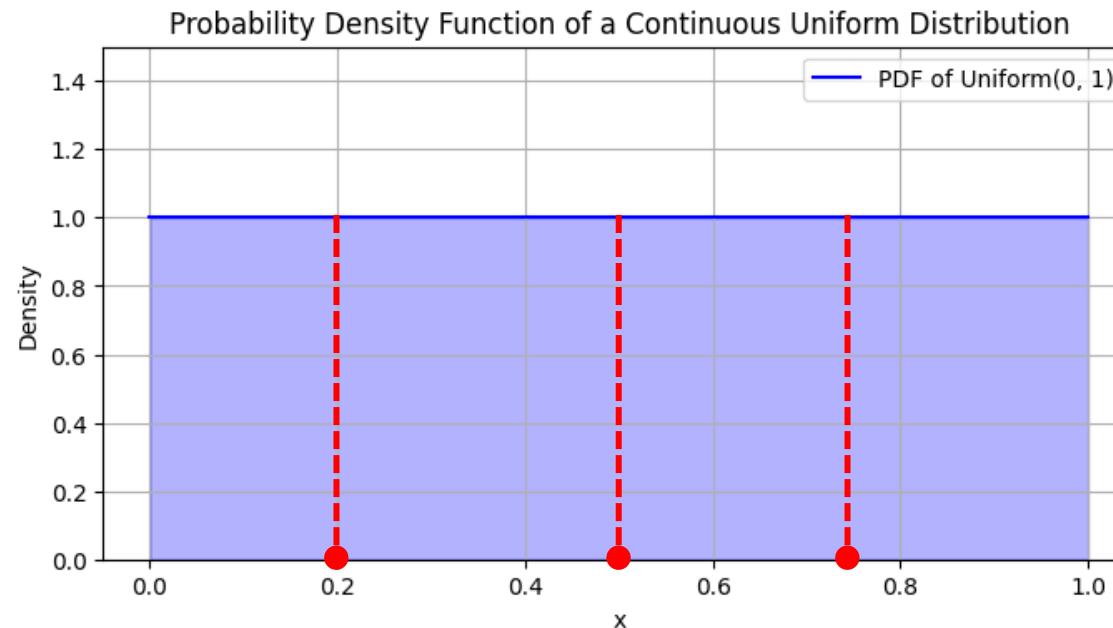


1.3 난수로 초기화된 Tensor 생성

1. Tensor의 생성

[0, 1] 구간의 연속균등분포란?

- [0, 1] 구간의 연속균등분포의 확률밀도함수 [그래프 표현](#)



1.3 난수로 초기화된 Tensor 생성

1. Tensor의 생성

표준정규분포 난수 Tensor 생성

- 표준정규분포에서 추출한 난수로 채워진 특정 크기의 Tensor를 생성하는 [코드 표현](#)
 - `k = torch.randn(3)`
 - `l = torch.randn([2, 3])`

표준정규분포란 무엇일까?

1.3 난수로 초기화된 Tensor 생성

1. Tensor의 생성

표준정규분포란?

- 표준정규분포란 평균이 0이고 표준편차가 1인 종 모양의 곡선
- 표준정규분포는 평균 0을 중심으로 좌우 대칭인 종모양을 가지기 때문에,
평균, 중앙값, 최빈값이 모두 0임

1.3 난수로 초기화된 Tensor 생성

1. Tensor의 생성

표준정규분포란?

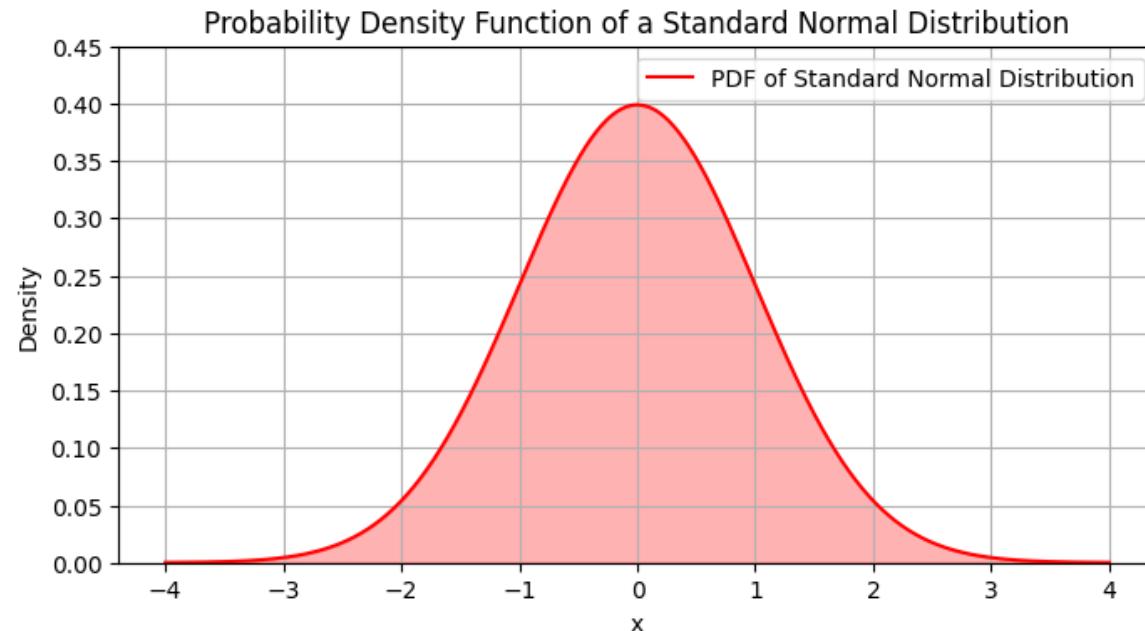
- 표준정규분포의 수학적 표현
 - 표준정규분포의 확률밀도함수식
 - $f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}}$
 - 표준정규분포의 모수
 - 평균(기대값): $\mu = 0$
 - 분산: $\sigma^2 = 1$
 - 표준편차: $\sigma = 1$

1.3 난수로 초기화된 Tensor 생성

1. Tensor의 생성

표준정규분포란?

- 표준정규분포의 확률밀도함수 [그래프 표현](#)

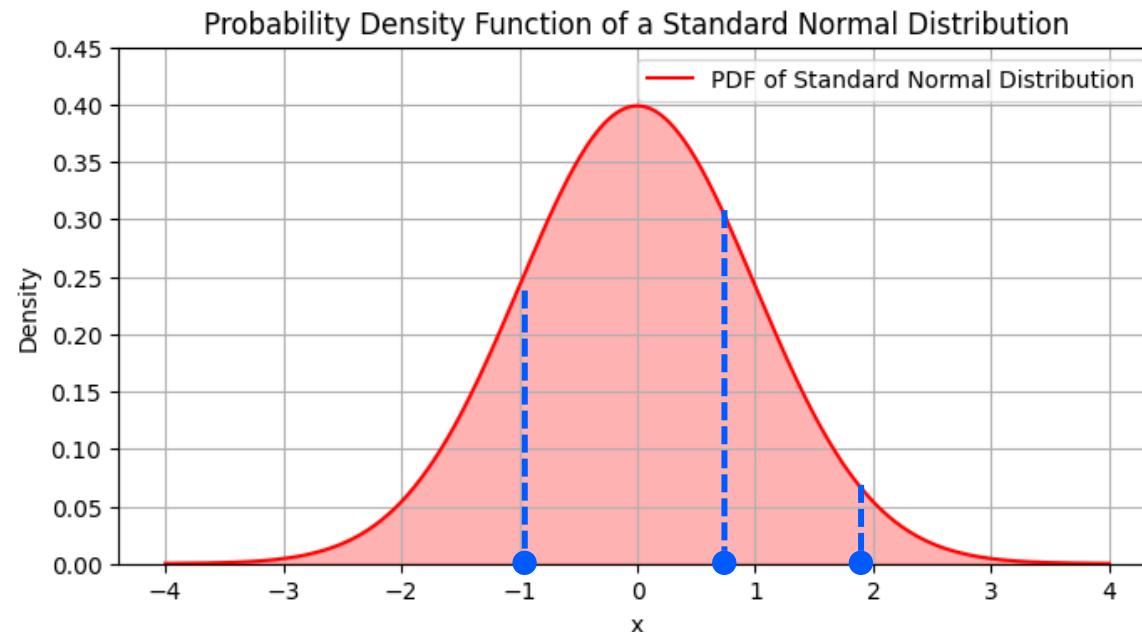


1.3 난수로 초기화된 Tensor 생성

1. Tensor의 생성

표준정규분포란?

- 표준정규분포의 확률밀도함수 [그래프 표현](#)

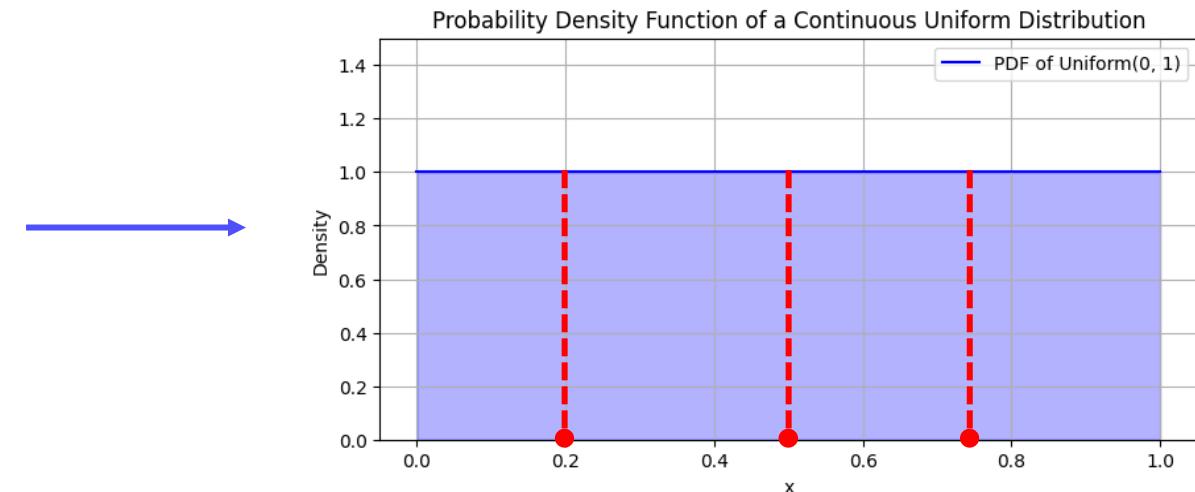
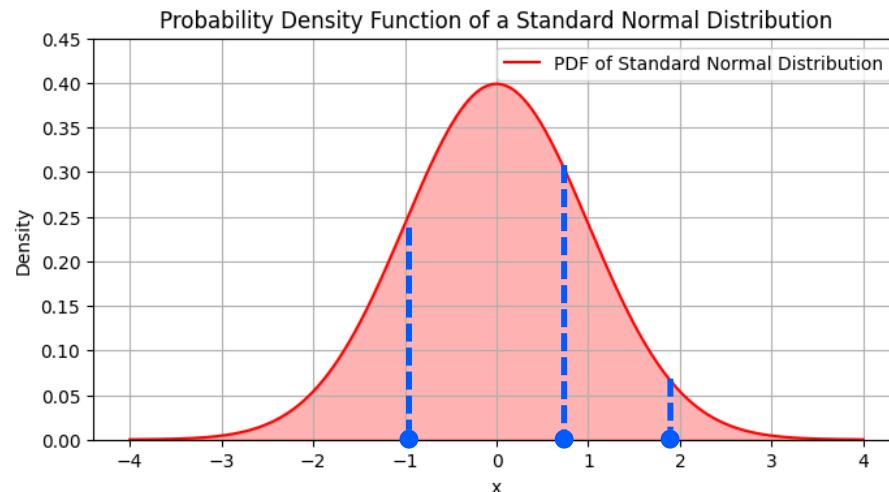


1.3 난수로 초기화된 Tensor 생성

1. Tensor의 생성

크기와 자료형이 같은 [0, 1] 구간의 연속균등분포 난수 Tensor로 변환

- k와 크기와 자료형이 같은 [0, 1] 구간의 연속균등분포 난수 Tensor m으로 변환하는 코드 표현
 - `m = torch.rand_like(k)`

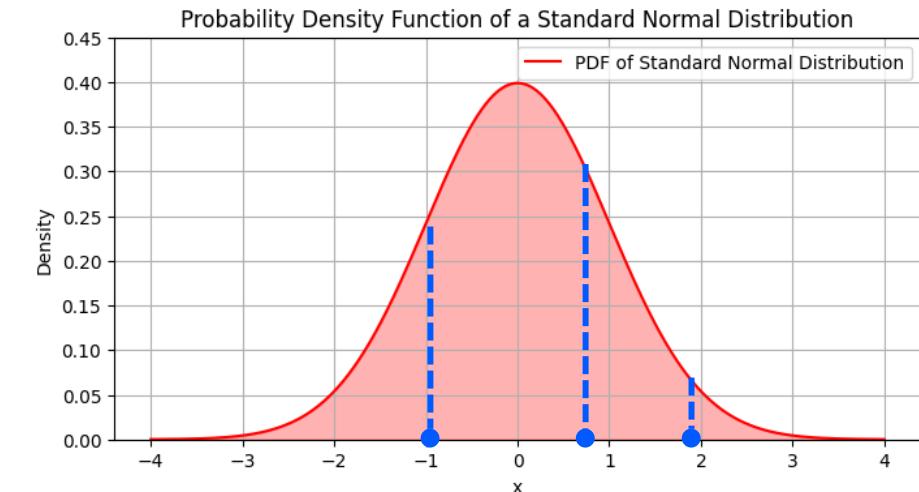
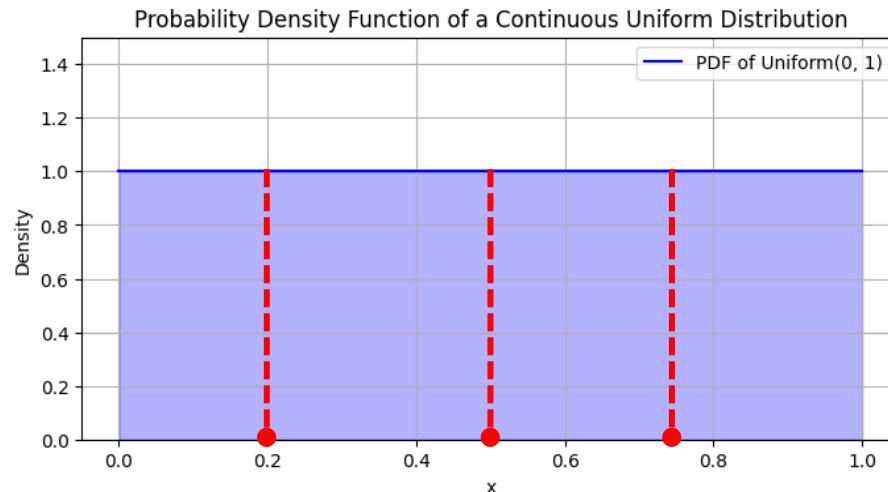


1.3 난수로 초기화된 Tensor 생성

1. Tensor의 생성

크기와 자료형이 같은 표준정규분포 난수 Tensor로 변환

- i와 크기와 자료형이 같은 표준정규분포 난수 Tensor n으로 변환하는 [코드 표현](#)
 - `n = torch.randn_like(i)`



1.4 지정된 범위 내에서 초기화된 Tensor 생성

1. Tensor의 생성

지정된 범위 내에서 일정 간격의 값을 가진 Tensor를 생성하는 표현

- 1에서 11(포함하지 않음)까지 2씩 증가하는 1-D Tensor

1	3	5	7	9
---	---	---	---	---

- `o = torch.arange(start = 1, end = 11, step = 2)`

1.4 지정된 범위 내에서 초기화된 Tensor 생성

1. Tensor의 생성

지정된 범위 내에서 일정 간격의 값을 가진 Tensor를 생성하는 표현

- 1에서 4(포함하지 않음)까지 0.5씩 증가하는 1-D Tensor

1.	1.5	2.	2.5	3.	3.5
----	-----	----	-----	----	-----

- `o = torch.arange(start = 1, end = 4, step = 0.5)`

1.5 초기화 되지 않은 Tensor 생성

1. Tensor의 생성

초기화 되지 않은 Tensor

- ‘초기화 되지 않았다’는 것은 생성된 Tensor의 각 요소가 명시적으로 0, 1, 2 등과 같은 다른 특정 값으로 설정되지 않았음을 의미함
- 초기화 되지 않은 Tensor를 생성할 때, 해당 Tensor는 메모리에 이미 존재하는 임의의 값들로 채워짐

1.5 초기화 되지 않은 Tensor 생성

1. Tensor의 생성

초기화 되지 않은 Tensor를 사용하는 이유

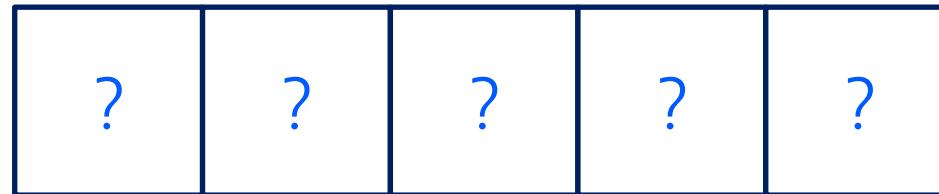
- **성능 향상**: Tensor를 생성하고 곧바로 다른 값들로 덮어쓸 예정인 경우라면, 초기 값을 설정하는 단계는 불필요한 자원을 소모하는 것임
- **메모리 사용 최적화**: 큰 Tensor를 다룰 때, 불필요한 초기화는 메모리 사용량을 증가시킴
초기화되지 않은 Tensor를 사용함으로써 메모리 할당 후 즉시 필요한 계산에 사용하여 메모리 효율성을 높일 수 있음

1.5 초기화 되지 않은 Tensor 생성

1. Tensor의 생성

초기화 되지 않은 Tensor를 생성하는 표현

- `q = torch.empty(5)`



1.5 초기화 되지 않은 Tensor 생성

1. Tensor의 생성

초기화 되지 않은 Tensor에 다른 데이터로 수정하는 표현

- `q.fill_(3.0)`

3.	3.	3.	3.	3.
----	----	----	----	----

- 이 때, 메모리 주소는 변경되지 않음

list 데이터로부터 Tensor를 생성하는 코드 표현

- Python의 list는 여러 값을 순차적으로 저장할 수 있는 가변적인 컨테이너 데이터타입
- list는 대괄호 '[]' 안에 쉼표로 구분한 요소들을 채워 생성할 수 있음
- $s = [1, 2, 3, 4, 5, 6]$ 의 list를 생성했을 때, s를 Tensor t로 생성하는 [코드 표현](#)
 - `t = torch.tensor(s)`

Numpy 데이터로부터 Tensor를 생성하는 코드 표현

- Numpy는 C언어로 구현된 Python 핵심 과학 컴퓨팅 라이브러리
 - 대규모 다차원 배열 연산을 지원
 - 배열을 효율적으로 조작할 수 있는 높은 수준의 수학 함수를 제공
- Numpy와 list의 코드 표현은 외형적으로 비슷해 보이나, Numpy는 대규모 수치 데이터 연산 또는 조작에 있어 적합한 반면, list는 그렇지 못함
 - `u = np.array([[0, 1], [2, 3]]) / u = [[0, 1], [2, 3]]`

Numpy 데이터로부터 Tensor를 생성하는 코드 표현

- u = [[0, 1], [2, 3]]의 Numpy 2-D Matrix를 생성했을 때, u를 Tensor v로 생성하는 [코드 표현](#)
 - `v = torch.from_Numpy(u)`
- Numpy로부터 생성된 Tensor는 기본적으로 정수형이므로 실수형으로 타입 캐스팅이 필요함
 - `v = torch.from_Numpy(u).float()`

1.7 CPU Tensor 생성

1. Tensor의 생성

정수형 CPU Tensor 생성하는 표현

- w = torch.IntTensor([1, 2, 3, 4, 5])

1	2	3	4	5
---	---	---	---	---

1.7 CPU Tensor 생성

1. Tensor의 생성

정수형 CPU Tensor 생성하는 표현

- $A, B, C, D, E = 1, 2, 3, 4, 5$
`w = torch.IntTensor([A, B, C, D, E])`

1	2	3	4	5
---	---	---	---	---

1.7 CPU Tensor 생성

1. Tensor의 생성

실수형 CPU Tensor 생성하는 표현

- $A, B, C, D, E = 1, 2, 3, 4, 5$
- $x = \text{torch.FloatTensor}([A, B, C, D, E])$

1.	2.	3.	4.	5.
----	----	----	----	----

기타 정수형, 실수형 CPU Tensor를 생성하는 코드 표현

- [torch.ByteTensor](#): 8비트 부호 없는 정수형 CPU Tensor 생성
- [torch.CharTensor](#): 8비트 부호 있는 정수형 CPU Tensor 생성
- [torch.ShortTensor](#): 16비트 부호 있는 정수형 CPU Tensor 생성
- [torch.LongTensor](#): 64비트 부호 있는 정수형 CPU Tensor 생성
- [torch.DoubleTensor](#): 64비트 부호 있는 실수형 CPU Tensor 생성

1.8 Tenor의 복제

1. Tensor의 생성

Tensor를 복제하는 코드 표현

- `x = torch.tensor([1, 2, 3, 4, 5, 6])`와 같이 Tensor를 생성했을 때,

```
y = x.clone()
```

또 다른 Tensor를 복제하는 코드 표현

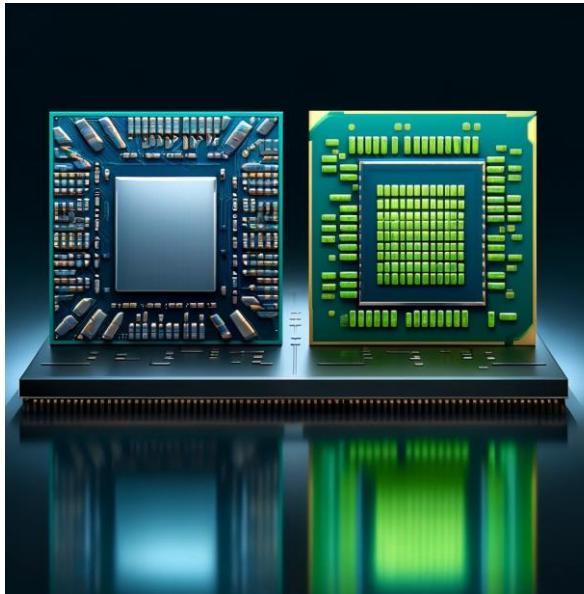
- `x = torch.tensor([1, 2, 3, 4, 5, 6])`와 같이 Tensor를 생성했을 때,
`z = x.detach()`
- `x.clone()` 메서드와 다른 점은 `x`를 계산그래프에서 분리하여
새로운 Tensor `z`에 저장한다는 것

GPU란

- GPU(Graphics Processing Unit)란 **그래픽 처리 장치**를 가리킴
- 원래는 컴퓨터 그래픽스를 렌더링하는데 사용되도록 설계되었음
 - 컴퓨터 그래픽스 렌더링이란 컴퓨터를 사용하여 이미지를 생성하는 과정
- 현재는 AI 연구 및 개발에 있어 대규모 데이터 처리와 복잡한 계산을 위해 사용됨

AI 분야에서 GPU를 사용해야하는 이유

- **병렬 처리 능력:** GPU는 수 천 개의 작은 코어를 가지고 있어 병렬 데이터 처리에 매우 효율적임. 즉, 대량의 연산을 동시에 수행할 수 있게 함



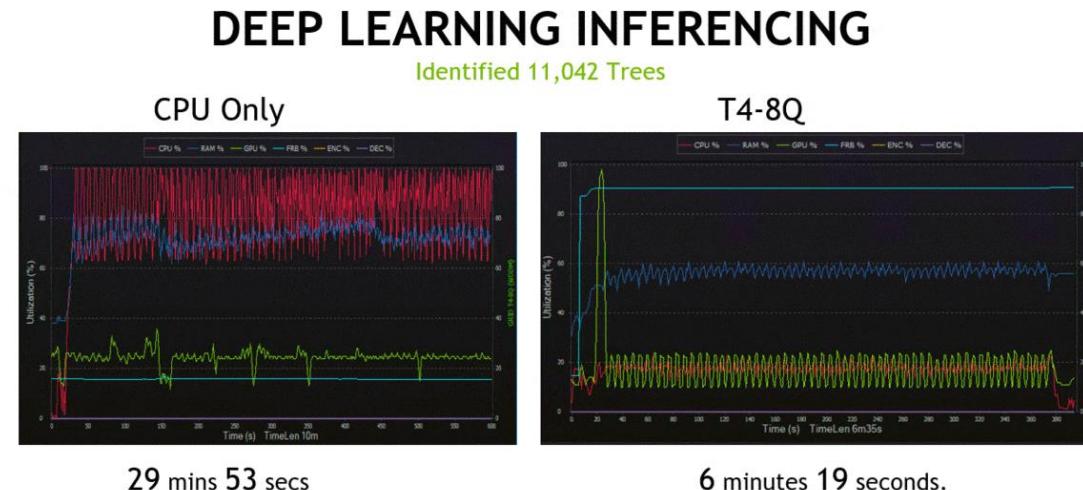
출처: OpenAI (2024). ChatGPT (ChatGPT 4o) [Large language model]. <https://chat.openai.com>

1.9 CUDA Tenor 생성과 변환

1. Tensor의 생성

AI 분야에서 GPU를 사용해야하는 이유

- 속도: GPU의 병렬 처리 능력은 AI 모델의 훈련과 추론 속도를 크게 향상시킴



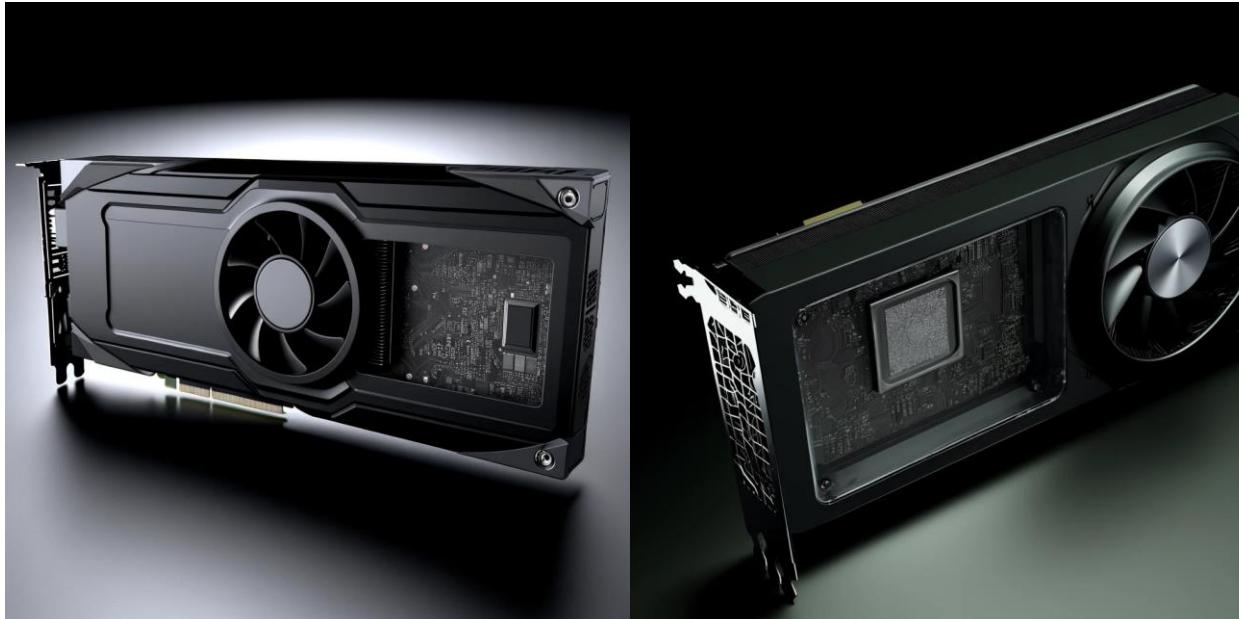
출처: <https://www.esri.com/arcgis-blog/products/arcgis-pro/announcements/arcgis-pro-leveraging-nvidia-vgpu/>

1.9 CUDA Tensor 생성과 변환

1. Tensor의 생성

AI 분야에서 GPU를 사용해야하는 이유

- **경제적 이점**: 초기에는 높은 비용이 들 수 있으나, GPU를 사용하면 시간 단축과 에너지 절약으로 인해 장기적인 투자 대비 높은 수익을 얻을 수 있음



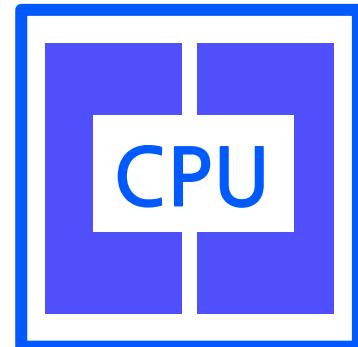
출처: OpenAI (2024). ChatGPT (ChatGPT 4o) [Large language model]. <https://chat.openai.com>

1.9 CUDA Tensor 생성과 변환

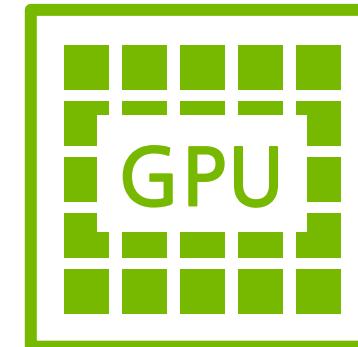
1. Tensor의 생성

디바이스를 확인하기

- Tensor가 현재 어떤 디바이스에 있는지 확인하는 [코드 표현](#)
 - `a = torch.tensor([1, 2, 3])`
`a.device`



또는

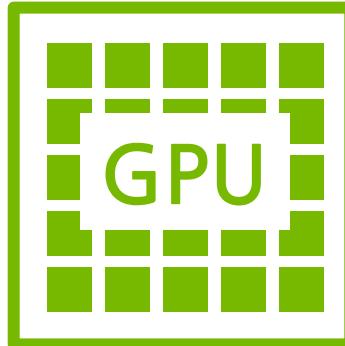


1.9 CUDA Tensor 생성과 변환

1. Tensor의 생성

CUDA 사용가능 환경 확인

- CUDA 기술을 사용할 수 있는 환경인지 확인하는 [코드 표현](#)
 - `torch.cuda.is_available()`



CUDA 이름 확인

- CUDA device 이름을 확인하는 [코드 표현](#)
 - `torch.cuda.get_device_name(device=0)`

Tensor를 GPU에 할당

- Tensor를 GPU에 할당하는 코드 표현
 - `b = torch.tensor([1, 2, 3, 4, 5]).to('cuda')`
 - `b = torch.tensor([1, 2, 3, 4, 5]).cuda()`

1.9 CUDA Tensor 생성과 변환

1. Tensor의 생성

GPU에 할당된 Tensor를 CPU Tensor로 변환

- GPU에 할당된 Tensor를 CPU Tensor로 변환하는 코드 표현
 - `c = b.to(device = 'cpu')`
 - `c = b.cpu()`

Tensor의 생성

- 0, 1과 같은 특정한 값으로 초기화된 Tensor를 생성하는 함수로 `torch.zeros()`, `torch.ones()`가 있다.
- [0, 1] 구간의 연속균등분포, 표준정규분포에서 추출한 난수로 채워진 Tensor를 생성하는 함수로 `torch.rand()`, `torch.randn()`이 있다.
- 지정된 범위 내에서 초기화된 Tensor를 생성하는 함수로 `torch.arange(start, end, step)`가 있다.
- 초기화 되지 않은 Tensor를 생성하는 함수로 `torch.empty()`가 있으며, `fill_()` 메서드로 데이터를 수정할 수 있다.

Tensor의 생성

- Numpy 데이터로부터 Tensor를 생성하는 함수로 `torch.from_Numpy()`가 있다.
- 정수형 및 실수형 CPU Tensor를 생성하는 함수로 `torch.IntTensor()`, `torch.FloatTensor()`가 있다.
- Tensor를 복제하는 메서드로 `clone()`, `detach()`가 있다.
- Tensor를 GPU에 할당하는 메서드로 `to('cuda')`, `cuda()`가 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