



딥러닝 이해2

데이터 표현

최 석 재

lingua@naver.com

Tensorflow's Data Type

Tensor

- Tensor는 데이터를 담는 자료형을 의미한다

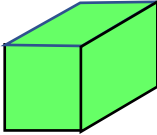
1D TENSOR, VECTOR

12
3
6
14
7

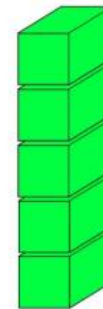
2D TENSOR, MATRIX

12	3	6	14	7
10	5	8	13	8
11	6	7	12	7

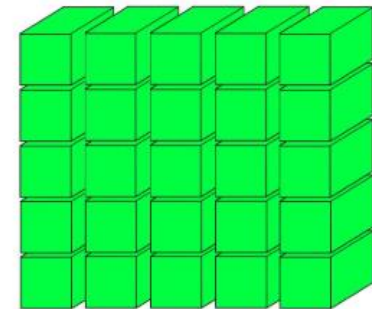
3D TENSOR, CUBE



12	3	6	14	7
10	5	8	13	8
11	6	7	12	7



4D TENSOR, VECTOR OF CUBES



5D TENSOR, MATRIX OF CUBES

Scala (0D Tensor)

- 하나의 숫자만 담고 있는 텐서는 축(axis, dimension)이 없으므로 0D tensor이다
 - Numpy에서는 float32, float64 타입의 숫자가 Scala Tensor로 사용된다
 - 축의 개수는 ndim 속성을 이용하여 알 수 있다
-
- `import numpy as np`
 - `x = np.array(12)`
-
- `print(x)`
 - `print(type(x))`
 - `print(x.ndim)`

```
12
<class 'numpy.ndarray'>
0
```

Vector (1D Tensor)

- 다음과 같이 하나의 축으로 된 배열을 vector 또는 1D tensor라고 부른다

- `x = np.array([12, 3, 6, 14, 7])` 이 숫자열 전체가 한 대상의 특징을 설명한다

- `print(x)`
 - `print(type(x))`
 - `print(x.ndim)`
- ```
[12 3 6 14 7]
<class 'numpy.ndarray'>
1
```

1D TENSOR, VECTOR

|    |
|----|
| 12 |
| 3  |
| 6  |
| 14 |
| 7  |

# Matrix (2D Tensor)

- 벡터가 2개의 축으로 결합된 것이 행렬이다
- Row(행)와 Column(열)으로 구성된다

- `x = np.array([[12, 3, 6, 14, 7],  
[10, 5, 8, 13, 8],  
[11, 6, 7, 12, 7]])`

이 숫자열 전체가 한 대상의 특징을 설명한다

2D TENSOR, MATRIX

|    |   |   |    |   |
|----|---|---|----|---|
| 12 | 3 | 6 | 14 | 7 |
| 10 | 5 | 8 | 13 | 8 |
| 11 | 6 | 7 | 12 | 7 |

- `print(x)`  
[[12 3 6 14 7]  
[10 5 8 13 8]  
[11 6 7 12 7]]
- `print(type(x))`  
<class 'numpy.ndarray'>
- `print(x.ndim)`  
2

# 3D & High-Dimensional Tensor

- 행렬을 하나의 새로운 배열로 합치면 큐브로 해석될 수 있는 3D 텐서가 만들어진다
- 이와 같은 식으로 딥러닝에서는 4D 텐서까지 만들어 사용한다
- 동영상 데이터는 5D 텐서까지 가기도 한다

```
• x = np.array([[[12, 3, 6, 14, 7],
 [10, 5, 8, 13, 8],
 [11, 6, 7, 12, 7]],
 [[12, 3, 6, 14, 7],
 [10, 5, 8, 13, 8],
 [11, 6, 7, 12, 7]],
 [[12, 3, 6, 14, 7],
 [10, 5, 8, 13, 8],
 [11, 6, 7, 12, 7]]])
```

층으로 된 숫자열 전체가 한 대상의 특징을 설명한다

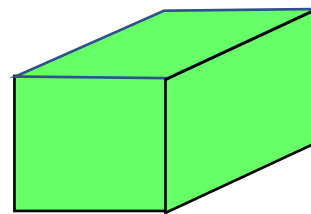
```
[[[12 3 6 14 7]
 [10 5 8 13 8]
 [11 6 7 12 7]]
```

```
[[12 3 6 14 7]
 [10 5 8 13 8]
 [11 6 7 12 7]]
```

```
[[12 3 6 14 7]
 [10 5 8 13 8]
 [11 6 7 12 7]]
```

```
<class 'numpy.ndarray'>
```

3



3D TENSOR, CUBE

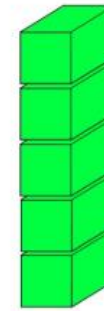
|    |   |   |    |   |
|----|---|---|----|---|
| 12 | 3 | 6 | 14 | 7 |
| 10 | 5 | 8 | 13 | 8 |
| 11 | 6 | 7 | 12 | 7 |

- print(x)
- print(type(x))
- print(x.ndim)

# 연습문제

- 4D 텐서를 만들어 보시오

※ print(x.ndim) 하였을 때 4가 나와야 합니다  
- 다음 슬라이드에 정답이 있습니다



4D TENSOR, VECTOR OF CUBES



# 연습문제 정답

```
import numpy as np
x = np.array([[[[12, 3, 6, 14, 7],
 [10, 5, 8, 13, 8],
 [11, 6, 7, 12, 7]],
 [[12, 3, 6, 14, 7],
 [10, 5, 8, 13, 8],
 [11, 6, 7, 12, 7]],
 [[12, 3, 6, 14, 7],
 [10, 5, 8, 13, 8],
 [11, 6, 7, 12, 7]]],
 [[12, 3, 6, 14, 7],
 [10, 5, 8, 13, 8],
 [11, 6, 7, 12, 7]],
 [[12, 3, 6, 14, 7],
 [10, 5, 8, 13, 8],
 [11, 6, 7, 12, 7]],
 [[12, 3, 6, 14, 7],
 [10, 5, 8, 13, 8],
 [11, 6, 7, 12, 7]]]])

print(x)
print(x.ndim)
```

이 층의 층으로 된 숫자열 전체가 한 대상의 특징을 설명한다

- | 1D | 2D | 3D | 4D | 5D |
|----|----|----|----|----|
| 12 | 3  | 6  | 14 | 7  |
| 10 | 5  | 8  | 13 | 8  |
| 11 | 6  | 7  | 12 | 7  |

|   | Sepal.Length | Sepal.Width | Petal.Length | Petal.Width | Species |
|---|--------------|-------------|--------------|-------------|---------|
| 1 | 5.1          | 3.5         | 1.4          | 0.2         | setosa  |
| 2 | 4.9          | 3.0         | 1.4          | 0.2         | setosa  |
| 3 | 4.7          | 3.2         | 1.3          | 0.2         | setosa  |
| 4 | 4.6          | 3.1         | 1.5          | 0.2         | setosa  |
| 5 | 5.0          | 3.6         | 1.4          | 0.2         | setosa  |
| 6 | 5.4          | 3.9         | 1.7          | 0.4         | setosa  |
| 7 | 4.6          | 3.4         | 1.4          | 0.3         | setosa  |

Diagram illustrating the concept of dimensionality in data. It shows a 3D stack of 3x5 grids. The top grid is labeled '1D' in red. The stack is labeled '2D' in red. The entire stack is labeled '3D' in red.

|    |   |   |    |   |
|----|---|---|----|---|
| 12 | 3 | 6 | 14 | 7 |
| 10 | 5 | 8 | 13 | 8 |
| 11 | 6 | 7 | 12 | 7 |

층(Z축)을 차원이라 부를 때는 각 층을 다른 종류의 정보로 볼 때