머신 러닝

2023 Fall

강미선

numpy 라이브러리

- **데이터 다루는 일 : 목공 작업과 유사**
  - 원시 데이터는 잡음도 있고, 손실값도 있고, 아웃라이어도 많다.
  - 이런 원시 데이터를 가공하여 핵심이 되는 패턴을 추출하고 패턴을 조합하고 변환하여 원하는 정보를 추출한다.

[그림 A-4] 데이터를 다루는 일과 비슷한 목공 작업 과정

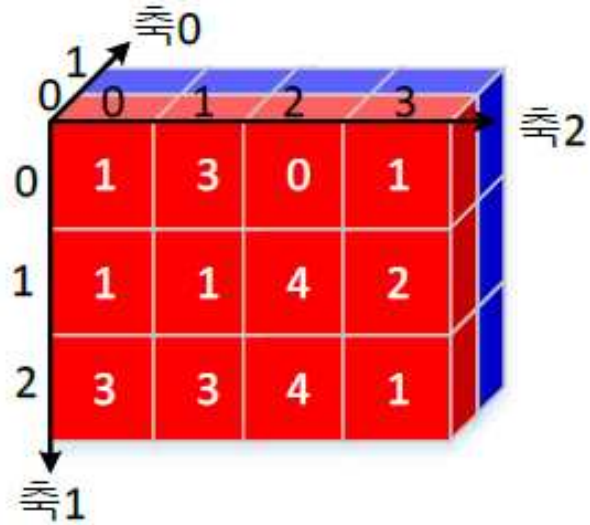
  - 목공에서는 벌레 먹고, 파이고, 갈라진 원목을 자르고, 붙이고, 대패질하여 만든 판재를 조립하여 멋진 피크닉 테이블을 만든다.
  - 목공을 잘하려면 연장을 아주 능숙하게 쓸 수 있어야 한다. 마찬가지로 기계학습을 잘하려면 데이터를 가공하는 도구를 능숙하게 쓸 수 있어야 한다.
  - 기계학습에서 도구란 파이썬의 라이브러리가 제공하는 각종 함수들
  - 공식사이트 https://numpy.org/ [Documentation]-『Numpy User Guide』3장 참조

numpy 라이브러리가 제공하는 함수 연습

```
> import numpy as np 01
> a=np.array([12,8,20,17,15]) 02
> print(a)
[12 8 20 17 15]
> a.sort() 03
> print(a)
[ 8 12 15 17 20]
> dir(a) 04
['T',
'__abs__',
'__add__',
'__and__',
…
'shape',
'size',
'sort’,
…]
```

numpy 라이브러리가 제공하는 함수 연습

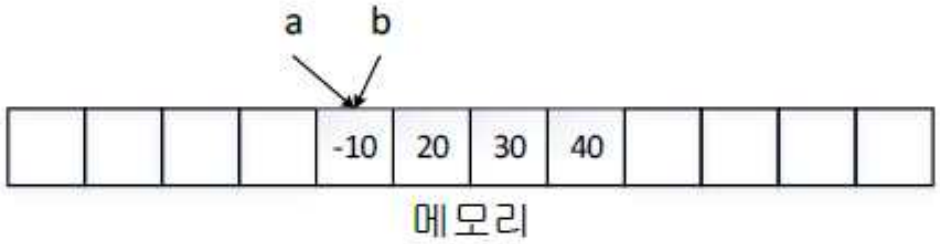
```
> type(a) 05
numpy.ndarray
> a.shape 06
(5,)
> b=np.array([[12,3,4.0],[1,4,5]]) 07
> print(b)
[[12. 3. 4.]
[ 1. 4. 5.]]
> b.ndim 08
2
> b.shape 09
(2,3)
> b.dtype 10
dtype('float64')
> c=np.array([[[1,3,0,1],[1,1,4,2],[3,3,4,1]], [[2,1,2,1] ,[1,0,1,0],[1,5,6,2]]]) 11
> print(c)
[[[1 3 0 1]
[1 1 4 2]
[3 3 4 1]]
[[2 1 2 1]
[1 0 1 0]
[1 5 6 2]]]
> c.shape
(2, 3, 4)
```



numpy 라이브러리가 제공하는 함수 연습

```
> d=np.zeros([2,3]) 12
> print(d)
[[0. 0. 0.]
[0. 0. 0.]]
> d.dtype
dtype('float64')
> e=np.random.random([2,5])
> print(e)
[[0.86997882 0.32528632 0.13021243 0.48668632 0.24810257]
[0.13114461 0.28725987 0.49369103 0.01422569 0.2509103 ]]
> f=np.arange(1,20,2.5) 13
> print(f)
[ 1. 3.5 6. 8.5 11. 13.5 16. 18.5]
> x=10
> y=x
> y=12
> print(x, y)
10 12
> a=np.array([10,20,30,40])
> b=a
> b[0]=-10
> print(a,b)
[-10 20 30 40] [-10 20 30 40]
```

(a) 서로 다른 스칼라 변수는 다른 메모리 공간을 사용



(b) 배열에서는 같은 메모리 공간을 공유

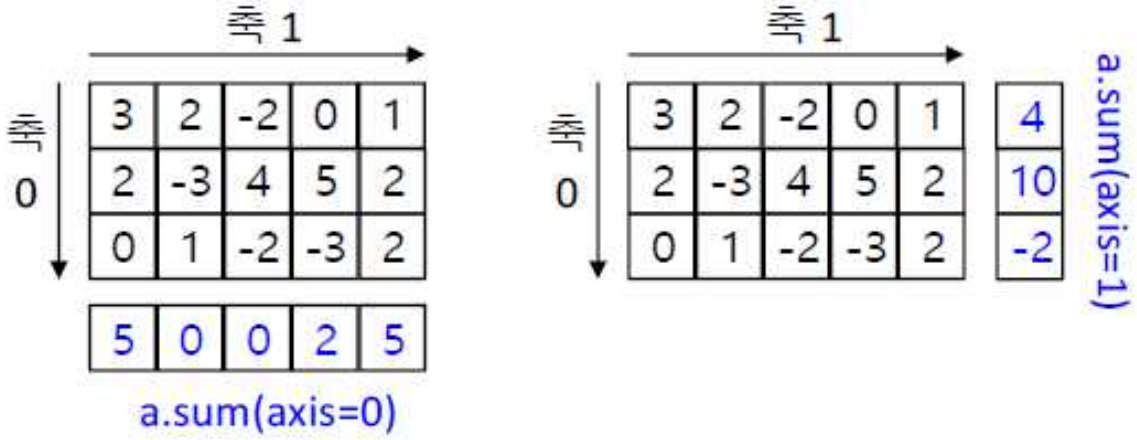
numpy 라이브러리가 제공하는 함수 연습

```
> c=a.copy() 16
> c[1]=-20
> print(a,c)
[-10 20 30 40] [-10 -20 30 40]
> a=np.array([1,2,3,4,5,6])
> b=a.reshape([2,3]) 17
> print(a)
[1 2 3 4 5 6]
> print(b)
[[1 2 3]
[4 5 6]]
> a[4]=-5
> print(b)
[[ 1 2 3]
[ 4 -5 6]]
> a=np.array([[1,2,3],[4,5,6]])
> b=a.T 18
> print(b) 14
[[1  4]
[2  5]
[3  6]]
```

numpy 라이브러리가 제공하는 함수 연습

```
> a=np.array([[3,2,-2,0,1],[2,-3,4,5,2],[0,1,-2,-3,2]])
> print(a)
[[ 3 2 -2 0 1]
[ 2 -3 4 5 2]
[ 0 1 -2 -3 2]]
> print(a.sum())
12
> print(a.sum(axis=0))
[5 0 0 2 5]
> print(a.sum(axis=1))
[ 4 10 -2]
> print(a.cumsum(axis=0))
[[ 3 2 -2 0 1]
[ 5 -1 2 5 3]
[ 5 0 0 2 5]]
> print(a.max(axis=0))
[3 2 4 5 2]
> print(a.argmax(axis=0))
[0 0 1 1 1]
```

```
> positive=a>0
> print(positive)
[[ True True False False True]
[ True False True True True]
[False True False False True]]
> b=a[positive]
> print(b)
[3 2 1 2 4 5 2 1 2]
> b.sum()
22
```



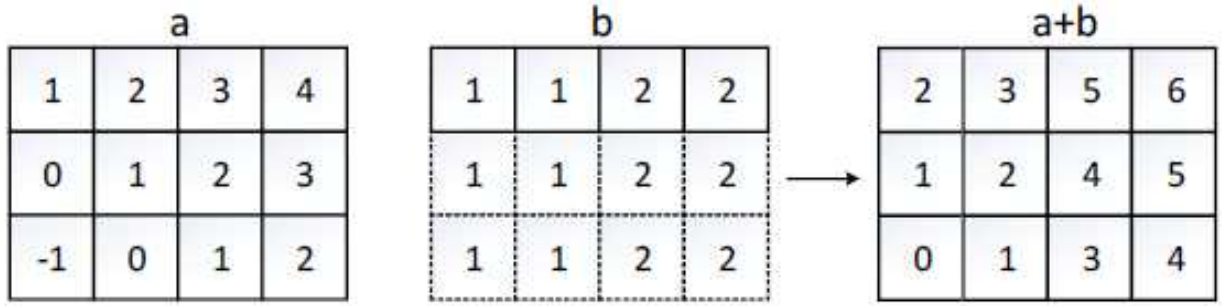

```
> a[a>0].sum()
```

numpy 라이브러리가 제공하는 함수 연습

```
> a=np.array([1,2,3])
> b=np.array([4,5,6])
> c=np.array([[7,8,9],[1,4,7]])
> x=np.vstack([a,b]) 25
> print(x)
[[1 2 3]
[4 5 6]]
> y=np.vstack([a,c]) 26
> print(y)
[[1 2 3]
[7 8 9]
[1 4 7]]
> z=np.hstack([a,b]) 27
> print(z)
[1 2 3 4 5 6]
> zz=np.hstack([a,c]) 28
ValueError: all the input arrays must have same number of dimensions, but the array at index 0 has 1 dimension(s) and
the array at index 1 has 2 dimension(s)
```

numpy 라이브러리가 제공하는 함수 연습

```
> a=np.array([1,2,3,4,5])
> b=np.array([0,-1,2,6,1])
> c=np.array([np.pi/2,np.pi,np.pi*2])
> print(c)
[1.57079633 3.14159265 6.28318531]
> print(np.add(a,b))
[1 1 5 10 6]
> print(np.log10(a))
[0. 0.30103 0.47712125 0.60205999 0.69897]
> print(np.sin(c))
[ 1.0000000e+00 1.2246468e-16 -2.4492936e-16]
> print(np.greater(a,b))
[ True True True False True]
> print(np.maximum(a,b))
[1 2 3 6 5]
> print(c.round(2))
[1.05 3.14 6.28]
```



[그림 A-8] 브로드캐스팅

```
> a=np.array([[1,2,3,4],[0,1,2,3],[-1,0,1,2]]) # shape: (3,4)
> b=np.array([[1,1,2,2]]) # shape: (1,4)
> print(a+b) # shape: (3,4)
[[2 3 5 6]
[1 2 4 5]
[0 1 3 4]]
```