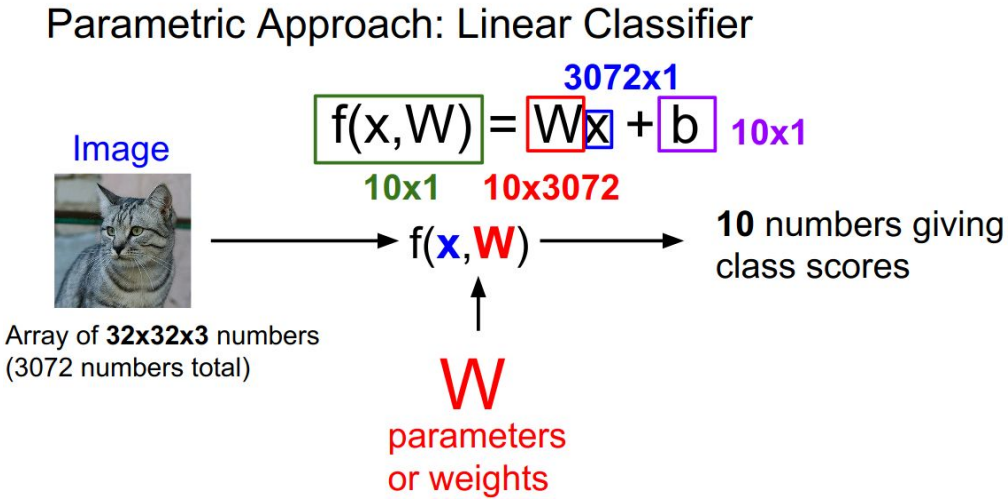
- Numpy - 수치 계산용 라이브러리
- pandas - 표 형태의 데이터를 가공
- matplotlib - 데이터 시각화
- 크롤링

Numpy

박 윤진

Numpy

- 수치 계산용 라이브러리



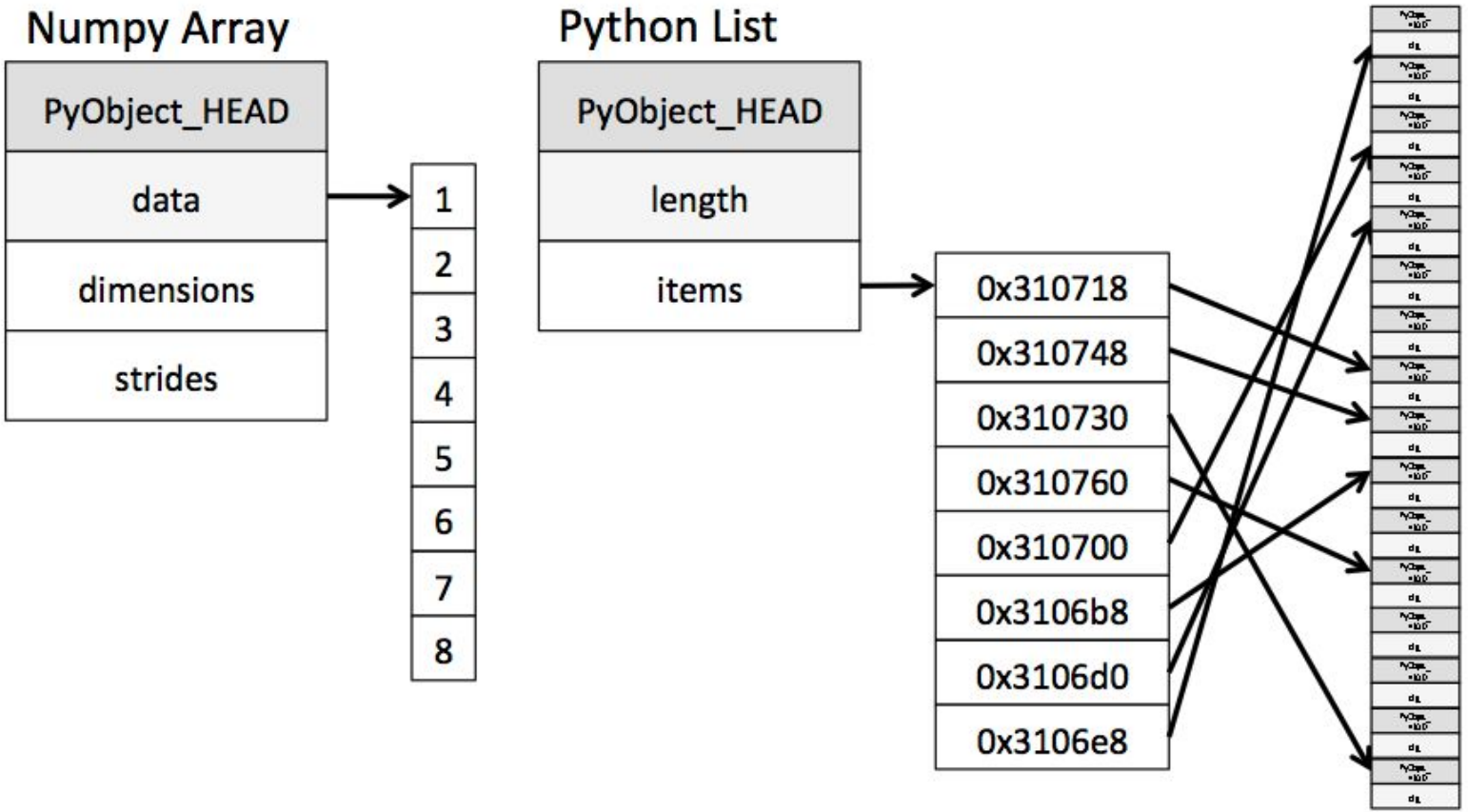
http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture02.pdf

Numpy 핵심 기능

- Ndarray (빠르고 메모리를 효율적으로 사용. 브로드캐스팅)
- 반복문 사용할 필요 없이 전체 데이터 배열에 대해 빠른 연산을 제공하는 표준 수학 함수
- C, C++, 포트란으로 쓰여진 코드를 통합하는 도구
- 선형대수, 푸리에 변환, 난수 발생기 등

자료구조 Ndarray

- **Numpy의 배열 클래스**
- N차원의 배열 객체
- 같은 종류의 데이터를 담는 포괄적인 다차원 배열.
- 연속된 메모리 블록에 저장.
- 대규모 데이터 집합을 담을 수 있는 빠르고 유연한 자료구조.



https://jakevdp.github.io/blog/2014/05/09/why-python-is-slow/

Ndarray 생성

1. 기존 데이터로 새로운 배열 생성

```
arr = np.array([1, 2, '1', 5], dtype=float)
arr
```

```
array([1., 2., 1., 5.])
```

```
arr = np.array(((1, 2), (5, 4)))
arr
```

```
array([[1, 2],
       [5, 4]])
```

Ndarray 생성

2. 배열 생성 함수

```
np.arange(10)
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
np.zeros((2, 10), dtype=np.int8)
```

```
array([[0, 0, 0, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 0, 0, 0, 0]], dtype=int8)
```

```
np.identity(4, dtype=int)
```

```
array([[1, 0, 0, 0],
       [0, 1, 0, 0],
       [0, 0, 1, 0],
       [0, 0, 0, 1]])
```

Ndarray 생성

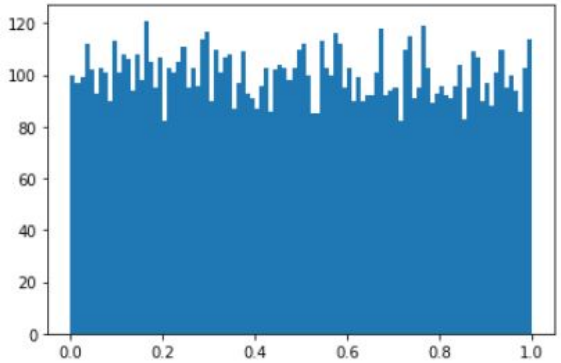
3. 난수 생성

```
#np.random.seed(1)
np.random.rand(2, 3, 5)
```

```
array([[[0.50288202, 0.08060164, 0.3905657 , 0.19249811, 0.76407301],
        [0.91586351, 0.5846998 , 0.56100169, 0.04449082, 0.36925879],
        [0.03572579, 0.82240462, 0.34053645, 0.39812181, 0.46640374]],

       [[0.71848453, 0.39810484, 0.57955681, 0.61051237, 0.71051238],
        [0.77855958, 0.64577428, 0.25143461, 0.91733223, 0.61736203],
        [0.14218331, 0.76845393, 0.72586682, 0.18670671, 0.86163692]]])
```

```
data = np.random.rand(10000)
plt.hist(data, bins=100)
plt.show()
```



Ndarray attributes

1. dtype

- int8, float16, complex32, bool, object, string_, unicode_ 등 다양한 데이터 타입이 있음. (참고)
- dtype이 있어 Numpy가 강력하면서 유연한 도구가 됨. C나 포트란 같은 저수준 언어로 작성된 코드와 쉽게 연동이 가능.

Ndarray attributes

1. dtype

```
arr = np.array([1, 2, 3, 4], dtype=np.float64)
print("Size of the array: ", arr.size)
print("Length of one array element in bytes: ", arr.itemsize)
print("Total bytes consumed by the elements of the array: ", arr.nbytes)
```

```
Size of the array:  4
Length of one array element in bytes:  8
Total bytes consumed by the elements of the array:  32
```

```
arr = np.array([1, 2, 3, 4], dtype=np.int8)
print("Size of the array: ", arr.size)
print("Length of one array element in bytes: ", arr.itemsize)
print("Total bytes consumed by the elements of the array: ", arr.nbytes)
```

```
Size of the array:  4
Length of one array element in bytes:  1
Total bytes consumed by the elements of the array:  4
```

Ndarray attributes

2. size: Number of elements in the array.

3. shape: Tuple of array dimensions.

4. ndim: Number of array dimensions.

```
arr = np.zeros((2, 4, 3))
```

```
arr.size
```

```
24
```

```
arr.shape
```

```
(2, 4, 3)
```

```
arr.ndim
```

```
3
```

Ndarray attributes

5. T (transpose)

```
x = np.arange(6).reshape((-1,3))
x
```

```
array([[0, 1, 2],
       [3, 4, 5]])
```

```
x.T
```

```
array([[0, 3],
       [1, 4],
       [2, 5]])
```

Handling shape

1. flatten

1	2	3
4	5	6
7	8	9

1
2
3
4
5
6
7
8
9

```
arr = np.zeros((2, 5))
arr
```

```
array([[0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.]])
```

```
arr.flatten()
```

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

Handling shape

2. reshape

```
arr = np.arange(20)
arr
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19])
```

```
arr.reshape(2, 10)
```

```
array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14, 15, 16, 17, 18, 19]])
```

```
arr.reshape(3, 10)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call l
<ipython-input-15-89997537a515> in <module>()
----> 1 arr.reshape(3, 10)

ValueError: cannot reshape array of size 20 into shape (3,10)
```

Handling shape

1. transpose

```
arr = np.ones((4, 6))
arr.transpose().shape
```

```
(6, 4)
```

```
arr = np.zeros((2, 3, 4, 5, 6))
arr.transpose().shape
```

```
(6, 5, 4, 3, 2)
```

```
arr = np.arange(30).reshape(3, 2, 5)
arr
```

```
array([[[ 0,  1,  2,  3,  4],
        [ 5,  6,  7,  8,  9]],

       [[10, 11, 12, 13, 14],
        [15, 16, 17, 18, 19]],

       [[20, 21, 22, 23, 24],
        [25, 26, 27, 28, 29]]])
```

```
arr.transpose((1, 0, 2))
```

```
array([[[ 0,  1,  2,  3,  4],
        [10, 11, 12, 13, 14],
        [20, 21, 22, 23, 24]],

       [[ 5,  6,  7,  8,  9],
        [15, 16, 17, 18, 19],
        [25, 26, 27, 28, 29]]])
```

indexing/ slicing

1. indexing

```
arr1d = np.arange(8)
arr1d
```

```
array([0, 1, 2, 3, 4, 5, 6, 7])
```

```
arr1d[1]
```

```
1
```

```
arr2d = np.arange(20).reshape(2, -1)
arr2d
```

```
array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14, 15, 16, 17, 18, 19]])
```

```
arr2d[1][0]
```

```
10
```

```
arr2d[1, 0]
```

```
10
```

indexing/ slicin

2. slicing

```
arr1d = np.arange(8)
arr1d
```

```
array([0, 1, 2, 3, 4, 5, 6, 7])
```

```
arr1d[:4]
```

```
array([0, 1, 2, 3])
```

```
arr1d[3:6] = 100 # 브로드캐스팅
```

```
arr1d
```

```
array([  0,   1,   2, 100, 100, 100,   6,   7])
```

```
arr2d = np.arange(20).reshape(2, -1)
arr2d
```

```
array([[ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14, 15, 16, 17, 18, 19]])
```

```
arr2d[:, :2]
```

```
array([[ 0,  1],
       [10, 11]])
```

```
arr3d = np.arange(20).reshape(2, 2, -1)
arr3d
```

```
array([[[ 0,  1,  2,  3,  4],
        [ 5,  6,  7,  8,  9]],

       [[10, 11, 12, 13, 14],
        [15, 16, 17, 18, 19]]])
```

```
arr3d[0, 1, 2]
```

```
7
```

indexing/ slicing

3. boolean indexing

```
arr = np.array([0, 1, 2, 3, 4], int)
arr[[True, False, True, False, True]]
```

```
array([0, 2, 4])
```

indexing/ slicing

4. fancy indexing:

Numpy에서 정수 배열을 인덱스로 사용하여 인덱싱 하는 것.

```
arr = np.arange(10)*10+5
arr
```

```
array([ 5, 15, 25, 35, 45, 55, 65, 75, 85, 95])
```

```
arr[[0, 2, 4, 6]]
```

```
array([ 5, 25, 45, 65])
```

산술연산

Numpy 배열의 중요한 특징:

for 문을 작성하지 않고 데이터를 일괄 처리할 수 있음(벡터화)

```
arr = np.array([[1, 2], [3, 4]])
arr
```

```
array([[1, 2],
       [3, 4]])
```

```
arr + arr
```

```
array([[2, 4],
       [6, 8]])
```

```
arr*3
```

```
array([[ 3,  6],
       [ 9, 12]])
```

```
arr/3
```

```
array([[0.33333333, 0.66666667],
       [1.        , 1.33333333]])
```

유니버셜 함수

Ndarray 안에 있는 데이터 원소별로

(element-wise)

연산을 수행하는 함수

```
arr = np.arange(0, 5, 0.5).reshape(2, -1)
arr
```

```
array([[0. , 0.5, 1. , 1.5, 2. ],
       [2.5, 3. , 3.5, 4. , 4.5]])
```

```
np.ceil(arr)
```

```
array([[0., 1., 1., 2., 2.],
       [3., 3., 4., 4., 5.]])
```

```
np.floor(arr)
```

```
array([[0., 0., 1., 1., 2.],
       [2., 3., 3., 4., 4.]])
```

유니버셜 함수

Function	설명
abs, fabs	각 원소의 절대값. 복소수가 아닌 경우 fabs를 쓰면 빠른 연산이 가능
sqrt	각 원소의 제곱근 계산(arr**0.5)
square	각 원소의 제곱을 계산(arr**2)
exp	각 원소에 지수e^x 계산
log, log10, log2, log1p	자연로그, 밑10인 로그, 밑2인 로그, log(1+x)
sign	각 원소의 부호(양수 1, 영 0, 음수 -1)
ceil	각 원소의 값보다 같거나 큰 정수 중 가장 작은 값
floor	각 원소의 값보다 같거나 작은 정수 중 가장 큰 값
rint	각 원소의 소수자리를 반올림
modf	각 원소의 몫과 나머지를 각각의 배열로 반환
isnan	각 원소가 숫자인지 아닌지. 불리언 배열로 반환
isfinite, isinf	각 원소가 유한한지, 무한한지. 불리언 배열로 반환
cos, cosh, sin, sinh, tan, tanh	일반 삼각함수, 쌍곡삼각함수
arccos, arccosh, arcsin, arcsinh, arctan, arctanh	역삼각함수

유니버셜 함수

```
arr1 = np.arange(10).reshape(2, 5)
arr2 = np.arange(-50, 50, 10).reshape(2, 5)
print(arr1)
print(arr2)
```

```
[[0 1 2 3 4]
 [5 6 7 8 9]]
[[-50 -40 -30 -20 -10]
 [  0  10  20  30  40]]
```

```
np.maximum(arr1, arr2)
```

```
array([[ 0,  1,  2,  3,  4],
       [ 5, 10, 20, 30, 40]])
```

유니버셜 함수

Function	설명
add	두 배열의 같은 위치의 원소끼리 합함
subtract	첫번째 배열의 원소에서 두번째 배열의 원소를 뺌
multiply	같은 위치의 원소끼리 곱함
divide, floor_divide	첫번째 배열의 원소를 두번째 배열의 원소를 나눔. floor는 몫만 취함
power	첫번째 매열의 원소를 두번째 배열의 원소만큼 제곱함
maximum, fmax	각 배열의 두 원소 중 큰 값을 반환. fmax는 NaN 무시
minimum, fmin	각 배열의 두 원소 중 작은 값을 반환. fmax는 NaN 무시
mod	첫번째 배열의 원소를 두번째 배열의 원소로 나눈 나머지
copysign	첫번째 배열의 원소의 기호를 두번째 배열의 원소의 기호로 바꿈
greater, greater_equal, less, less_equal, less, less_equal,	각 두 원소 간의 비교 연산(<, <=, >, >=, ==, !=)를 불리언 배열로 반환

기본 통계 메서드

```
arr = np.random.randn(1000, 5000)
arr.shape
```

```
(1000, 5000)
```

```
arr.mean()   # ndarray의 메서드 이용
```

```
-2.67739981543579e-05
```

```
np.mean(arr)  # numpy의 최상위 함수를 이용
```

```
-2.67739981543579e-05
```

- sum
- mean
- std, var
- min, max
- argmin, argmax
- cumsum
- cumprod
- cov
- corrcoef

any, all

```
arr = np.array([True, False, True])
arr.any(), arr.all()
```

```
(True, False)
```

```
arr = np.array([0, 0, 0])
arr.any(), arr.all()
```

```
(False, False)
```

where

```
xarr = np.array([100, 200, 300, 400])
yarr = np.array([1, 2, 3, 4])
cond = np.array([True, False, True, False])
```

```
# 파이썬 기본
result = [x if c else y
          for x, y, c in zip(xarr, yarr, cond)]
result
```

```
[100, 2, 300, 4]
```

```
# numpy 사용 => 큰 배열을 빠르게 처리 가능. 다차원일 때도 간결하게 표현 가능
result = np.where(cond, xarr, yarr)
```

정렬

```
arr = np.random.randint(1, 100, size=10)
arr
```

```
array([83, 18,  1, 67, 55, 61, 82, 69, 29, 81])
```

```
arr.sort()
```

```
arr
```

```
array([ 1, 18, 29, 55, 61, 67, 69, 81, 82, 83])
```

집합 관련 함수

```
arr = np.array(['Kim', 'Park', 'Kim', 'Lee', 'Kim', 'Na', 'Kim', 'Park', 'Lee', 'Na', 'Bae'])
np.unique(arr)
```

```
array(['Bae', 'Kim', 'Lee', 'Na', 'Park'], dtype='<U4')
```

```
arr1 = np.array(['a', 'b', 'c', 'd'])
arr2 = np.array(['e', 'd'])
np.intersect1d(arr1, arr2)
```

```
array(['d'], dtype='<U1')
```

```
np.in1d(arr1, arr2)
```

```
array([False, False, False,  True])
```

```
np.in1d(arr2, arr1)
```

```
array([False,  True])
```

선형대수

np.dot(x, y)

```
x = np.random.randint(-5, 5, size=10)
y = np.random.randint(-10, 10, size=10)
x.shape, y.shape
```

```
((10,), (10,))
```

```
x.dot(y)
```

```
-39
```

```
np.dot(x, y)
```

```
-39
```

선형대수

np.matmul(x, y)

```
a = np.random.randint(-3, 3, 10).reshape(2, 5)
b = np.random.randint(0, 5, 15).reshape(5, 3)
a.shape, b.shape
```

```
((2, 5), (5, 3))
```

```
np.matmul(a, b)
```

```
array([[17,  6, 14],
       [-5,  8,  5]])
```

```
np.matmul(b, a)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-68-af3b88aa2232> in <module>()
----> 1 np.matmul(b, a)

ValueError: matmul: Input operand 1 has a mismatch in its core dimension 0, with gufu
```

선형대수

np.diag(x)

```
matrix = np.arange(25).reshape(5, 5)
```

```
np.diag(matrix)
```

```
array([ 0,  6, 12, 18, 24])
```

```
np.diag([10, 30, 50])
```

```
array([[10,  0,  0],
       [ 0, 30,  0],
       [ 0,  0, 50]])
```

np.linalg.det(x)

```
matrix = np.array([[10, 5], [2, 3]])
matrix
```

```
array([[10,  5],
       [ 2,  3]])
```

```
np.linalg.det(matrix)
```

```
20.000000000000007
```

선형대수

np.linalg.inv(x)

```
matrix = np.array([[10, 20], [30, 40]])
```

```
inv_matrix = np.linalg.inv(matrix)
inv_matrix
```

```
array([[-0.2 ,  0.1 ],
       [ 0.15, -0.05]])
```

```
matrix@inv_matrix
```

```
array([[1.0000000e+00, 0.0000000e+00],
       [4.4408921e-16, 1.0000000e+00]])
```

np.linalg.solve(A, b)

```
A = np.array([[3, 1, -2], [0, -5, 7], [4, 0, 1]])
b = np.array([-5,  1, -1])
x = np.linalg.solve(A, b)
x
```

```
array([-1.,  4.,  3.])
```

참고 자료

강의 노트

파이썬 라이브러리를 활용한 데이터 분석(웨스 맥키니, 한빛미디어)

edwith, 머신러닝을 위한 Python(최성철)

퀴즈

밑바닥부터 시작하는 딥러닝 (사이토 고키, 한빛미디어)

https://www.w3resource.com/python/python-tutorial.php