# Lab - Numpy in a nutshell



## Introduction

[PDF 파일 다운로드]()

오래 기다리셨습니다. 드디어 Machin Learning 강의 첫 번째 Lab Assignment입니다. 머신러닝 강의는 사실 Lab 제작에 있어 많은 고민을 했습니다. 처음이야 상관없겠지만 뒤로 갈수록 데이터도 커지고, 좋은 머신이 아닐 경우 시간도 오래 걸려서 어려움이 많을 거 같습니다. 그러나, 일단은 시작하기로 했습니다.

첫 번째 LAB은 Numpy 입니다. Numpy 강의는 사실 그냥 보고만 있으면, 그렇게 어렵지 않게 이해할 수 있습니다. 그러나 실제로 문제를 풀다보면 잘 못하는 경우가 굉장히 많습니다. 그런 사태를 미연에 방지하기 위해 한번 도전해 보도록 합시다.

## backend.ai 설치

숙제를 제출하기 앞서, 레블업의 backend.ai를 여러분의 파이썬에 설치하셔야 합니다. 설치하는 과정은 매우 쉽습니다. 아래처럼 터미널 또는 cmd 창에서 입력을 하시면 됩니다.

```
pip install backend.ai-client
```

## 숙제 파일(lab_numpy.zip) 다운로드

먼저 해야 할 일은 숙제 파일을 다운로드 받는 것 입니다. Chrome 또는 익스플로러와 같은 웹 브라우저 주소창에 아래 주소를 입력합니다.

lab_numpy.zip

다운로드 된 `lab_bla.zip` 파일을 작업 폴더로 이동한 후 압축해제 후 작업하시길 바랍니다.
압축해제 하면 폴더가 `linux_mac` 과 `windows` 로 나눠져 있습니다. 자신의 OS에 맞는 폴더로 이동해서 코드를 수정해 주시기 바랍니다.

## numpy_lab.py 코드 구조

본 Lab은 Numpy의 실행 방법을 이해하기 위해 8개의 함수를 작성해야 합니다. `데이터 과학을 위한 파이썬 입문` 강의를 이미 수강해 본 분들은 쉽게 이해할 수 있을 거라고 생각합니다.

각 함수별 구체적인 작성 방법은 아래와 같습니다.

## n_size_ndarray_creation

- Template

```
def n_size_ndarray_creation(n, dtype=np.int):
    X = None
    return X
```

- 함수목적
  - n의 제곱수로 2 dimentional array를 생성하는 ndarray.
- Args
  - n: 생성하고자 하는 ndarray의 row와 column의 개수
  - dtype: 생성하려는 ndarray의 data type (np.int)
- Returns
  - row와 column의 길이가 n인 two dimentional ndarray로 X[0,0]은 0으로 순차적으로 X[n-1,n-1]은 n^2이 할당됨

## zero_or_one_or_empty_ndarray

- Template

```
def zero_or_one_or_empty_ndarray(shape, type=0, dtype=np.int):
    X = None
  return X
```

- 함수목적
  - shape이 지정된 크기의 ndarray를 생성, 이때 행렬의 element는 type에 따라 0, 1 또는 empty로 생성됨.
- Args
  - shape: 생성할려는 ndarray의 shape
  - type: 생성되는 element들의 값을 지정함0은 0, 1은 1, 99는 empty 타입으로 생성됨
  - dtype: 생성하려는 ndarray의 data type (np.int)
- Returns
  - shape의 크기로 생성된 ndarray로 type에 따라 element의 내용이 변경됨
- Examples

```
>>> zero_or_one_or_empty_ndarray(shape=(2,2), type=1)
array([[ 1,  1],
       [ 1,  1]])
>>> zero_or_one_or_empty_ndarray(shape=(3,3), type=99)
#임의의수 생성
array([[1773984320,        487, 1774114944],
       [       487, 1947927088,          0],
       [1947927088,          0, 1701605485]])
```

## change_shape_of_ndarray

- Template

```
def change_shape_of_ndarray(X, n_row):
    return X
```

- 함수목적
  - 입력된 ndarray X를 n_row의 값을 row의 개수로 지정한 matrix를 반환함.
  - 이때 입력하는 X의 size는 2의 거듭제곱수로 전제함.
  - 만약 n_row과 1일 때는 matrix가 아닌 vector로 반환함.
- Args
  - X: 입력하는 ndarray
  - n_row: 생성할려는 matrix의 row의 개수
- Returns
  - row의 개수가 n_row인 Matrix 또는 Vector
  - n_row가 1이면 Vector 값으로 반환함
- Examples

```
>>> import numpy as np
>>> import numpy_lab as testcode
>>> X = np.ones((32,32), dtype=np.int)
>>> testcode.change_shape_of_ndarray(X, 1)
array([1, 1, 1, ..., 1, 1, 1])
>>> testcode.change_shape_of_ndarray(X, 512)
array([[1, 1],
       [1, 1],
       [1, 1],
       ...,
       [1, 1],
       [1, 1],
       [1, 1]])
```

## concat_ndarray

- Template

```
def concat_ndarray(X_1, X_2, axis):
    pass
```

- 함수목적
  - 입력된 ndarray X_1과 X_2를 axis로 입력된 축을 기준으로 통합하여 반환하는 함수
  - X_1과 X_2는 matrix 또는 vector 임, 그러므로 vector 일 경우도 처리할 수 가 있어야 함
  - axis를 기준으로 통합할 때, 통합이 불가능하면 False가 반환됨.
  - 단 X_1과 X_2 Matrix, Vector 형태로 들어왔다면, Vector를 row가 1개인 Matrix로 변환하여 통합이 가능한지 확인할 것
- Args
  - X_1: 입력하는 ndarray
  - X_2: 입력하는 ndarray
  - axis: 통합의 기준이 되는 축 0 또는 1임
- Returns
  - X_1과 X_2과 통합된 matrix 타입의 ndarray
- Examples

```
>>> import numpy as np
>>> import numpy_lab as testcode
>>> a = np.array([[1, 2], [3, 4]])
>>> b = np.array([[5, 6]])
>>> testcode.concat_ndarray(a, b, 0)
array([[1, 2],
       [3, 4],
       [5, 6]])
>>> testcode.concat_ndarray(a, b, 1)
False
>>> a = np.array([1, 2])
>>> b = np.array([5, 6, 7])
>>> testcode.concat_ndarray(a, b, 1)
array([[1, 2, 5, 6, 7]])
>>> testcode.concat_ndarray(a, b, 0)
False
```

## concat_ndarray

- Template

```
def normalize_ndarray(X, axis=99, dtype=np.float32):
    pass
```

- 함수목적
  - 입력된 Matrix 또는 Vector를 ndarray X의 정규화된 값으로 변환하여 반환함
  - 이때 정규화 변환 공식 `Z = (X - X의평균) / X의 표준편차` 로 구성됨.
  - X의 평균과 표준편차는 axis를 기준으로 axis 별로 산출됨.
  - Matrix의 경우 axis가 0 또는 1일 경우, row 또는 column별로 Z value를 산출함.
  - axis가 99일 경우 전체 값에 대한 normalize 값을 구함.
- Args
  - X: 입력하는 ndarray,
  - axis: normalize를 구하는 기준이 되는 축으로 0, 1 또는 99임, 단 99는 axis 구분 없이 전체값으로 평균과 표준편차를 구함
  - dtype: data type으로 np.float32로 구정
- Returns
  - 정규화된 ndarray
- Examples

```
>>> import numpy as np
>>> import numpy_lab as testcode
>>> X = np.arange(12, dtype=np.float32).reshape(6,2)
>>> testcode.normalize_ndarray(X)
array([[-1.59325504, -1.3035723 ],
       [-1.01388955, -0.72420681],
       [-0.43452409, -0.14484136],
       [ 0.14484136,  0.43452409],
       [ 0.72420681,  1.01388955],
       [ 1.3035723 ,  1.59325504]], dtype=float32)
>>> testcode.normalize_ndarray(X, 1)
array([[-1.,  1.],
       [-1.,  1.],
       [-1.,  1.],
       [-1.,  1.],
       [-1.,  1.],
       [-1.,  1.]], dtype=float32)
>>> testcode.normalize_ndarray(X, 0)
array([[-1.46385002, -1.46385002],
       [-0.87831002, -0.87831002],
       [-0.29277   , -0.29277   ],
       [ 0.29277   ,  0.29277   ],
       [ 0.87831002,  0.87831002],
       [ 1.46385002,  1.46385002]], dtype=float32)
```

## save_ndarray

- Template

```
def save_ndarray(X, filename="test.npy"):
  pass
```

- 함수목적
  - 입력된 ndarray X를 argument filename으로 저장함
- Args
  - X: 입력하는 ndarray
  - filename: 저장할려는 파일이름
- Examples

```
>>> import numpy as np
>>> import numpy_lab as testcode
>>> X = np.arange(32, dtype=np.float32).reshape(4, -1)
>>> filename = "test.npy"
>>> testcode.save_ndarray(X, filename) #test.npy 파일이 생성됨
```

## boolean_index

- Template

```
def boolean_index(X, condition):
  pass
```

- 함수목적
  - 입력된 ndarray X를 String type의 condition 정보를 바탕으로 해당 컨디션에 해당하는 ndarray X의 index 번호를 반환함
  - 단 이때, str type의 조건인 condition을 코드로 변환하기 위해서는 `eval(str("X") + condition)` 를 사용할 수 있음
- Args
  - X: 입력하는 ndarray
  - condition: string type의 조건 (">3", "== 5", "< 15")
- Returns
  - 조건에 만족하는 ndarray X의 index
- Examples

```
>>> import numpy as np
>>> import numpy_lab as testcode
```

```
>>> X = np.arange(32, dtype=np.float32).reshape(4, -1)
>>> testcode.boolean_index(X, "== 3")
(array([0]), array([3]))
>>> X = np.arange(32, dtype=np.float32)
>>> testcode.boolean_index(X, "> 6")
(array([ 7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22,
        23, 24, 25, 26, 27, 28, 29, 30, 31]),)
```

## find_nearest_value

- Template

```
def find_nearest_value(X, target_value):
    pass
```

- 함수목적
  - 입력된 vector type의 ndarray X에서 target_value와 가장 차이가 작게나는 element를 찾아 리턴함
  - 이때 X를 list로 변경하여 처리하는 것은 실패로 간주함.
- Args
  - X: 입력하는 vector type의 ndarray
  - target_value : 가장 유사한 값의 기준값이 되는 값
- Returns
  - target_value와 가장 유사한 값
- Examples

```
>>> import numpy as np
>>> import numpy_lab as testcode
>>> X = np.random.uniform(0, 1, 100)
>>> target_value = 0.3
>>> testcode.find_nearest_value(X, target_value)
0.29260674329282488 # 출력되는 값은 random 하게 바뀜
```

## get_n_largest_values

- Template

```
def get_n_largest_values(X, n):
    pass
```

- 함수목적
  - 입력된 vector type의 ndarray X에서 큰 숫자 순서대로 n개의 값을 반환함.
- Args
  - X: vector type의 ndarray
  - n: 반환할려는 element의 개수
- Returns
  - ndarray X의 element중 큰 숫자 순서대로 n개 값이 반환됨 ndarray
- Examples

```
>>> import numpy as np
>>> import numpy_lab_solution as t
>>> X = np.random.uniform(0, 1, 100)
>>> t.get_n_largest_values(X, 3)
array([ 0.98935239,  0.98494578,  0.98317255])
>>> t.get_n_largest_values(X, 5)
array([ 0.98935239,  0.98494578,  0.98317255,  0.96859596,  0.96485649])
# 출력되는 값은 random 하게 바뀜
```

```
```

## 숙제 template 파일 제출하기 (윈도우의 경우)

1. `windows` + `r` 를 누르고 cmd 입력 후 확인을 클릭합니다.
2. 작업을 수행한 폴더로 이동 합니다.
3. 밑에 명령어를 cmd창에 입력합니다.

```
install.bat
submit.bat [YOUR_HASH_KEY]
```

## 숙제 template 파일 제출하기 (Mac or Linux)

1. 터미널을 구동합니다.
2. 작업을 수행한 디렉토리로로 이동 합니다.
3. 밑에 bash창을 입력합니다.

```
bash install.sh
bash submit.sh [YOUR_HASH_KEY]
```

> backend.ai 서비스의 업데이트에 의해 실행전 반드시 `bash install.sh` 또는 `install.bat` 수행을 바랍니다.

# Next Work

고생하셨습니다. 조금 Numpy를 성실하게 공부하셨던 분이라면 어렵지 않게 푸셨을 것 입니다. 지금 잘 이해하는게 정말 중요합니다. Machine Learning 과목에 가장 쉽고 재밌는 숙제였습니다.

> **Human knowledge belongs to the world** - from movie 'Password' -