Lab - Normal equation

Introduction

[PDF 파일 다운로드]()

이 번 랩은 우리가 강의를 통해 들은 Normal equation을 활용하여 LinearRegression 모듈을 구현하는 것을 목표로 한다. LinearRegression 모듈의 구현을 위해서는 numpy와 Python OOP의 기본적인 개념이해가 필요하다. 혹시 해당 개념들이 조금 헷갈린다면 다시 복습을 하고 올 것을 추천한다. 이번 랩을 통해 scikit-learn이나 tensorflow 같은 머신러닝 라이브러리들이 어떻게 구성되는지 조금이라도 알게 되면 좋겠다.

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_normal_equation.zip) 다운로드

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

- 링크 2_lab_bulid_matrix.zip
- https://s3.ap-northeast-2.amazonaws.com/teamlab-gachon/mooc_pic/lab_normal_equation.zip

  또는 Mac OS에서는 아래 명령을 쓰셔도 됩니다.

```
wget https://s3.ap-northeast-2.amazonaws.com/teamlab-gachon/mooc_pic/lab_normal
```

다운로드 된 `lab_normal_equation.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.

linear_model.py 코드 구조

본 Lab은 LinearRegression 모듈을 만들기 위해 `linear_model.py` 라는 template 파일을 제공합니다. 제공하는 template 파일은 아래와 같은 구조를 가지며 각 변수와 함수의 역할은 아래 테이블과 같습니다.

이름	종류	개요
fit	function	Linear regression 모델을 적합한다. Matrix X와 Vector Y가 입력값으로 들어오면 Normal equation을 활용하여, weight값을 찾는다.
predict	function	적합된 Linear regression 모델을 사용하여 입력된 Matrix X의 예측값을 반환한다. 이 때, 입력된 Matrix X는 별도의 전처리가 없는 상태로 입력되는 걸로 가정한다.
coef	property	normal equation의 결과 생성된 계수 (theta, weight) 값들을 numpy array로 저장한다.
intercept	property	normal equation의 결과 생성된 절편 (theta_0, weight_0) 값을 scalar 로 저장한다.

```
import numpy as np

class LinearRegression(object):
    def __init__(self, fit_intercept=True, copy_X=True):
        self.fit_intercept = fit_intercept
        self.copy_X = copy_X

        self._coef = None
        self._intercept = None
        self._new_X = None

    def fit(self, X, y):
      pass

    def predict(self, X):
      pass

    @property
    def coef(self):
        return self._coef
```

```
    @property
    def intercept(self):
        return self._intercept
```

Mission

이번 랩 부터는 제출에 의해서 컴퓨터가 직접 채점을 하는 auto-grading 문제와 수강자가 스스로 문제를 점검하는 self-grading 문제로 나눠서 진행된다. 본 랩에서는 auto-grading 문제는 `linear_model.py`의 LinearRegression class를 수정하는 것이고, self-grading 문제는 스스로 `ipynb`파일을 만들어서 제시된 문제들을 실제 수행하는 것이다.

fit 함수 만들기

fit 함수는 Linear regression 모델을 적합하는 함수 이다. 본 함수는 Matrix X와 Vector Y가 입력 값으로 들어오면 Normal equation을 활용하여, weight값을 찾는다. 이 때, fit_intercept 설정에 따라 다른 방시으로 적합이 실행이 달라진다. 또한 fit을 할 때는 입력되는 X의 값은 반드시 새로운 변수(self._new_X)에 저장한 후 실행되어야 한다.

> fit_intercept가 True일 경우: Matrix X의 0번째 Column에 값이 1인 column vector를 추가한다.
> 단 fit_intercept가 False일 경우의 Matrix X의 변수는 2개 이상일 것이라고 가정한다.

fit을 할 때는 아래의 Normal equation의 공식을 활용한다.

$$\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$$

적합이 종료된 후 각 변수의 계수(coefficient 또는 weight값을 의미)는 self._coef와 self._intercept_coef에 저장된다. 이때 self._coef는 numpy array을 각 변수항의 weight값을 저장한 1차원 vector이며, self._intercept_coef는 상수항의 weight를 저장한 scalar(float) 이다. 입력되는 parameter와 리턴은 아래 정보를 참고한다.

```
Parameters
----------
X : numpy array, 2차원 matrix 형태로 [n_samples,n_features] 구조를 가진다
y : numpy array, 1차원 vector 형태로 [n_targets]의 구조를 가진다.

Returns
-------
self : 현재의 인스턴스가 리턴된다
```

predict 함수 만들기

적합된 Linear regression 모델을 사용하여 입력된 Matrix X의 예측값을 반환한다. 이 때, 입력된 Matrix X는 별도의 전처리가 없는 상태로 입력되는 걸로 가정한다.

> fit_intercept가 True일 경우: Matrix X의 0번째 Column에 값이 1인 column vector를추가한다.

predict 함수는 다음 수식을 사용한다.

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{w}}$$

입력되는 parameter와 리턴은 아래 정보를 참고한다.

```
Parameters
----------
X : numpy array, 2차원 matrix 형태로 [n_samples,n_features] 구조를 가진다

Returns
-------
y : numpy array, 예측된 값을 1차원 vector 형태로 [n_predicted_targets]의
    구조를 가진다.
```

구현 모듈 테스트하기

여러분이 만들 모듈이 잘 작동하는지 테스트 하기 위해서 이제 `ipynb` 파일을 만들어 쥬피터 노트북에서 실행해보겠습니다.

모듈 호출하기

먼저 우리가 사용할 모듈을 호출한다. 기본적으로 pandas와 numpy를 호출한다. `linear_model` 모듈을 호출하기 위해서는 `import linear_model` 명령을 사용한다. 단 여기서 `linear_model.py` 파일이 자주 바꾸니 `imp` 모듈을 활용하여, 파일이 바뀌고 다시 저장될 때 `reload` 할 수 있도록 하여야 한다.

```
import pandas as pd
import numpy as np

import linear_model
import imp
imp.reload(linear_model)
```

데이터 불러하기

첫 번째 데이터는 one variable의 형태를 `test.py` 파일이다. 파일을 pandas로 호출하면 아래와 같은 구조를 확인할 수 있다.

```
df = pd.read_csv("./test.csv")
df.head()
```

	x	y
0	77	79.775152
1	21	23.177279
2	22	25.609262
3	20	17.857388
4	36	41.849864

numpy data 구성하기

pandas로 부터 필요한 데이터를 구성한다.

```
X = df["x"].values.reshape(-1,1)
y = df["y"].values
```

Model 생성하기

LinearRegression 클래스를 사용해서 모델을 생성한다.

```
lr = linear_model.LinearRegression(fit_intercept=True)
lr.fit(X, y)
lr.intercept # -0.46181077366099998
lr.coef # array([ 1.01433536])
lr.predict(X)[:10]
```

```
array([ 77.64201157,  20.83923168,  21.85356704,  19.82489633,
        36.05426201,  14.75321955,  62.42698124,  95.90004796,
        19.82489633,   4.609866  ])
```

Model Validation

구현한 LinearRegression 클래스와 scikit-learn의 linear regression 클래스로 만든 결과물을 비교한다.

```
from sklearn import linear_model
sk_lr = linear_model.LinearRegression(normalize=False)
sk_lr.fit(X, y)
sk_lr.intercept_  # -0.46181077366117762
sk_lr.coef_ # array([ 1.01433536])
np.isclose(lr.coef, sk_lr.coef_) #True
```

```
df_test = pd.read_csv("./train.csv")
df_test.head()
X_test = df["x"].values.reshape(-1,1)
lr.predict(X_test)[:5] # array([ 23.88223775,  50.25495698,  14.75321955,  38.08293
lr.predict(X_test)[:5] # array([ 23.88223775,  50.25495698,  14.75321955,  38.08293
```

Multiple variable

동일한 과정으로 다변수일 경우에도 작동할 수 있는지 확인한다.

```
df = pd.read_csv("./mlr09.csv")
df.head()
```

	height_in_feet	weight_in_pounds	successful_field_goals	percent_
0	6.8	225	0.442	0.672
1	6.3	180	0.435	0.797
2	6.4	190	0.456	0.761
3	6.2	180	0.416	0.651
4	6.9	205	0.449	0.900

```
y = df["average_points_scored"].values
X = df.iloc[:,:-1].values

mu_X = np.mean(X, axis=0)
std_X = np.std(X, axis=0)
```

```
rescaled_X = (X - mu_X) / std_X # RESCALED
```

Fitting 실행하기

```
lr.fit(rescaled_X, y)
lr.coef # array([-1.67779283,  0.28359762,  2.68586629,  1.12816882])
lr.intercept # 11.790740740740738

sk_lr.fit(rescaled_X, y)
sk_lr.coef_ # array([-1.67779283,  0.28359762,  2.68586629,  1.12816882])
sk_lr.intercept_ # 11.790740740740736
```

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

드디어 우리는 Numpy를 가지고 Machine Learning의 첫 번째 알고리즘을 구현해 보았다. 비교적 쉬운 숙

제였지만 Matrix 다루기 쉽지 않은 분들은 상당히 헷갈린 숙제일 것이다. 실제로 여러분이 짠 코드는 scikit-learn의 기본 LinearRegression Class와 크게 다르지 않다. 자부심을 가지고 한 발씩 나아가 보자!!

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