Machine Learning & Scikit-Learn

Scikit-learn을 이용한 전처리

- 표준화(standardization = mean removal and variance scaling)

```
from sklearn.preprocessing import scale, robust_scale, minmax_scale, maxabs_scale
```

```
import numpy as np
x = (np.arange(10, dtype=np.float) - 3).reshape(-1, 1)
```

```
x
```

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

```
import pandas as pd
df = pd.DataFrame(np.hstack([x, scale(x), robust_scale(x), minmax_scale(x), maxabs_scale(x)]),
columns=["x", "scale(x)", "robust_scale(x)", "minmax_scale(x)", "maxabs_scale(x)"])
```

Scikit-learn을 이용한 전처리

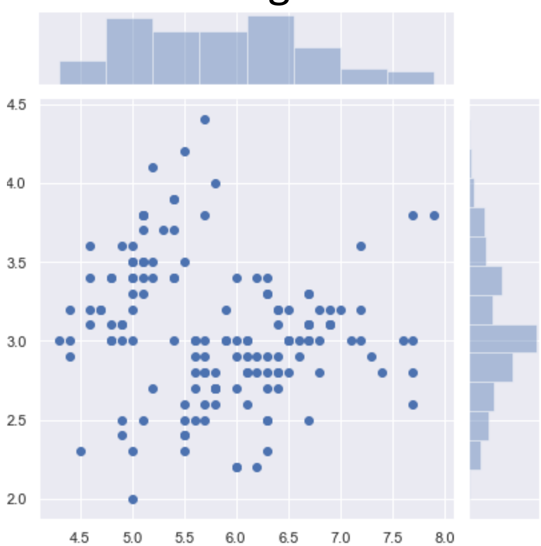
- 표준화(standardization = mean removal and variance scaling)

```python
from sklearn.datasets import load_iris
iris = load_iris()
data1 = iris.data
```

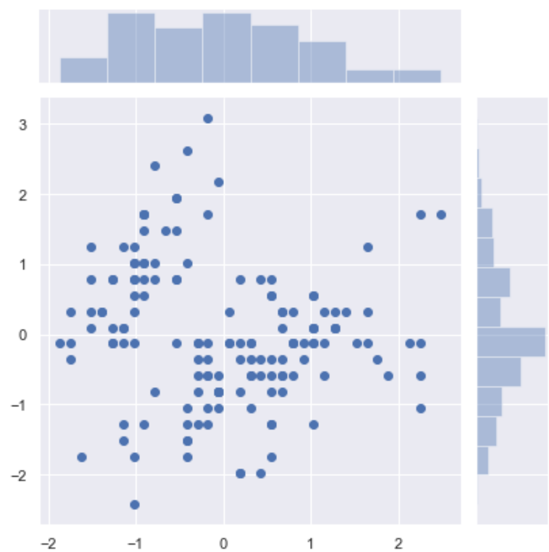
```python
data4 = minmax_scale(data1)
```

```python
sns.jointplot(data1[:,0], data1[:,1])
plt.show()
sns.jointplot(data4[:,0], data4[:,1])
plt.show()
```

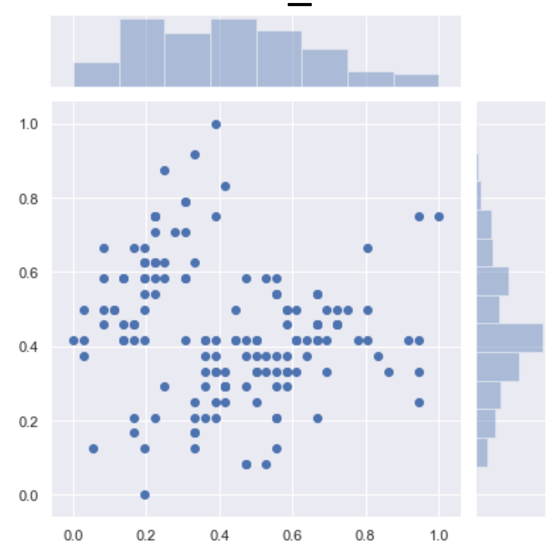
original



scale



minmax_scale



강의 시작 전 미리 실행해 둘 명령어

```
import seaborn as sns
import pandas as pd
import numpy as np
%matplotlib inline
```

```
sns.get_dataset_names() # seaborn이 지원하는 데이터 세트
```

```
data = sns.load_dataset(name = "iris")
```

```
data.info()
```

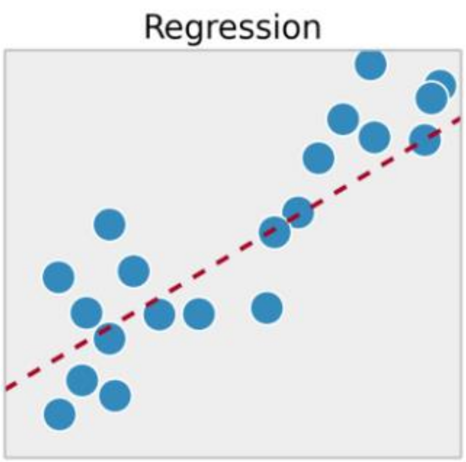
```
data.head()
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Day2. 회귀분석

통계모델:[2] 회귀분석

- 회귀분석(regression)
  - 학습 자료에 대한 근사식을 구해
    새로운 자료에 대한 레이블을 예측하는 분야
  - 오늘날의 활용 분야
    - ✓ 주식 가격 예측
    - ✓ 주문 및 판매량 분석
    - ✓ 의료 진단
    - ✓ 어떤 값과 시간과의 상관관계 분석
  - 방법론
    - ✓ Linear regression, Logistic regression, Polynomial regression, etc.



#1

선형회귀분석

회귀분석-(1) 선형회귀

1) 선형 회귀 분석(linear regression)

- 독립변수(x)가 1차항인 회귀분석

$$Y = AX = a_0 + a_1x_1 + \cdots + a_nx_n$$

- 종류
  - ✓ 단순선형회귀
    - 독립변수인 x가 한 개인 회귀 $Y = a_0 + a_1x_1$
  - ✓ 다중선형회귀
    - 독립변수 x가 두개 이상인 회귀 $Y = AX = a_0 + a_1x_1 + \cdots + a_nx_n$

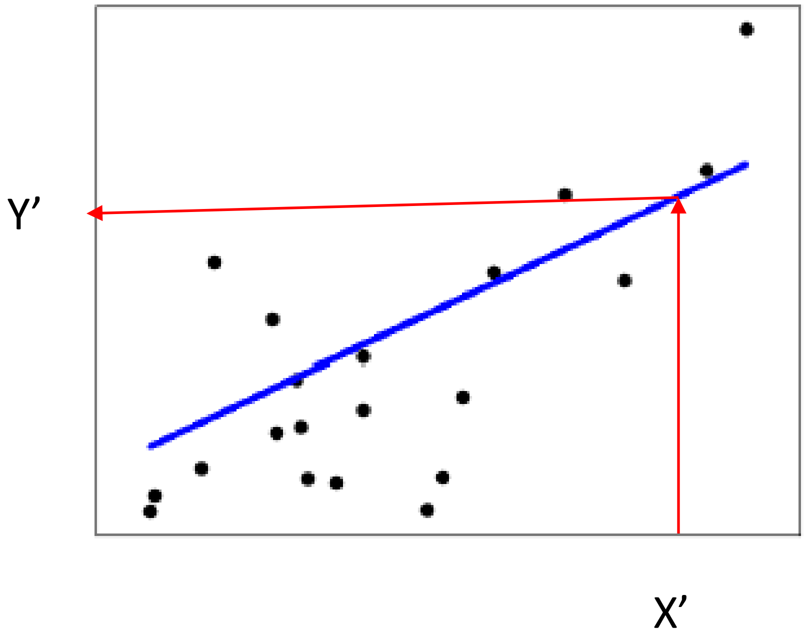
회귀분석-(1) 선형회귀

1) 선형 회귀 분석(linear regression)

- 종류
  - ✓ 단순선형회귀
    - (예) '몸무게가 늘어남에 따라 뇌의 무게가 증가한다'--> 내 강아지의 뇌 무게는 얼마일까?
  - ✓ 다중선형회귀
    - (예) 야식을 먹는 횟수, 운동 시간, 수면 시간으로 몸무게를 예측하면 얼마일까?

회귀분석-(1) 선형회귀

- 선형회귀(Linear Regression) 의 학습 및 예측 원리

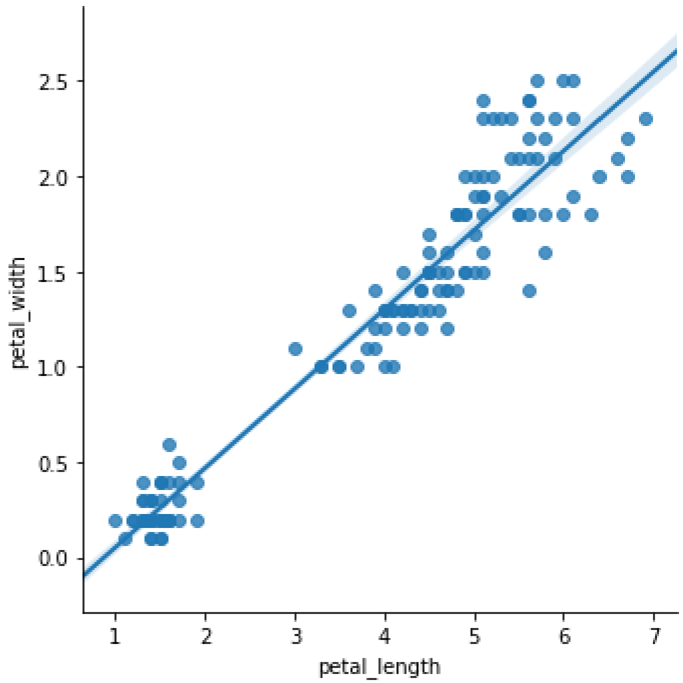


회귀분석-선형회귀-단순선형회귀

- 단순 선형 회귀
  - ✓ 회귀선: iris

```
# 데이터와 회귀를 한 번에 보임
sns.lmplot('petal_length','petal_width',data)
```

<seaborn.axisgrid.FacetGrid at 0x1f475c4f4a8>



회귀분석-선형회귀-단순선형회귀

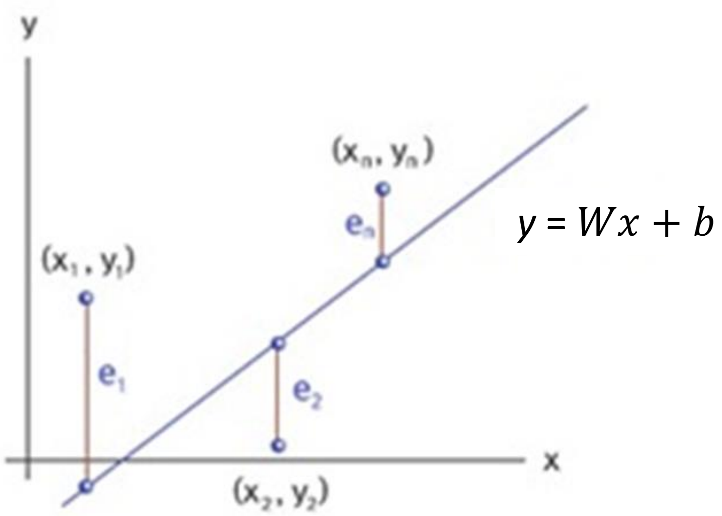
- 단순 선형 회귀
  - ✓ 회귀식 $y = Wx + b$
  - ✓ 모델 추정법 : 근사식 추정 방법
    - 해석적 방법
    - Gradient Descent 방법

회귀분석-선형회귀-단순선형회귀

- 단순 선형 회귀
  - ✓ 에러: $e_i = y_i - \hat{y}_i = y_i - Wxi - b$

    $(\hat{y}_i = Wx_i + b)$
  - ✓ Cost function : 최소 제곱법(mean square)

$$E(W, b) = \sum_{i=1}^{n}(y_i - Wx_i - b)^2$$



회귀분석-선형회귀-단순선형회귀

- 단순 선형 회귀
  - ✓ (예)게임 캐릭터의 파워
    - 가설: $y = Wx$
    - Cost function: $E(W) = \sum_{i=1}^{n}(y_i - Wx_i)^2$
      - ○ W를 1로 선택!!
    - 생명수 3병을 먹으면 파워는 얼마가 될까?

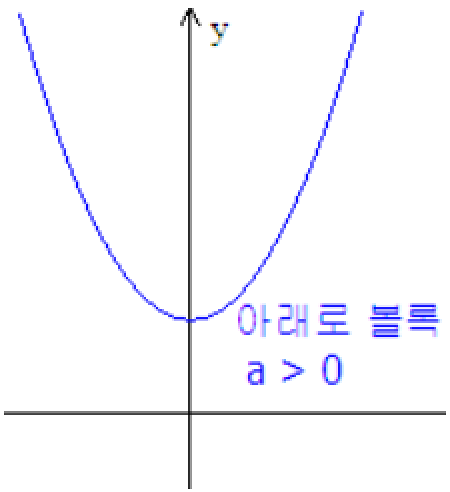
➔ '3마력'이 될 것임!

생명수(x)	파워(y)
1	1
2	2
4	4

W에 따른 cost E(W)
W=0 일 때, E(0) = 1 + 4 + 16 = 21
W=1 일 때, E(1) = 0
W=2 일 때, E(2) = 1 + 4 + 16 = 21

회귀분석-선형회귀-단순선형회귀

- 단순 선형 회귀
  - ✓ 모델 추정법
    - 해석적 방법
      - W와 b를 풀어서 구함
      - 가설: $y = Wx + b$
      - Cost function: $\mathrm{E(W,b)} = \sum_{i=1}^{n}(y_i - Wx_i - b)^2$
      - E(W,b)가 최소값이 되려면 $\frac{\partial E}{\partial W}$와 $\frac{\partial E}{\partial b}$가 0이 되어야 함

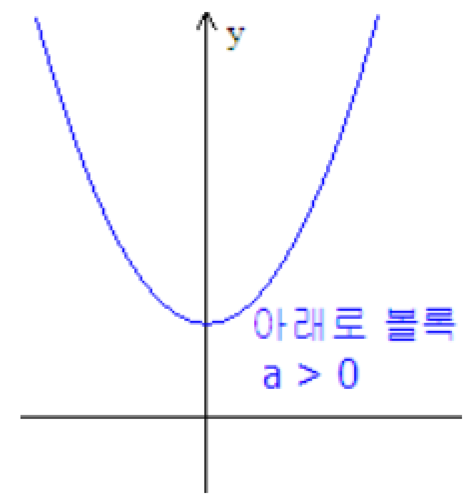


회귀분석-선형회귀-단순선형회귀

- 단순 선형 회귀
  - Cost function: $E(W, b) = \sum_{i=1}^{n}(y_i - Wx_i - b)^2$
  - E(W,b)가 최소값이 되려면 $\frac{\partial E}{\partial W}$와 $\frac{\partial E}{\partial b}$가 0이 되어야 함

$$\begin{cases} 0 = \frac{\partial E}{\partial W} = \sum 2(y_i - Wx_i - b)\ \ (-x_i) = 2\sum(Wx_i^2 + bx_i - x_i y_i) \\ 0 = \frac{\partial E}{\partial b} = \sum 2(y_i - Wx_i - b)\ \ (-1) = 2(W\sum x_i + b\sum 1 - \sum y_i) \end{cases}$$

$$\begin{cases} W\sum x_i^2 + b\sum x_i = \sum x_i y_i & \text{...(1)} \\ W\sum x_i + b\sum 1 = \sum y_i & \text{...(2)} \end{cases}$$



회귀분석-선형회귀-단순선형회귀

- 단순 선형 회귀

$$\text{가설: } y = Wx + b$$

$$\begin{cases} W\sum x_i^2 + b\sum x_i = \sum x_i y_i & \text{...(1)} \\ W\sum x_i + b\sum 1 = \sum y_i & \text{...(2)} \end{cases}$$

식(2) x$\frac{1}{n}$ 한 뒤 b로 정리하면 $\boxed{b = \bar{y} - W\bar{x}}$

$$\boxed{W = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}}$$

회귀분석-선형회귀-단순선형회귀

- 단순 선형 회귀
  - ✓ 모델 추정법
    - 해석적 방법의 구현

$$b = \bar{y} - W\bar{x}$$

$$W = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

```python
# Method 1: statistical
print('====================================================')
print(' Method 1:  Statistical Method')
print('====================================================')
Xmean = sum(X) / len(X)
Ymean = sum(y) / len(y)
print('Xmean:', Xmean, 'Ymean:',Ymean)
print('-----------------------------------------------------')
```

회귀분석-선형회귀-단순선형회귀

- 단순 선형 회귀
  - ✓ 모델 추정법
    - 해석적 방법의 구현

$$b = \bar{y} - W\bar{x}$$

$$W = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

```
M1_W=[0.0]
M1_b=[0.0]
total1 = 0
total2 = 0
for i in range(len(X)):
```

```
M1_W[0] = total1/total2
M1_b[0] = Ymean - M1_W[0]*Xmean
```

회귀분석-선형회귀-단순선형회귀

- 단순 선형 회귀
  - ✓ 모델 추정법
    - 해석적 방법의 구현

$$b = \bar{y} - W\bar{x}$$

$$W = \frac{\sum_{i=1}^{n}(y_i - \bar{y})(x_i - \bar{x})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

```
print('Linear Regression by Method 1 : y =', M1_W[0],'* x +',M1_b[0])
print('--------------------------------------------------------')
print('y        y_hat ')
print('--------------------')
```

```
print('--------------------')
M1_cost = costfunction(X, y, M1_W, M1_b, 0)
print('cost: ', M1_cost)
print('--------------------')
```

- cost function $E(W, b) = \sum_{i=1}^{n}(y_i - Wx_i - b)^2$

```
# cost function
def  costfunction(x, y, W, b, iters):
```

```
#테스트 세트
X = [-3, -2, -1, 0, 1, 2, 3]
y = [-3, -2, -1, 0, 1, 2, 3]
```

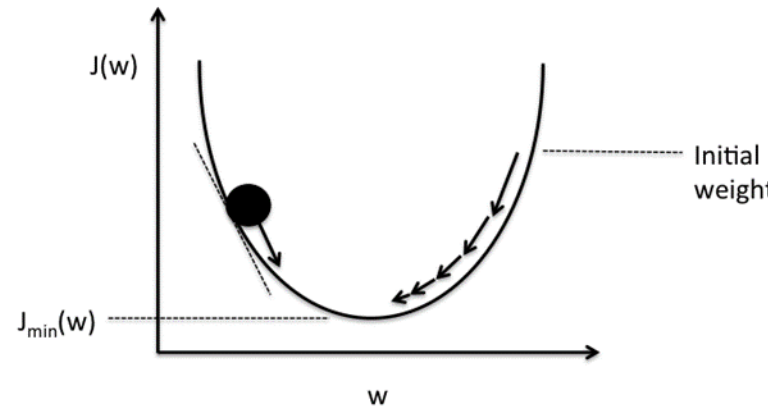
실행

```
# cost function
def  costfunction(x, y, W, b, iters):
    total = 0.0
    for i in range(len(x)):
        total += pow(W[iters]*x[i] +b[iters]- y[i], 2)
    return total/2

#테스트 세트
#X = [-3, -2, -1, 0, 1, 2, 3]
#y = [-3, -2, -1, 0, 1, 2, 3]  #y1
#y = [-2, -1, -0, 1, 2, 3, 4] #y2
# 화학 반응
#X=[10, 20, 30, 40]
#y=[71, 45, 24, 8]
# 동아리 키-몸무게 관계
X=[157, 160, 160, 168, 172, 175, 175, 177, 182, 184, 188, 190]
y=[42,   48,  54,  58,  63,  69,  71,  73,  70,  80, 79, 81]

M1_W=[0.0]
M1_b=[0.0]

# Method 1: statistical
print('====================================================')
print(' Method 1:  Statistical Method')
print('====================================================')
Xmean = sum(X) / len(X)
Ymean = sum(y) / len(y)
print('Xmean:', Xmean, 'Ymean:',Ymean)
print('-----------------------------------------------------')
total1 = 0
total2 = 0
for i in range(len(X)):
    total1 += (y[i] - Ymean)*(X[i]-Xmean)
    total2 += pow(X[i]-Xmean, 2)
M1_W[0] = total1/total2
M1_b[0] = Ymean - M1_W[0]*Xmean
print('Linear Regression by Method 1 : y =', M1_W[0],'* x +',M1_b[0])
```

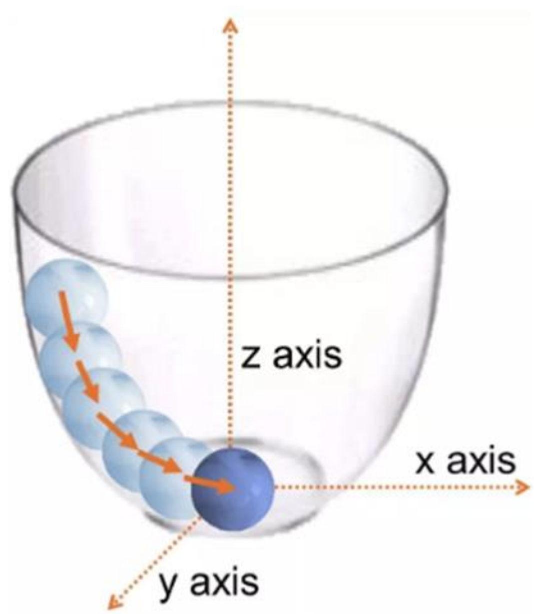
회귀분석-선형회귀-단순선형회귀

- 단순 선형 회귀
  - ✓ 모델 추정법
    - Gradient Descent 방법
      - Cost function

$$E(W,b) = \frac{1}{2}\sum_{i=1}^{n}(y_i - Wx_i - b)^2$$



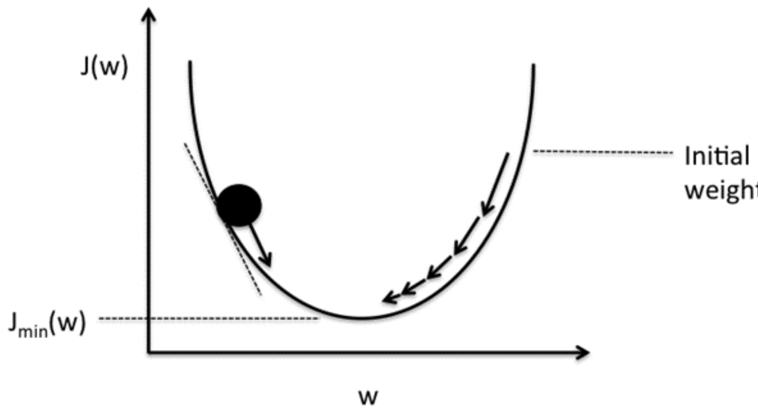
- Cost function의 기울기 방향으로 ΔW와 Δb 를 수정해 감
- $W(t+1) = W(t) - \eta \cdot \Delta W(t)$
- $b(t+1) = b(t) - \eta \cdot \Delta b(t)$



회귀분석-선형회귀-단순선형회귀

- 단순 선형 회귀
  - ✓ 모델 추정법
    - Gradient Descent 방법
      - Cost function

$$\mathrm{E(W,b)} = \frac{1}{2}\sum_{i=1}^{n}(y_i - Wx_i - b)^2$$

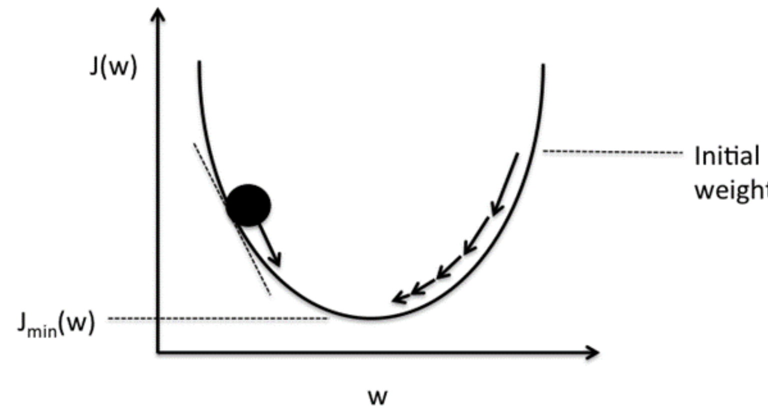


$$\Delta W(t) = \frac{\partial E}{\partial W} = \frac{1}{2}\sum 2(y_i - Wx_i - b)\ \ (-x_i) = \sum(Wx_i^2 + bx_i - x_i y_i)$$

$$\Delta b(t) = \frac{\partial E}{\partial b} = \frac{1}{2}\sum 2(y_i - Wx_i - b)\ \ (-1) = \sum(Wx_i + b - y_i)$$

회귀분석-선형회귀-단순선형회귀

- 단순 선형 회귀
  - ✓ 모델 추정법
    - Gradient Descent 방법
      - *가설:* $y = Wx + b$
      - $W(t+1) = W(t) - \eta \cdot \Delta W(t)$
      - $b(t+1) = b(t) - \eta \cdot \Delta b(t)$



$$\Delta W(t) = \frac{\partial E}{\partial W} = \frac{1}{2}\sum 2(y_i - Wx_i - b)\ \ (-x_i) = \sum(Wx_i^2 + bx_i - x_i y_i)$$

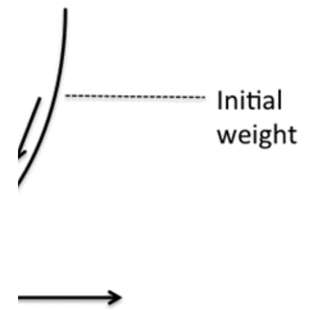
$$\Delta b(t) = \frac{\partial E}{\partial b} = \frac{1}{2}\sum 2(y_i - Wx_i - b)\ \ (-1) = \sum(Wx_i + b - y_i)$$

회귀분석-선형회귀-단순선형회귀

- 단순 선형 회귀

```
# Method 2
print('==================================================')
print(' Method 2:  Gradient Descent Method')
print('==================================================')
W=[0.0]
b=[0.0]
W[0] = float( random.randint(-100, 100))   ## M1_W[0] #
b[0] = float(random.randint(-100, 100))    ##  M1_b[0] #

 #This tells us when to stop the algorithm
iters = 0 #iteration counter
cost = costfunction(X, y,W,b, iters)

print("Iteration",iters,"\tW[0]:",W[0],"\tb[0]:",b[0],"\tcost:",cost)
```

```
rate =  10/cost # Learning rate   ## 0.000001* cost #
MaxItrs = cost   ## 10000  #
precision = 0.0001       ## M1_cost *0.8
while cost > precision: # and iters < MaxItrs:
    iters = iters+1 #iteration count
    gradientW = g_W(X, y, iters-1)
    gradientB = g_b(X, y, iters-1)
    newW = W[iters-1] - rate *  gradientW  #Grad descent
    newb = b[iters-1] - rate *  gradientB  #Grad descent
    W.append(newW)
    b.append(newb)
```



- $W(t+1) = W(t) - \eta \cdot \Delta W(t)$
- $b(t+1) = b(t) - \eta \cdot \Delta b(t)$

- Gradient descent 구현하기

$$\Delta W(t) = \frac{\partial E}{\partial W} = \frac{1}{2}\sum 2(y_i - Wx_i - b)\ \ (-x_i) = \sum(Wx_i^2 + bx_i - x_i y_i)$$

```
#Gradient
def g_W(x, y, iters):
    total=0.0
```

W, b 리스트의 iters 인덱스의 값 사용

```
    return total
```

- Gradient descent 구현하기

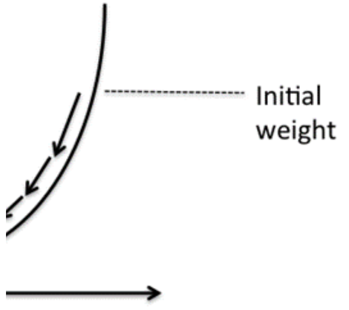
$$\Delta b(t) = \frac{\partial E}{\partial b} = \frac{1}{2}\sum 2(y_i - Wx_i - b) \quad (-1) = \sum(Wx_i + b - y_i)$$

```
def g_b(x, y, iters):
    total=0.0
```

W, b 리스트의 iters 인덱스의 값 사용

```
    return total
```

```
rate =  10/cost # Learning rate   ## 0.000001* cost #
MaxItrs = cost   ## 10000  #
precision = 0.0001       ## M1_cost *0.8
while cost > precision: # and iters < MaxItrs:
    iters = iters+1 #iteration count
    gradientW = g_W(X, y, iters-1)
    gradientB = g_b(X, y, iters-1)
    newW = W[iters-1] - rate *  gradientW  #Grad descent
    newb = b[iters-1] - rate *  gradientB  #Grad descent
    W.append(newW)
    b.append(newb)
    cost = costfunction(X, y, W, b, iters)

    if iters %100==0:  # 동아리 키-몸무게 사례에서는 1000000로 설정
        print('iteration: ', iters, end=',')
        print('gradient W: %.1f, gradient b:%.1f, ' % (gradientW, gradientB), end=' ')
        print('W[ %d ]:%.1f, b[%d]:%.1f, cost: %.1f'
                % (iters, W[iters] , iters, b[iters], cost))
        ans = input()
        if ans =='q':
            break
```



- ✓ 프로그램 실행
- ✓ Method1과 Method2의 결과 비교
- ✓ 실행 테스트 세트 : y1→y2 → 화학반응→동아리 키-몸무게 관계
  - 동아리 키-몸무게 관계는 실행 설정 변경 필요

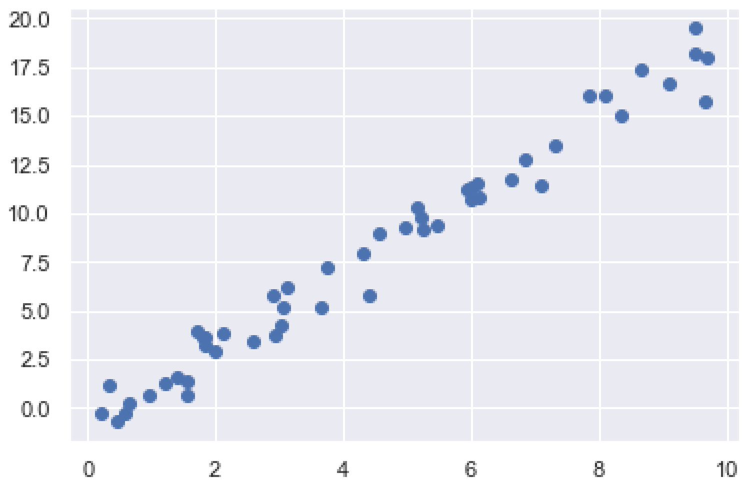
```
#테스트 세트
#X = [-3, -2, -1, 0, 1, 2, 3]
#y = [-3, -2, -1, 0, 1, 2, 3]  #y1
#y = [-2, -1, -0, 1, 2, 3, 4] #y2
# 화학 반응
#X=[10, 20, 30, 40]
#y=[71, 45, 24, 8]
# 동아리 키-몸무게 관계
X=[157, 160, 160, 168, 172, 175, 175, 177, 182, 184, 188, 190]
y=[42,   48,   54,   58,   63,   69,   71,   73,   70,   80, 79, 81]
```

Scikit-learn 을 사용한 선형회귀

1. 자료 준비 및 데이터 분포 보기

```
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
sns.set();
rng = np.random.RandomState(42)
x = 10 * rng.rand(50)
y = 2*x -1 + rng.randn(50)
plt.scatter(x, y)
```

```
<matplotlib.collections.PathCollection at 0x1f475b8be10>
```



Scikit-learn 을 사용한 선형회귀

2. 모듈 import

```
#모델 클래스 선택: 선형회귀
from sklearn.linear_model import LinearRegression
```

3. 모델을 인스턴스화

```
model= LinearRegression(fit_intercept = True)
```

```
LinearRegression(copy_X=True, fit_intercept=True,n_jobs=1, normalize=False)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

Scikit-learn 을 사용한 선형회귀

4. 데이터를 특징과 대상 벡터로 배치

```
X = x[:, np.newaxis]
```

```
x.shape
```

```
(50,)
```

```
X.shape
```

```
(50, 1)
```

Scikit-learn 을 사용한 선형회귀

5. 주어진 데이터로 모델을 학습시키기

```
model.fit(X, y)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Scikit-learn 을 사용한 선형회귀

6. 결과 확인

```
model.coef_
```

```
array([1.9776566])
```

```
model.intercept_
```

```
-0.903310725531164
```

Scikit-learn 을 사용한 선형회귀

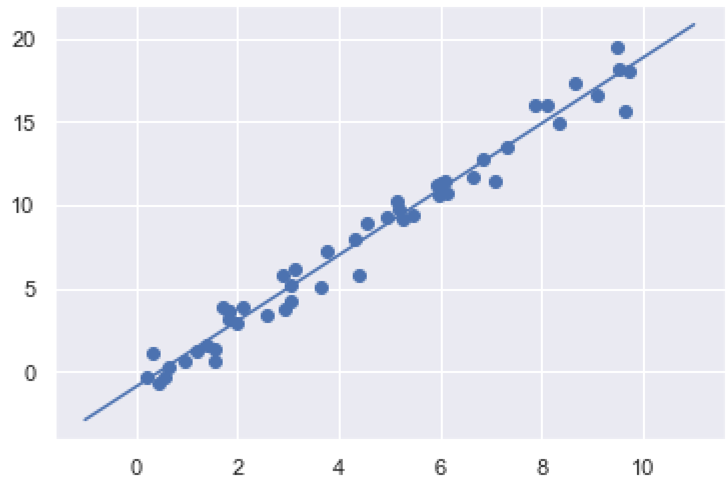
7. 학습된 모델로 새로운 데이터에 레이블 예측하기

```
xfit = np.linspace(-1, 11)
```

```
Xfit = xfit[:, np.newaxis]
yfit = model.predict(Xfit)
```

```
plt.scatter(x,y)
plt.plot(xfit, yfit)
```

[<matplotlib.lines.Line2D at 0x1f475a90be0>]

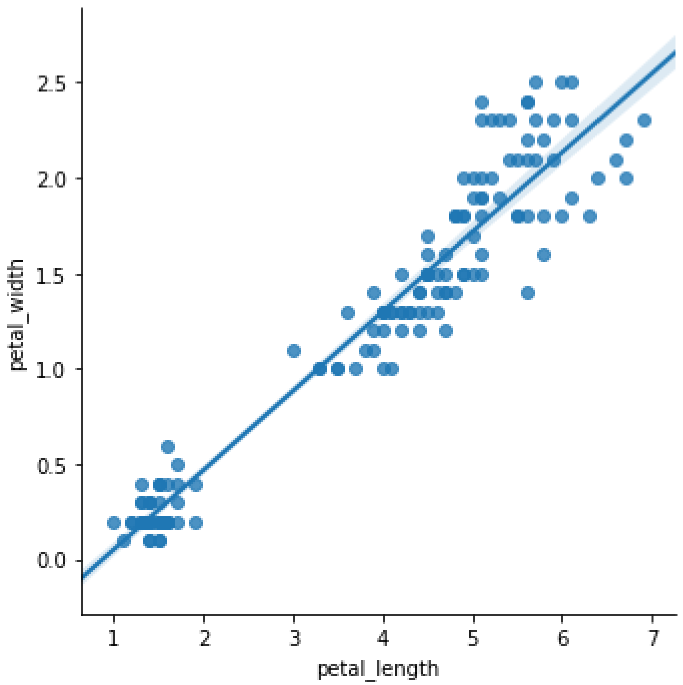


Scikit-learn 을 사용한 선형회귀

- IRIS 데이터에 적용하기

```
# 데이터와 회귀를 한 번에 보임
sns.lmplot('petal_length','petal_width',data)
```

<seaborn.axisgrid.FacetGrid at 0x1f475c4f4a8>



Scikit-learn 을 사용한 선형회귀

- IRIS 데이터에 적용하기

(1) 자료 준비하기

```
iris_x = data['petal_length']
```

```
iris_x.shape
```

```
(150,)
```

```
iris_X = iris_x[:, np.newaxis]
```

```
iris_X.shape
```

```
(150, 1)
```

```
iris_y = data['petal_width']
```

Scikit-learn 을 사용한 선형회귀

- IRIS 데이터에 적용하기

(2) 주어진 데이터에 인스턴스화된 모델 학습 시키기

```
model.fit(iris_X, iris_y)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

Scikit-learn 을 사용한 선형회귀

- IRIS 데이터에 적용하기

(3) 결과 확인

```
model.coef_
```

```
array([0.41575542])
```

```
model.intercept_
```

```
-0.36307552131190291
```

- IRIS 데이터에 적용하기

(4) 그래프 출력과 비교

```
model.coef_
```

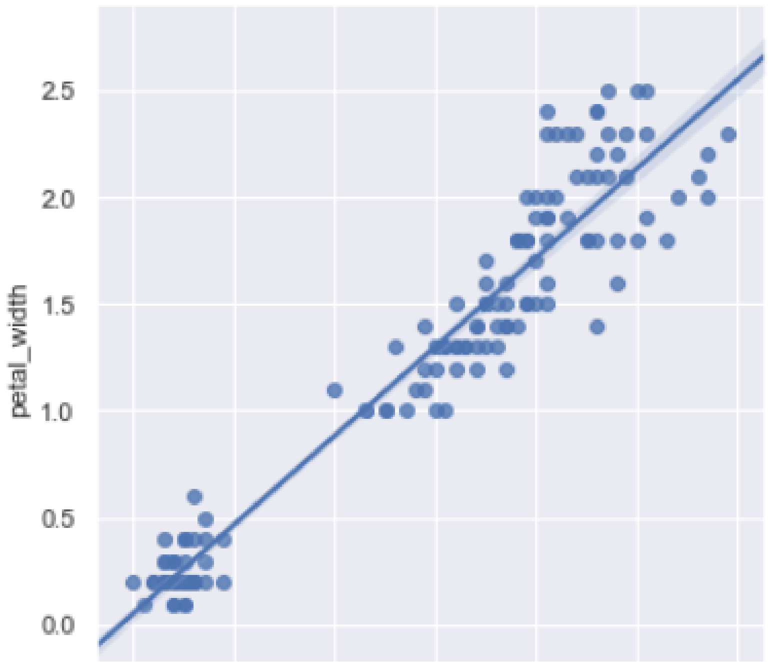
```
array([0.41575542])
```

```
model.intercept_
```

```
-0.3630755213190291
```

```
sns.lmplot('petal_length','petal_width',data)
```

<seaborn.axisgrid.FacetGrid at 0x572f291e48>



Q & A