Mechine Learning

-6주차 실습 과제-

chap.1 선형회귀

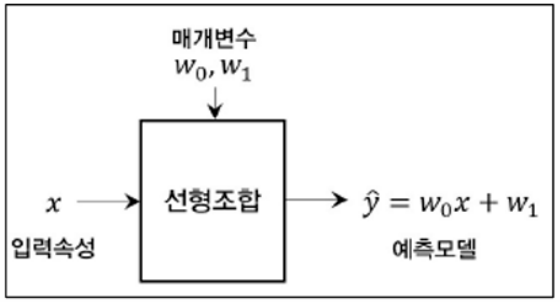
전자공학과

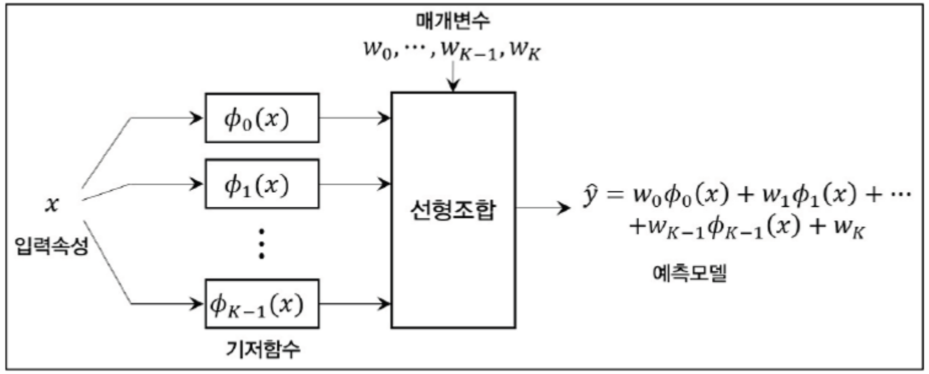
2022144007

김의진

#과제4

1) K개의 다항식 기저함수를 이용한 선형 기저 함수모델의 Analytic Solution을 구하는 사용자 지정 함수를 구현하라.





입력값을 Non-linear한 기저함수에 넣고 그 값을
선형조합해 Non-linear한 예측모델 얻기

#과제4

1) K개의 다항식 기저함수를 이용한 선형 기저 함수모델의 Analytic Solution을 구하는 사용자 지정 함수를 구현하라.

1)Polynominal basis function:

$$\phi_{\mathrm{k}}(x) = x^{k}$$

일반적인 지수함수 형태

Analystic solution

기저 함수 모델

$$w = (\Phi^{T}\Phi)^{-1}\Phi^{T}y$$

다중 선형 회귀

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

-> 둘은 겉보기엔 같아보이지만 다름

-> 다중 선형 회귀는 Linear인 예측값이 나옴

-> 기저 함수 모델은 Non-Linear 예측값이 나옴

#과제4

1) K개의 다항식 기저함수를 이용한 선형 기저 함수모델의 Analytic Solution을 구하는 사용자 지정 함수를 구현하라.

```python
'''by polynominal BSF analystic solution 구하는 함수'''
def PBSF_ANALYSTIC_SOLUTION(k):
    x_pbsf_matrix = x_vector                                          # x_pbsf_matrix 1행 설정
    for i in range(2, k + 1):
        x_pbsf_matrix = np.column_stack([x_pbsf_matrix, x_vector**i])  # Low방향으로 k제곱 한것 쌓기

    x_pbsf_matrix = np.column_stack([x_pbsf_matrix, x_dummy])          # dummy data 추가
    x_pbsf_matrix_T = np.transpose(x_pbsf_matrix)                      # transpose

    w = np.dot(np.dot(np.linalg.inv(np.dot(x_pbsf_matrix_T, x_pbsf_matrix)),x_pbsf_matrix_T), y_vector)   # analystic solution
    return w, x_pbsf_matrix
```

1)받아온 데이터에 입력값 k(k ≥ 1, 정수)에 따라 k개의 기저함수를 이용해 비선형적인 예측을 함
2) k+1개의 weight와 (50, k)개의 입력 데이터를 얻을 수 있음.
3) np.dot()과 np.linalg.inv()를 이용해 행렬곱과 역행렬 연산을 함.

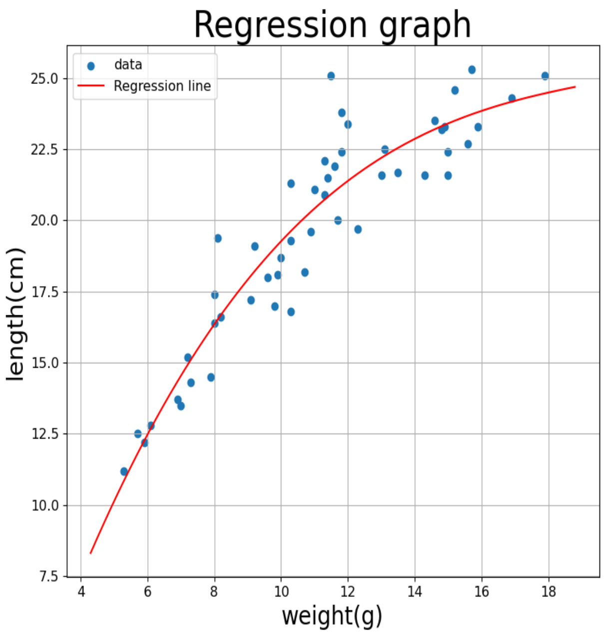
#과제 4

2) 1)에서 구현한 기저 함수 모델을 활용하여, K=3, 6, 8 일 때에 대한 weight를 표로 나타내고, raw 데이터와 회귀곡선을 그래프로 나타내라.

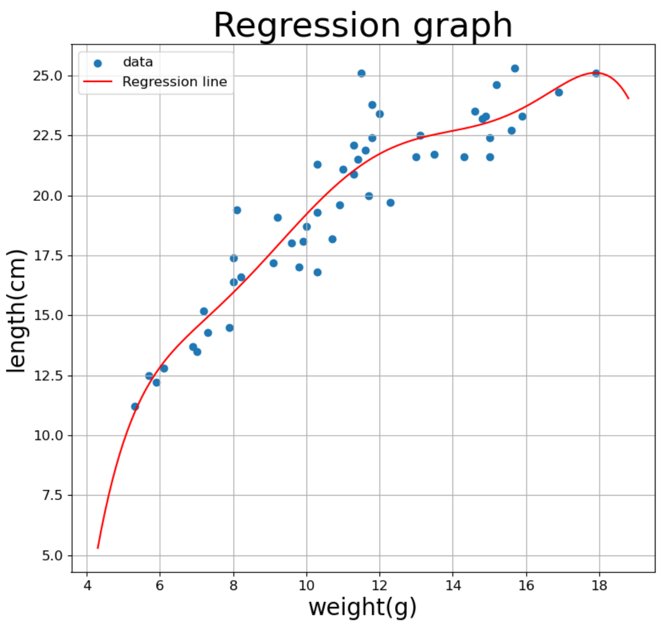
weight 표

```
Weights for k = 3:
w0: 4.2081
w1: -0.1960
w2: 0.0032
w3: -6.4142

Weights for k = 6:
w0: 127.3626
w1: -30.7400
w2: 3.8709
w3: -0.2645
w4: 0.0093
w5: -0.0001
w6: -204.1404

Weights for k = 8:
w0: -3058.2796
w1: 1055.8075
w2: -202.4170
w3: 23.6046
w4: -1.7164
w5: 0.0761
w6: -0.0019
w7: 0.0000
w8: 3773.1973
```

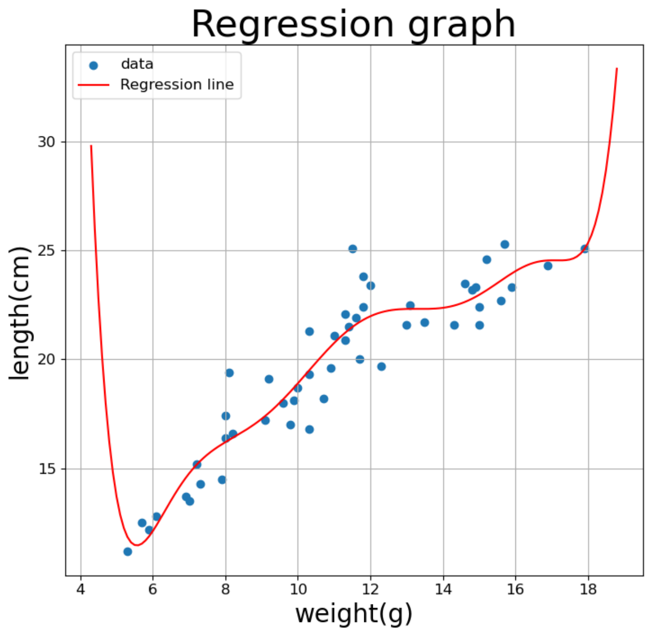
k = 3



k = 6



k = 8

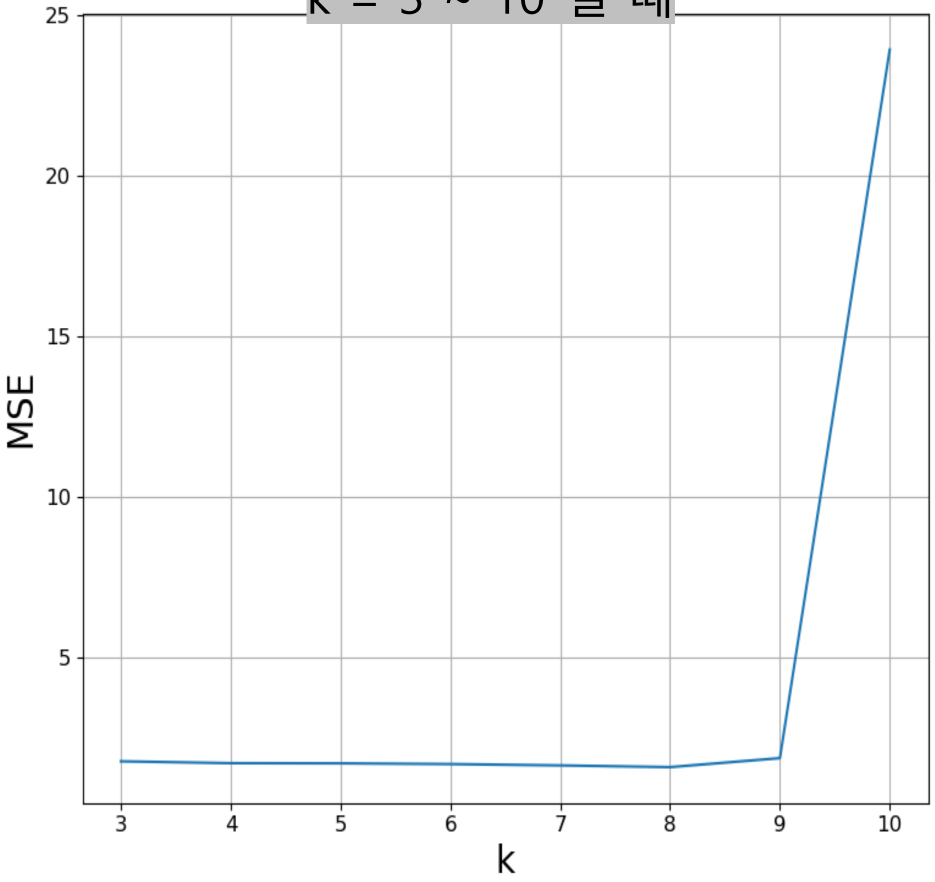


1) 기저 함수의 개수가 늘어날수록 곡선이 휘는 횟수가 더 많아짐
2) 입력데이터에 대한 예측의 정확도가 더 높아 보임

#과제4

3) K= 3~10에 대한 MSE 값을 그래프로 나타내라. (x축: K, y축:MSE)

k = 3 ~ 10 일 때

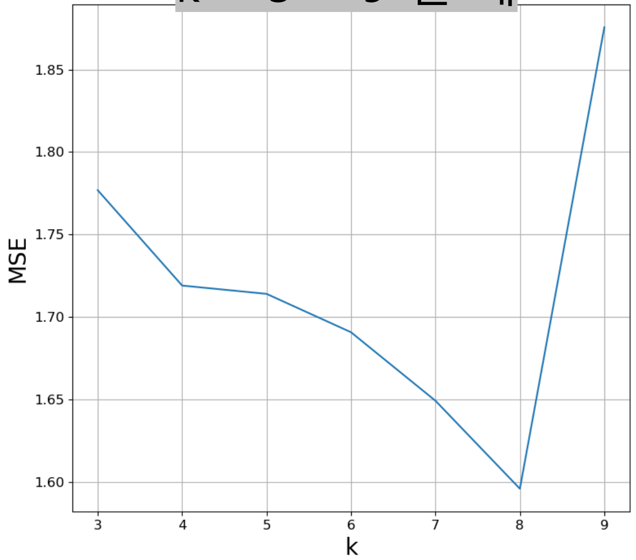


3~9까지는 1.~~가 나왔는데 10에서 갑자기 커져버림

->이유

1) Overfitting 됐을 가능성있음
2) 정규화 하지 않아서 파라미터들 값이 너무 커져 mse 폭발 가능성있음

k = 3 ~ 9 일 때



k = 8일 때
MSE값이 제일 작음

#과제4

Gaussian basis function일 때 1~3) 과정 반복

```python
''' by gaussian BSF analystic solution 구하는 함수'''
def GBSF_ANALYSTIC_SOLUTION(K):
    x_gbsf_matrix = []
    k = np.arange(K)
    mu = np.min(x_vector) + ((np.max(x_vector) - np.min(x_vector)) / (K - 1)) * k
    mu = np.reshape(mu, [len(mu), 1])
    sigma = (np.max(x_vector) - np.min(x_vector)) / (K - 1)

    basis = np.zeros([len(x_vector), K])
    for i in range(K):
        basis[:, i] = np.exp(-0.5 * ((x_vector - mu[i]) / sigma) ** 2)


    x_gbsf_matrix = basis  # Low방향으로 k제곱 한것 쌓기
    x_gbsf_matrix = np.column_stack([x_gbsf_matrix, x_dummy])  # dummy data 추가

    x_gbsf_matrix_T = np.transpose(x_gbsf_matrix)  # transpose

    w = np.dot(np.dot(np.linalg.inv(np.dot(x_gbsf_matrix_T, x_gbsf_matrix)),x_gbsf_matrix_T), y_vector)   # analystic solution
    return w, x_gbsf_matrix
```

$$\mu_k = x_{\min} + \frac{x_{\max} - x_{\min}}{K-1}k, \quad k = 0, 1, \cdots, K-1$$

$$\sigma = \frac{x_{\max} - x_{\min}}{K-1}$$

$$\phi_k(x) = e^{-\frac{1}{2}\left(\frac{x-\mu_k}{\sigma}\right)^2}$$

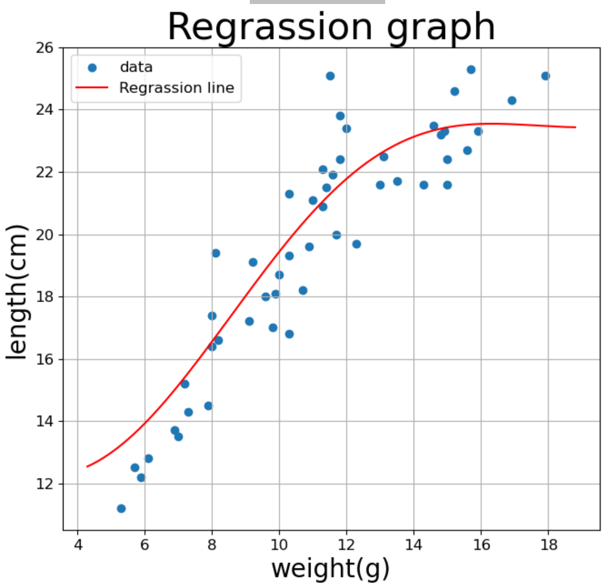
1) 옆의 Gaussian basis function을 참고하면 K개의 기저함수가 있으면 k개의 평균값과 하나의 분산 값을 구할 수 있음
2) basis함수를 만들어 주소 지정을 해 50, K 크기의 데이터를 만듦
3) Analystic solution 구하는 식에 대입

#과제4

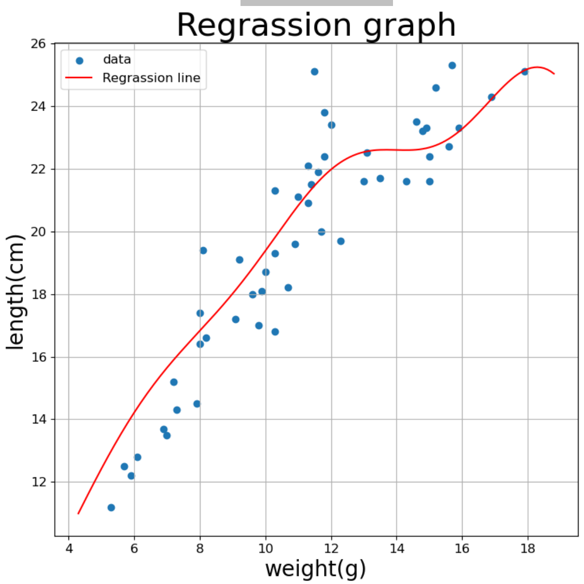
Gaussian basis function일 때 1~3) 과정 반복

```
Weights for k = 3:
w0: -28.3690
w1: 12.8479
w2: -15.7648
w3: 35.2459

Weights for k = 6:
w0: 2.6390
w1: 8.5952
w2: 0.9536
w3: 17.1944
w4: -3.7136
w5: 22.1192
w6: 2.8229

Weights for k = 8:
w0: -26.6901
w1: -1.6414
w2: -13.2456
w3: -10.4665
w4: -0.8778
w5: -17.7128
w6: 3.6682
w7: -16.8671
w8: 41.9130
```

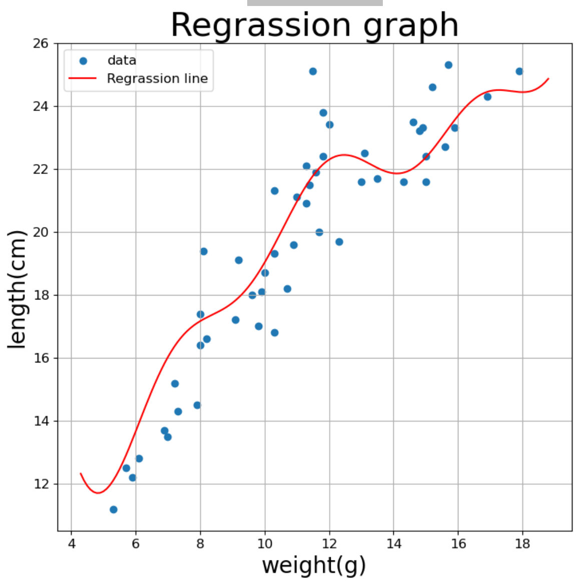
k = 3



k = 6



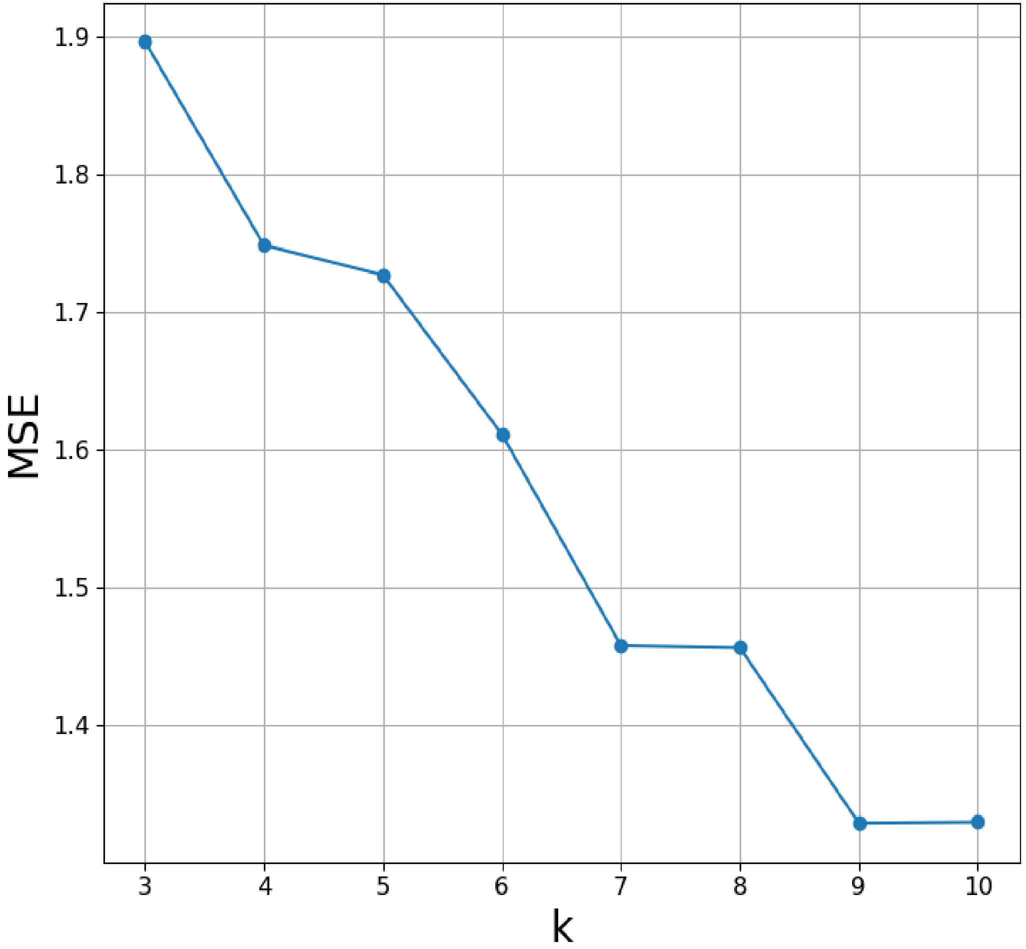
k = 8



Polynominal basis function과 다른 회귀 곡선이 나옴

#과제4

Gaussian basis function일 때 1~3) 과정 반복



1) K가 커질수록 MSE가 작아짐
2) Polynominal basis function과 달리 k 개수가 달라져도 값들이 안정적임

➔ Polynominal basis function은 발산하므로 입력값이 커질 수록 출력값도 기하급수적으로 커짐

➔ Gaussian basis function은 수렴하므로 입력값이 커진다고 출력값이 기하급수적으로 커지지 않음

➔ Gaussian basis function이 보다 안정적으로 regression 할 수 있음

#과제 4

Gaussian basis fucntion을 이용한 GDM으로 regression 하기

```python
''' by gaussian BSF GDM으로 weight 구하는 함수'''
def GBSF_GDM(K, epoch, lr):
    '''gaussian basis function 구현 '''
    x_gbsf_matrix = []
    k = np.arange(K)
    mu = np.min(x_vector) + ((np.max(x_vector) - np.min(x_vector)) / (K - 1)) * k
    mu = np.reshape(mu, [len(mu), 1])
    sigma = (np.max(x_vector) - np.min(x_vector)) / (K - 1)

    basis = np.zeros([len(x_vector), K])
    for i in range(K):
        basis[:, i] = np.exp(-0.5 * ((x_vector - mu[i]) / sigma) ** 2)


    x_gbsf_matrix = basis
    x_gbsf_matrix = np.column_stack([x_gbsf_matrix, x_dummy])  # dummy data 추가

    '''경사하강법 이용한 w 구하기'''

    for i in range(epoch):
        if i == 0:
            w = np.random.rand(K+1) * 5

        y_hat = np.dot(x_gbsf_matrix, w).reshape([len(y_vector), 1])
        error = y_hat - y_vector

        MSE = np.mean(error ** 2)

        w_dif = sum(2 * error * x_gbsf_matrix)/len(y_hat)
        w = w - lr * w_dif

    return w, x_gbsf_matrix
```

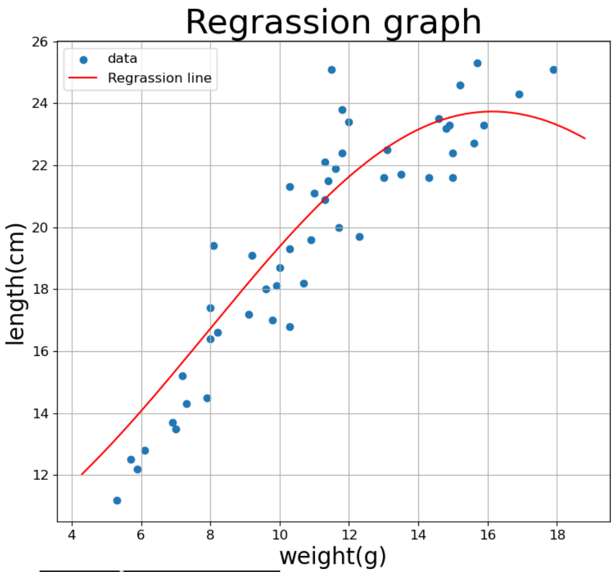
1) Analytic solution 대신 경사하강법 이용함
2) Weight 개수 K+1 개로 설정(dummy data 추가)

#과제 4

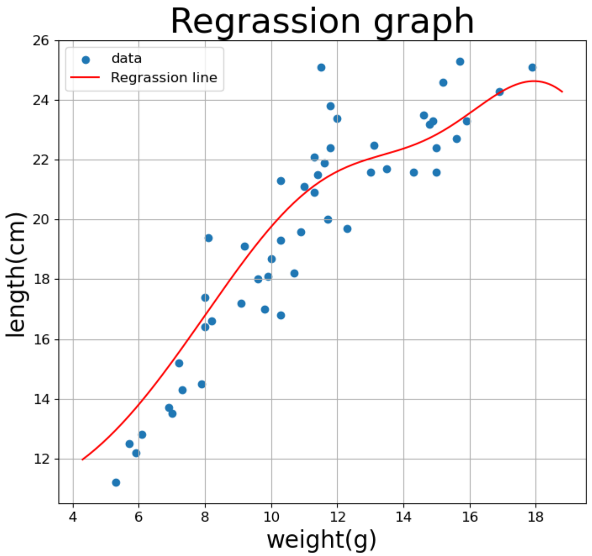
Gaussian basis fucntion을 이용한 GDM으로 regression 하기

```
Weights for k = 3:
w0: -2.8571
w1: 8.7899
w2: 9.6781
w3: 8.2447

Weights for k = 6:
w0: -0.3855
w1: 1.8397
w2: 3.9572
w3: 6.5436
w4: 0.9630
w5: 12.1409
w6: 10.6261

Weights for k = 8:
w0: -0.9646
w1: 3.9343
w2: 2.9148
w3: 2.8052
w4: 8.8902
w5: 2.1435
w6: 5.9730
w7: 11.1851
w8: 9.2183
```

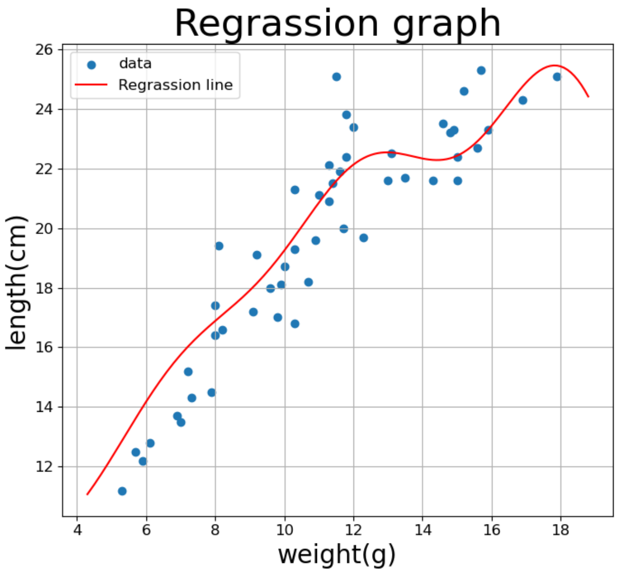
k = 3



k = 6



k = 8

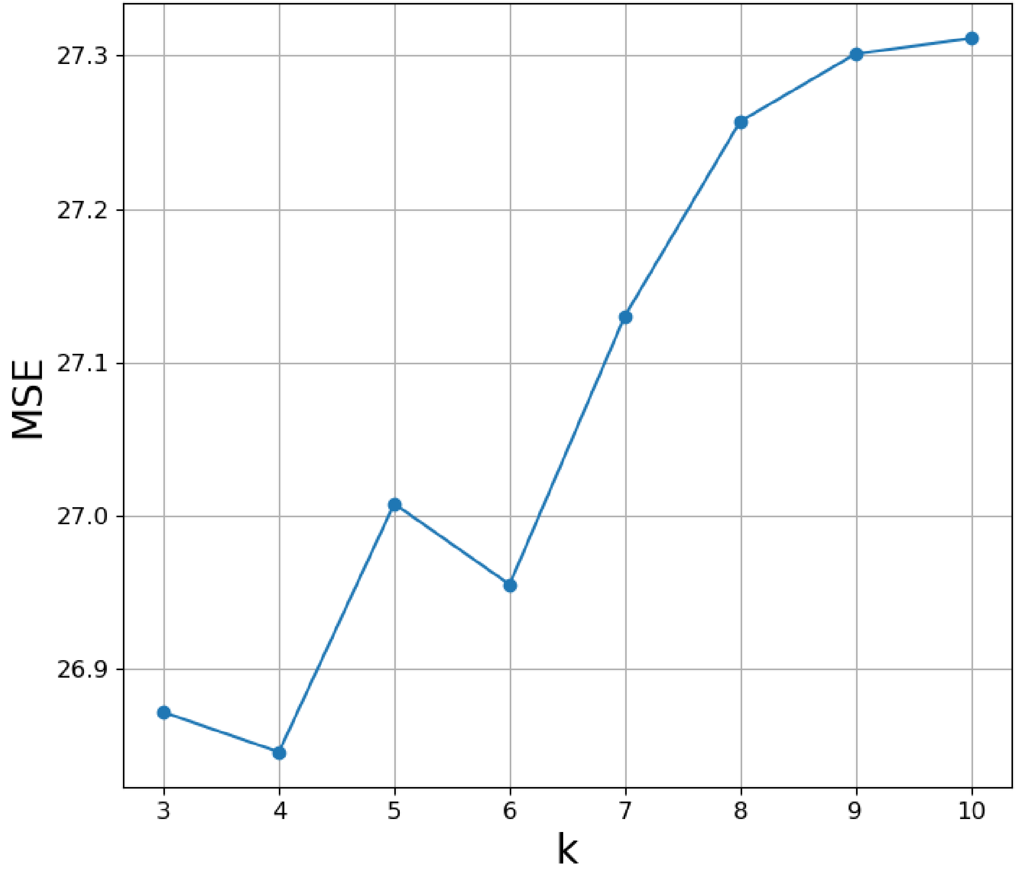



```
epoch = 6000
lr = 0.05
```

경사하강법으로 작성하여 regression그래프를 그림
Learning_rate와 epoch조절해가며 그래프 그려봤지만 weight가 작을 때 값이 제대로 작동하지 않음 → 코드에 이상이 있는 것으로 보임

#과제 4

Gaussian basis fucntion을 이용한 GDM으로 regression 하기



제대로 된 regression이 되지 않아 MSE값이 상대적으로 큰 것을 볼 수 있음

실패 이유

1. 경사하강법으로 구한 weight의 값이 정확하지 않음. ->유력

#과제4

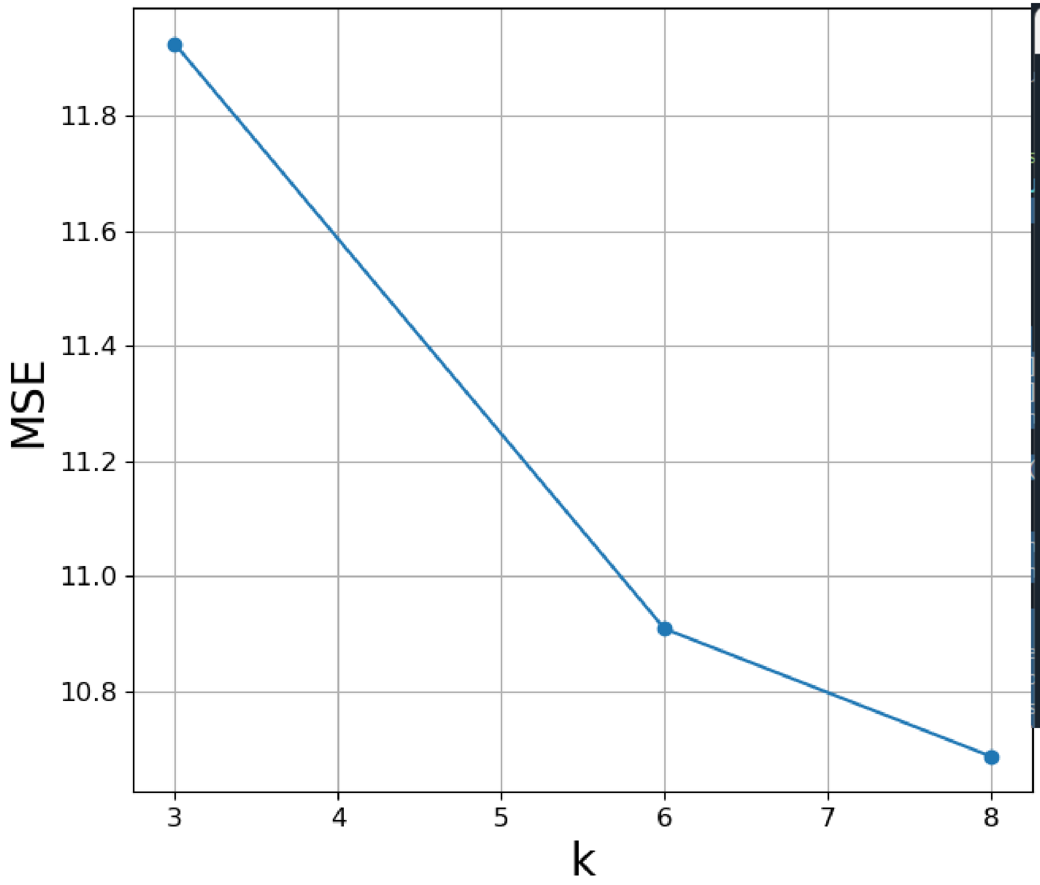
특징 2개, y값 하나인 데이터에 대해 Gaussian basis function을 이용한 analystic solution을 구하기

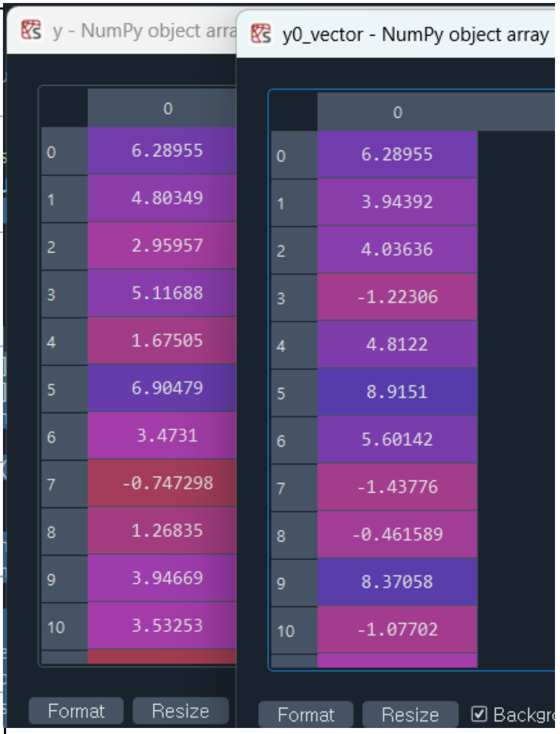
```
def GBSF_3d_ANALYSTIC_SOLUTION(x, K):
    x_gbsf_matrix = []

    # x = x_work3_matrix
    # K = 8
    k = np.arange(K)
    mu = np.zeros([2, K])
    mu[0] = np.min(x[:, 0]) + ((np.max(x[:, 0]) - np.min(x[:, 0])) / (K - 1)) * k
    mu[1] = np.min(x[:, 1]) + ((np.max(x[:, 1]) - np.min(x[:, 1])) / (K - 1)) * k
    sigma = (np.max(x) - np.min(x)) / (K - 1)

    basis = np.zeros([len(x),len(x), K])
    # basis_list =

    for i in range(K):
        basis[:, 0, i] = np.exp(-0.5 * ((x[:, 0] - mu[0, i]) / sigma) ** 2)
        basis[0, :, i] = np.exp(-0.5 * ((x[:, 1] - mu[1, i]) / sigma) ** 2)

    x_gbsf_matrix = basis  # Low방향으로 k제곱 한것 쌓기
    x_work3_dummy = np.ones([len(x_gbsf_matrix), len(x_gbsf_matrix), 1])
    x_gbsf_matrix = np.append(x_gbsf_matrix, x_work3_dummy, axis = 2)  # dummy data 추가
    x_gbsf_matrix = np.reshape(x_gbsf_matrix, [len(x_gbsf_matrix), -1])
    x_gbsf_matrix_T = np.transpose(x_gbsf_matrix)  # transpose
    w = np.dot(np.dot(np.linalg.pinv(np.dot(x_gbsf_matrix_T, x_gbsf_matrix)),x_gbsf_matrix_T), y0_vector)   # analystic solution

    y = np.dot(x_gbsf_matrix, w)
    # w = np.reshape(w, [len(x_gbsf_matrix), K + 1])
    return w, x_gbsf_matrix
```

1) 3차원이기 때문에 차원축 기준으로 평균값 2개를 구함
2) 이를 기반으로 y예측값 구하기

#과제4

특징 2개, y값 하나인 데이터에 대해 Gaussian basis function을 이용한 analytic solution을 구하기





y - NumPy object array

	0
0	6.28955
1	4.80349
2	2.95957
3	5.11688
4	1.67505
5	6.90479
6	3.4731
7	-0.747298
8	1.26835
9	3.94669
10	3.53253

Format Resize

y0_vector - NumPy object array

	0
0	6.28955
1	3.94392
2	4.03636
3	-1.22306
4	4.8122
5	8.9151
6	5.60142
7	-1.43776
8	-0.461589
9	8.37058
10	-1.07702

Format Resize Backgr

그래프 그리기, 예측용 데이터를 이용한 데이터 예측은 실패함

기저 함수 개수가 많아질수록 MSE값은 점점 수렴함을 볼 수 있음

실패 이유 예상

1. 차원2개에 따른 가우시안 기저함수의 꼴의 코드를 만들지 못함.
2. 예측용 데이터를 제대로 작성하지 못함.