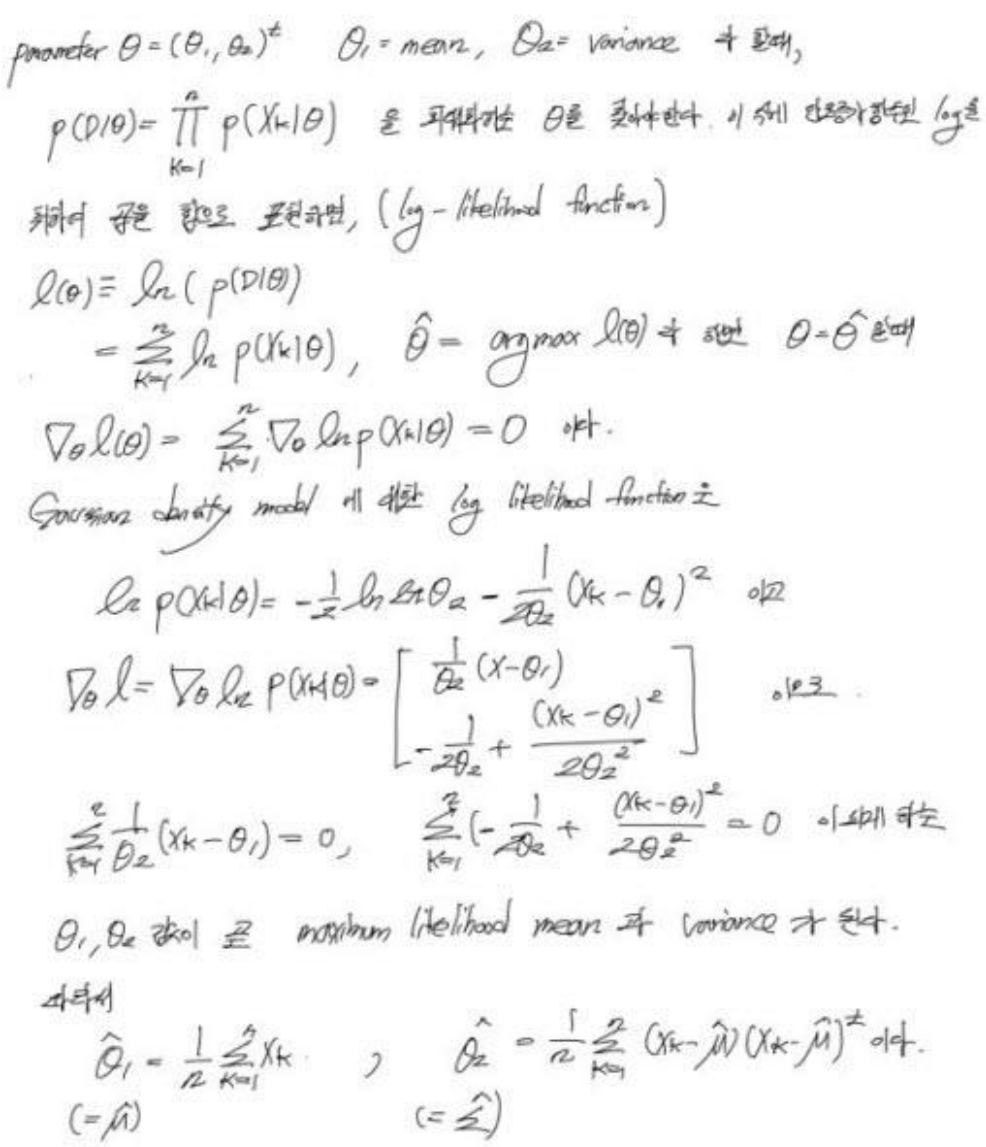
Ⅰ.이론 및 Code 구현 방법

① Maximum likelihood estimation

Gaussian density model : $p(x) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu)^t \Sigma^{-1}(x-\mu)\right]$

주어진 Sample들의 분포가 각 feature 마다 dimension Gaussian density model을 따른다고 가정할 때, maximum likelihood estimation(이하 MLE)을 통해 model의 parameter인 mean과 variance을 구하고자 한다. 이는 liklihood function 의 값을 maximize하는 parameter를 찾음으로써 구할 수 있으며, 과정은 다음과 같다.

Likelihood of mean,variance with respect to the set of samples



위에서 볼 수 있듯이, sample들의 mean과 variance가 곧 MLE를 통해 추정한 Gaussian 분포의 mean과 variance임을 알 수 있다. 이러한 parameter estimation이 올바른지 확인하기 위해 log-likelihood function이 최대 값을 가지게 하는 θ_1, θ_2 값을 찾고 sample들의

mean, variance와 비교해 보았다. 위 수식전개에선 미분을 사용하여 log-liklihood function의 1차 미분식의 값을 0으로 만들어 주는 θ_1,θ_2를 찾았지만 code로 미분을 구현할 수 없으므로(∵연속적인 parameter 변화를 구현할 수 없으므로) grid search를 통해 최적의 θ_1,θ_2를 찾았다.

② Code 구현

Code상으로 구현한 과정을 나열하면 다음과 같다.

1) Gaussian density model

```
def Gaussian_density_probability(mean, variance, data):

    sigma = sqrt(variance)
    temp = data-mean
    temp = temp/sigma
    Gaussian_constant = 1 / sqrt(2 * pi)
    probability = (Gaussian_constant/sigma)*exp(-(1/2)*(np.power(temp,2)))

    return probability
```

```
def multi_dimension_Gaussian_density_probability(mean,cov_matrix,data,dimension):

    det = np.linalg.det(cov_matrix)
    if det <0:
        probability = np.zeros((data.shape[1],data.shape[1]))
    elif det >0:
        cov_inverse = np.linalg.inv(cov_matrix)
        temp = data- mean
        constant = 1/((sqrt(2*pi)**dimension)*sqrt(det))
        probability =constant*exp((-1/2)*np.dot(np.dot(temp.T,cov_inverse),temp))

    return probability
```

1차 Gaussian model과 multi-dimension gaussian model을 구현한 코드이다. 이를 따로 구현한 이유는, multi-dimension gaussian model의 covariance matrix에서 발생 가능한 예외사항을 따로 처리하기 위함이다.(뒤에서 자세히 논한다.)

2) Log-likelihood function

```
def log_likelihood_function(probability):
    log_probability = log(probability)
    log_probability_sum = np.sum(log_probability,axis=0)
    return log_probability_sum
```

Log likelihood function 으로, Gaussian density model에서 계산한 probability에 log를 취한 후 값을 summation하여 반환 한다.

3) Mean, variance value for grid search

```
mean_resolution_1 = 100
mean_resolution_2 = 100
mean_resolution_3 = 100
var_resolution_1 = 10
var_resolution_2 = 10
var_resolution_3 = 10

mean_value1 = np.linspace(-1,1,num=mean_resolution_1,endpoint=False)
mean_value2 = np.linspace(-0.61,-0.6,num=mean_resolution_2,endpoint=False)
mean_value3 = np.linspace(-1,-0.9,num=mean_resolution_3,endpoint=False)

var_value1 = np.linspace(4, 4.5, num=var_resolution_1, endpoint=False)
var_value2 = np.linspace(4, 4.5, num=var_resolution_2, endpoint=False)
var_value3 = np.linspace(0.3, 1, num=var_resolution_3, endpoint=False)
```

Grid search를 할 때 필요한 mean, variance를 생성하기 위해 numpy의 linspace 함수를 사용하였다. 이를 통해 mean과 variance가 특정 범위 내에서 일정한 간격으로 생성될 수 있도록 하였으며, 이를 vector화 하여 Gaussian model에 대입하였다. 대입할 mean, variance의 시작, 끝 범위와 resolution을 정할 수 있으므로 이론 값 근처에서 결과 값을 도출 하였다.

전체 code에서 이런 식으로 grid search에 사용된 변수는 최대 6개 이며, 그 이유는 Gaussian density 모델의 dimension이 높아 질수록 covariance matrix를 구성하는 원소들의 개수가 많아지기 때문이다. 예를 들어 3차 gaussian density 모델의 covariance matrix는 3x3 으로 총 9개의 원소를 가지는데, 대칭행렬이므로 6개의 변수를 통해 지정할 수 있다.

4) 최대값 산출 방법

```
def maximum_likelihood_mean_2d(mean_value_1,mean_value_2,cov_mat,data):

    mean_value = Mean_matrix_2d(mean_value_1,mean_value_2)
    result_buffer = np.zeros((mean_value.shape[0]))
    for index,mean in enumerate(mean_value):
        probability = multi_dimension_Gaussian_density_probability(mean.reshape((2,1)),cov_mat,data,2)
        prob_diag = np.diag(probability)
        result_buffer[index] = log_likelihood_function(prob_diag)

    MLE_mean = mean_value[np.argmax(result_buffer)]

    return MLE_mean
```

위 그림은 MLE mean을 산출하는 함수이다. 먼저, 3)에서 대입할 mean 값들의 범위와 resolution을 정한 후 이들을 Gaussian model에 대입하여 log likelihood 값을 계산 한다. 이 후, 계산된 값들 중 최대 값을 가지는 mean 값을 반환 한다. 예를 들어,2 dimension gaussian model에 대해 추정해야 하는 mean 값은 2개 이므로, 각 mean에 대해 10개의 grid를 생성하였다면 총 100쌍의 mean vector에 대해 log likelihood를 계산 한다. 이 값들은 result_buffer라는 array에 저장된 후, 저장된 값들 중 최대 값을 갖는 mean vector를 찾아 MLE mean으로 추정 한다. Variance와 covariance matrix 또한 같은 방식으로 MLE 값을 추정 한다. (단, dimension이 높아져 연산 결과를 한 array에 저장하기 힘든 경우 divde and conquer 방식으로 분할 연산을 하였다.)

Ⅱ. 구현 결과

① (문제 (a)) w_1의 각 feature x_1, x_2, x_3 에 대해 μ, σ^2 추정

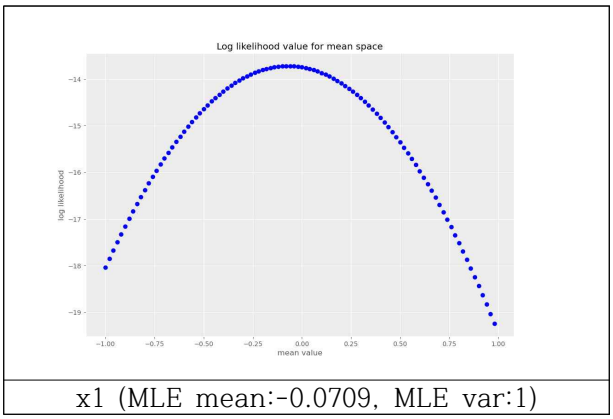
```
mean_resolution_1 = 100
mean_value1 = np.linspace(-1,1,num=mean_resolution_1,endpoint=False)

var_resolution_1=50
var_resolution_2=50
var_resolution_3=50

var_value1 = np.linspace(0.8, 1.2, num=var_resolution_1, endpoint=False)
var_value2 = np.linspace(4, 4.5, num=var_resolution_2, endpoint=False)
var_value3 = np.linspace(4.5, 4.7, num=var_resolution_3, endpoint=False)
var_value = np.vstack((var_value1,var_value2,var_value3))

for index in range(data.shape[0]):
    MLE_mean =maximum_likelihood_mean(mean_value1,1,index)
    MLE_variance = maximum_likelihood_variance(MLE_mean,var_value[index],index)
    print("Feauter index : ",index+1)
    print("Mean : ",MLE_mean)
    print("Variance : ", MLE_variance)
```

MLE mean 값은 variance를 1로 고정한 후, mean값을 -1부터 1까지의 값을 100개로 나누어 대입하여 얻은 값 중 최대 값을 산출하여 구하였다. 마찬가지로, MLE variance의 값은 앞서 구한 MLE mean 값과 함께 각 feature마다 특정 범위 내에서 50개로 나눈 값을 대입한 후 최대 값을 산출하여 구하였다. 실제 data set의 mean, variance 값과 resolution에 따른 추정 값 변화는 다음과 같다.

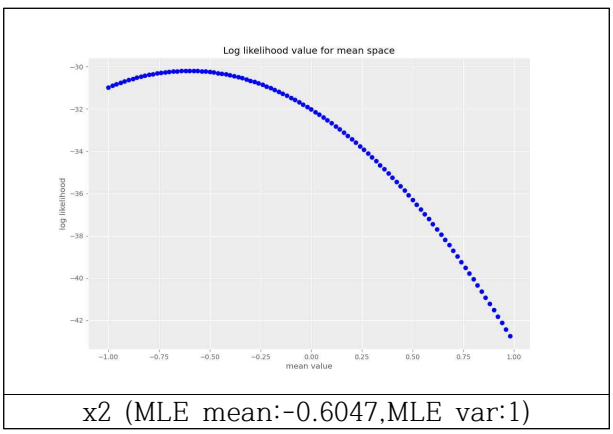
Resolution (Mean =100, Var =50)	Resolution (Mean =500, Var =500)
Problem (a) Feature index : 1 Mean : -0.07999999999999996 Variance : 0.904 Feature index : 2 Mean : -0.6 Variance : 4.2 Feature index : 3 Mean : -0.92 Variance : 4.5440000000000005	Problem (a) Feature index : 1 Mean : -0.07199999999999995 Variance : 0.9064 Feature index : 2 Mean : -0.604 Variance : 4.201 Feature index : 3 Mean : -0.912 Variance : 4.542

	x1	x2	x3
Sample mean	**-0.0709**	**-0.6047**	**-0.9110**
MLE mean(resolution:100)	-0.0799	-0.6	-0.92
MLE mean(resolution:500)	-0.07199	-0.604	-0.912
Sample variance	**0.9062**	**4.2007**	**4.5419**
MLE var(resolution:50)	0.904	4.2	4.5440
MLE var(resolution:500)	0.9064	4.201	4.542

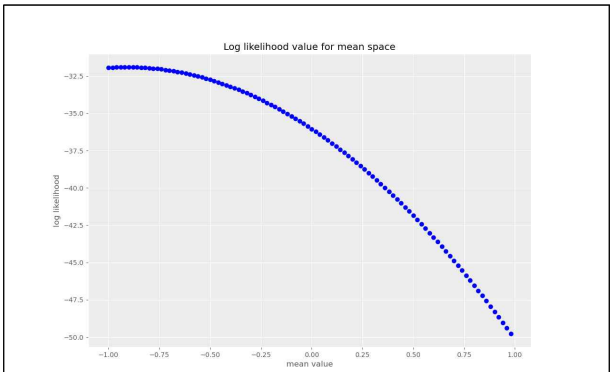
위 표에 나타난 것과 같이, linspace의 resolution이 증가할수록(= grid의 간격이 줄어들수록) 실제 sample mean, variance 값에 근접한 값으로 추정하는 것을 알 수 있었다. linspace를 통해 만들어진 mean 값들(mean space)의 변화에 따른 log likelihood function 값을 plot하면 다음과 같다.



x1 (MLE mean:-0.0709, MLE var:1)



x2 (MLE mean:-0.6047,MLE var:1)



x3 (MLE mean:-0.9110,MLE var: 1)

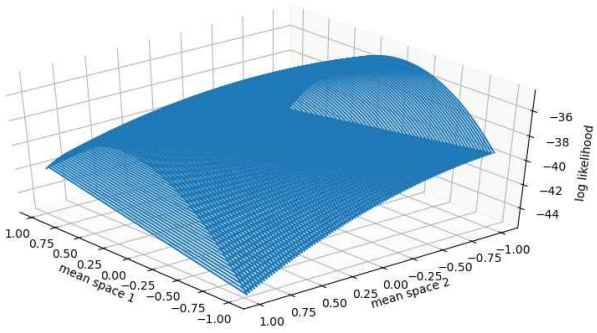
위 plot들로 부터 알 수 있듯이, MLE mean 값에서 log likelihood 값은 최대값을 가진다.

② (문제 (b)) w_1의 각 feature 쌍 $(x_1,x_2),(x_1,x_3),(x_2,x_3)$ 에 대해 two dimension mean vector 와 covariance matrix Σ 추정

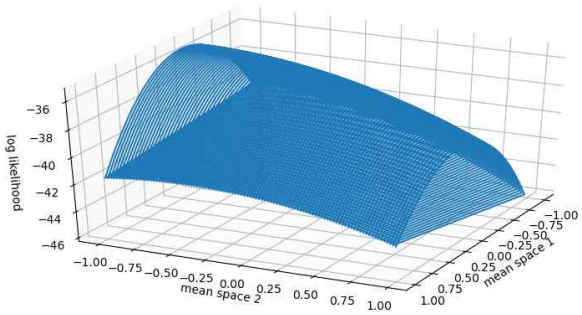
multi- dimension Gaussian model 은 mean vector 와 covariance matrix를 추정해야 하는데, 먼저 식의 covariance matrix를 각 feature data간의 covariance matrix로 고정한 후 mean vector를 추정하였다.

```
Cov_mat1 = Covariance_matrix_2d(data[0],data[1])
Cov_mat2 = Covariance_matrix_2d(data[0],data[2])
Cov_mat3 = Covariance_matrix_2d(data[1],data[2])

MLE_mean1 = maximum_likelihood_mean_2d(mean_value1,mean_value2,Cov_mat1,data[0:2])
MLE_mean2 = maximum_likelihood_mean_2d(mean_value1,mean_value3,Cov_mat2,np.vstack((data[0],data[2])))
MLE_mean3 = maximum_likelihood_mean_2d(mean_value2,mean_value3,Cov_mat3,data[1:])
print("MLE mean for feature 1,2 : ",MLE_mean1)
print("MLE mean for feature 1,3 : ",MLE_mean2)
print("MLE mean for feature 2,3 : ",MLE_mean3)

var_resolution_1 = 30
var_resolution_2 = 30
var_resolution_3 = 30

var_value1 = np.linspace(0.8, 1.2, num=var_resolution_1, endpoint=False)
var_value2 = np.linspace(4, 4.5, num=var_resolution_2, endpoint=False)
var_value3 = np.linspace(0.5, 0.7, num=var_resolution_3, endpoint=False)
MLE_variance1 = maximum_likelihood_var_2d(MLE_mean1,data[0:2])
print("MLE Covariance matrix for feature 1,2 : ")
print(MLE_variance1)
```

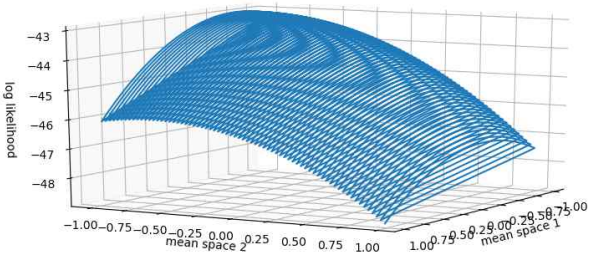
각 data 쌍의 mean space 변화 값에 따른 log likelihood 값을 시각화 하면 다음과 같다.



x1,x2 (MLE mean:-0.072,-0.6054)



x1,x3 (MLE mean:-0.072,-0.911)

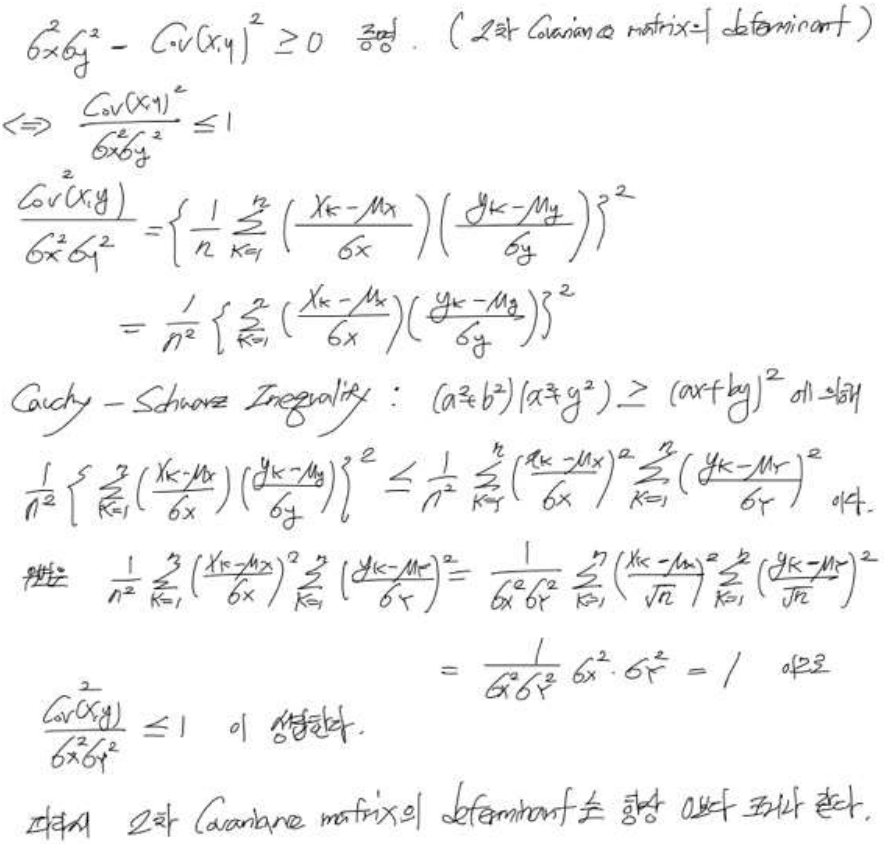


x2,x3 (MLE mean:-0.6047,-0.911)

이 후, 앞서 구한 MLE mean vector와 임의로 지정한 covariance matrix를 대입하여 최대값을 산출하는 covariance matrix를 구하였다. Covariance matrix를 임의로 지정한 방법은 다음과 같다.

$$Cov = \begin{bmatrix} \sigma_x^2 & cov(x,y) \\ cov(x,y) & \sigma_y^2 \end{bmatrix} = \begin{bmatrix} Var linspace_1 & Var linspace_2 \\ Var linspace_2 & Var linspace_3 \end{bmatrix}$$

Covariance matrix는 대칭 행렬이므로, 2 x 2 matrix를 결정하기 위해선 3개의 값이 필요하다. linspace로 3개의 grid를 생성한 후, 각 값들을 대입하여 matrix를 생성한다. 예를 들어, 위 그림의 경우 각 linspace 마다 30개의 variance 값을 가지므로 총 27000개의 covariance matrix를 생성 하여 대입한 것이다. (Var_matrix_2d, Var_matrix_3d 라는 함수로 정의하여 구현하였다.) 그러나 이렇게 covariance matrix를 무작위로 생성하여 대입하면 문제가 생기게 되는데 , 바로 matrix의 determinant가 음수가 되는 경우가 발생한다는 것이다. 2차 Covariance matrix의 determinant 는 $|\Sigma| = \sigma_x^2\sigma_y^2 - cov(x,y)^2$ 이며, 이 값은 음수가 될 수 없다. 증명은 다음과 같다.



따라서 설정한 grid의 임의의 조합을 통해 만들어진 matrix 중 determinant 값이 음수 인 것은 covariance matrix가 될 수 없으므로 제외 시켜주어야 한다. 이를 위해, Ⅰ-②에서 언급하였던 것과 같이 multi Gaussian density function의 연산에 대해선 covariance matrix의 determinant의 부호를 판단하는 과정이 선행되어야 한다. 각 feature 쌍들의 MLE 결과 값은 다음과 같다.

```
Problem (b)
MLE mean for feature 1,2 :  [-0.072  -0.6054]
MLE mean for feature 1,3 :  [-0.072 -0.911]
MLE mean for feature 2,3 :  [-0.6047 -0.911 ]
MLE Covariance matrix for feature 1,2 :
[[0.90666667 0.56666667]
 [0.56666667 4.2       ]]
MLE Covariance matrix for feature 1,3 :
[[0.90666667 0.394     ]
 [0.394      4.54      ]]
MLE Covariance matrix for feature 2,3 :
[[4.2        0.73333333]
 [0.73333333 4.54333333]]
```

문제 (a)에서 구한 MLE mean과 variance의 값들과 유사한 값이 나온 것을 통해, 올바르게 추정된 것을 알 수 있다.

③ (문제 (c)) w_1의 feature 쌍 (x_1, x_2, x_3) 에 대해 three dimension mean vector와 covariance matrix Σ 추정

문제 (b)를 해결한 방식과 같은 방식으로 mean과 covariance matrix를 추정하였다. 다른 점

```
print("============================================================")
print("Problem (c)")
MLE_mean_3d = maximum_likelihood_mean_3d(mean_value1,mean_value2,mean_value3,data)
print("Mean : ",MLE_mean_3d)
print("Covariance matrix:")
print(maximum_likelihood_var_3d(MLE_mean_3d,data))
print("Sample covariance matrix")
print(Covariance_matrix_3d(data[0],data[1],data[2]))
print("============================================================")
```

```
Problem (c)
Mean :  [-0.08   -0.6048 -0.912 ]
Covariance matrix:
[[0.906  0.568  0.39  ]
 [0.568  4.2008 0.73  ]
 [0.39   0.73   4.54  ]]
Sample covariance matrix
[[0.90617729 0.56778177 0.3940801 ]
 [0.56778177 4.20071481 0.7337023 ]
 [0.3940801  0.7337023  4.541949  ]]
```

이 있다면, 3x3 covariance matrix를 추정하는데 6개의 grid 조합이 필요하므로 resolution 이 증가함에 따라 가능한 조합의 수는 6승의 크기로 증가하기 때문에, grid의 range를 기존보다 작게 설정하고 resolution의 크기를 낮추어 추정치의 정확도를 유지하였다. Mean vector 는 문제 (a)와 문제 (b)에서 추정한 값과 비슷하게 추정되었으며, covariance matrix 또한 sample의 covariance matrix와 비교하였을 때 값이 일치함을 알 수 있었다.

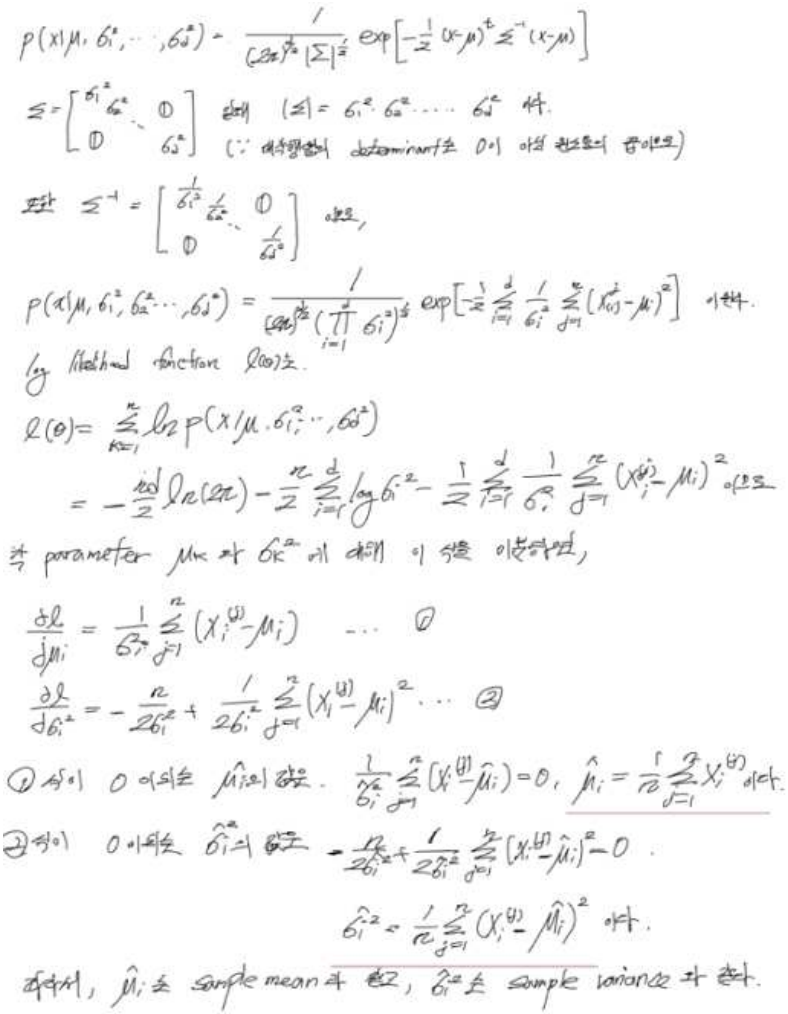
④ (문제 (d)) w_2의 feature x_1, x_2, x_3가 서로 독립이라고 가정하고 (Covariance matrix가 diagonal 하다고 가정) three dimension mean vector와 covariance matrix Σ 추정

Covariance matrix가 diagonal 해야 하므로, mean vector를 추정할 때엔 covariance matrix를 identity matrix로 가정하였다. 이후, MLE mean vector를 이용하여 covariance matrix를 추정할 때엔 문제 (c) 와 같은 방법을 사용하되 아래 코드와 같이 대각성분이 아닌 성분들의 grid 범위를 0으로 설정하여 주었다.

```
var_value_31 = np.linspace(0., 0., num=var_resolution_1, endpoint=False)  # [1,3]
var_value_32 = np.linspace(0., 0., num=var_resolution_2, endpoint=False)  # [2,3]
var_value_33 = np.linspace(0., 0., num=var_resolution_3, endpoint=False)  # [1,2]
```

```
Problem (d)
Mean :  [-0.12  0.44  0.  ]
Covariance matrix:
[[0.054  0.     0.    ]
 [0.     0.046  0.    ]
 [0.     0.     0.0074]]
Sample Covariance matrix:
[[ 0.05392584 -0.01465126 -0.00517993]
 [-0.01465126  0.04597009  0.00850987]
 [-0.00517993  0.00850987  0.00726551]]
```

위의 추정된 결과로 부터, covariance matrix의 대각 성분은 sample covariance matrix의 대각성분과 일치하는 것을 알 수 있다. 수식을 통해 covariance matrix가 diagonal 할때의 mean likelihood estimation을 전개해 보면 다음과 같다.



이를 통해, Gaussian model이 각 feature들에 대해 seperable 하게 mean과 variance를 추정하기 때문에 feature 간 covariance 값에 관계 없이 각 feature의 variance (=대각 성분) 추정 값은 같다는 것을 알 수 있다.

⑤ (문제 (e)) 각 w_i의 feature x_1, x_2, x_3에 대해 maximum likelihood mean vector 를 추정한 후, sample mean vector와 비교하기

```
Problem (e)
dataset 1 mean vector:  [-0.0709 -0.6047 -0.911 ]
MLE mean of dataset 1:  [-0.08 -0.6  -0.92]
Difference:  [0.0091 0.0047 0.009 ]
dataset 2 mean vector:  [-0.1126   0.4299   0.00372]
MLE mean of dataset 2:  [-0.12  0.44  0.  ]
Difference:  [0.0074  0.0101  0.00372]
dataset 3 mean vector:  [0.2747 0.3001 0.6786]
MLE mean of dataset 3:  [0.28 0.32 0.68]
Difference:  [0.0053 0.0199 0.0014]
```

위 그림은 sample data의 mean vector와 MLE mean vector, 그 둘의 차이 값을 나열한 것이다. 이를 통해 log likelihood function 값을 최대화 하는 mean 값과 sample mean 값

이 거의 일치함을 알 수 있다. Grid search 방식을 통해 찾았기 때문에 약간의 오차가 존재하지만, grid의 resolution을 높일 수록 오차는 0에 수렴한다.

⑥ (문제 (f)) 각 w_i의 feature x_1, x_2, x_3에 대해 maximum likelihood covariance matrix를 추정한 후, sample covariance matrix와 비교하기

dataset 1 Cov matrix: [[0.90617729 0.56778177 0.3940801] [0.56778177 4.20071481 0.7337023] [0.3940801 0.7337023 4.541949]] MLE Cov matrix of dataset 1: [[0.906 0.568 0.39] [0.568 4.2008 0.73] [0.39 0.73 4.54]] Difference: [[1.7729e-04 2.1823e-04 4.0801e-03] [2.1823e-04 8.5190e-05 3.7023e-03] [4.0801e-03 3.7023e-03 1.9490e-03]]	dataset 2 Cov matrix: [[0.05392584 -0.01465126 -0.00517993] [-0.01465126 0.04597009 0.00850987] [-0.00517993 0.00850987 0.00726551]] MLE Cov matrix of dataset 2: [[0.054 -0.014 -0.0052] [-0.014 0.046 0.0084] [-0.0052 0.0084 0.0074]] Difference: [[7.416000e-05 6.512600e-04 2.007200e-05] [6.512600e-04 2.991000e-05 1.098720e-04] [2.007200e-05 1.098720e-04 1.344944e-04]]	dataset 3 Cov matrix: [[0.30186081 0.40474153 -0.18042342] [0.40474153 0.64496409 -0.20130386] [-0.18042342 -0.20130386 1.26214164]] MLE Cov matrix of dataset 3: [[0.32 0.42 -0.2] [0.42 0.64 -0.23] [-0.2 -0.23 1.28]] Difference: [[0.01813919 0.01525847 0.01957658] [0.01525847 0.00496409 0.02869614] [0.01957658 0.02869614 0.01785836]]
w_1	w_2	w_3

위 표는 sample data의 covariance matrix와 MLE covariance matrix, 그 둘의 차이 값을 나열한 것이다. 이를 통해 log likelihood function 값을 최대화 하는 covariance 값과 sample covariance 값이 거의 일치함을 알 수 있다. 문제 (e) 와 (f)에 대한 증명은 I -①에서 다루었다.