120200153 이성훈
2020/10/08

HW#1 chap3

Problem A

Write a program to find the maximum-likelihood values $\hat{\mu}$ and $\hat{\sigma}^2$. Apply your program individually to each of the three features x_i of category w_1 in the table above.

```
print("Problem A")
x, k, mu, sig = symbols('x k mu sig')
diff_mu = diff(-log(2 * pi.evalf() * sig) / 2 - ((x - mu) ** 2) / (2 * sig), mu)
diff_sig = diff(-log(2 * pi.evalf() * sig) / 2 - ((x - mu) ** 2) / (2 * sig), sig)

print("mu, sigma derivate results")
print(diff_mu)
print(diff_sig)

diff_mu = -(10 * mu - w1x1.sum())/sig
diff_sig = -5/sig + (mu**2 - 2 * mu * w1x1.sum() + (w1x1**2).sum())/(2*sig**2)

print()
print("put value manually")
print(diff_mu)
print(diff_sig)


mu = float(solve(Eq(diff_mu, 0), mu)[0])
x_minus_mu = w1x1 - mu
diff_sig = -5/sig + ((x_minus_mu**2).sum())/(2*(sig**2))


print("solve equation and comparison for w1 x1")
print("mean = {}".format(mu))
print("real mean = {}".format(w1x1.mean()))
print("sigma = {}".format(solve(Eq(diff_sig, 0), sig)[0]))
print("real sigma = {}".format((w1x1**2).mean() - w1x1.mean()**2))
```

```
Problem A
mu, sigma derivate results
-(2*mu - 2*x)/(2*sig)
-0.5/sig + (-mu + x)**2/(2*sig**2)

put value manually
(-10*mu - 0.709)/sig
-5/sig + (mu**2 + 1.418*mu + 9.112041)/(2*sig**2)
solve equation and comparison for w1 x1
mean = -0.0709
real mean = -0.07089999999999999
sigma = 0.906177290000000
real sigma = 0.9061772899999999


solve equation and comparison for w1 x2
mean = -0.6047
real mean = -0.6047
sigma = 4.20071481000000
real sigma = 4.20071481


solve equation and comparison for w1 x3
mean = -0.911
real mean = -0.9109999999999999
sigma = 4.54194900000000
real sigma = 4.541949000000001
```

Figure 1: Problem A's code and result

$$\nabla_\theta l = \nabla_\theta ln(p(x_k|\theta)) = \begin{bmatrix} \frac{1}{\theta_2}(x_k - \theta_1) \\ -\frac{1}{2\theta_2} + \frac{(x_k-\theta_1)^2}{2\theta_2^2} \end{bmatrix} \tag{1}$$

Figure 1는 Problem A의 코드와 결과를 나타낸 것이다. Univariate gaussian 분포를 가지는 data라고 가정하면, log-likelihood의 미분값을 Equation 1와 같이 구할 수 있다. 강의에서와 동일하게 θ_1은 mean을, θ_2는 variance를 나타낸다. Log-likelihood의 maximum 값을 가지도록 하기위해 Equation 1가 0이 되도록 하는 μ와 σ를 구할 수 있게된다. 위 방법을 python module인 "sympy"를 이용하여 log-likelihood를 미분한 식을 구하여 Figure 1 결과창의 상단 부분과 같이출력한다. 변수에 대한 summation 기능은 찾지 못하여 출력한 식을 바탕으로 직접 해당하는 값을 넣어주어 먼저 Equation 1의 1행 식을 통해 μ를 구한다. 구한 μ를 다시 1의 아래 식을 통해 σ에 대한 방정식을 세우고 문제를 풀면 Figure 1의 우측과 같은 결과가 나온다. 이러한 과정을 x1, x2, x3에 대해 반복하였으며 코드는 너무 길어지기 때문에 x1에 대해서만 첨부하였다.

Problem B

Modify your program to apply to two-dimensional Gaussian data $p(x) \sim N(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Apply your data to each of the three possible pairings of two features for w_1.

$$l(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\boldsymbol{x_k}) = \Sigma_{k=1}^{m}(-\frac{p}{2}log(2\pi) - \frac{1}{2}log|\boldsymbol{\Sigma}| - \frac{1}{2}(\boldsymbol{x_k} - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{x_k} - \boldsymbol{\mu})) \tag{2}$$

$$\frac{\partial}{\partial \boldsymbol{\Sigma}} l(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\boldsymbol{x_k}) = \frac{n}{2}\boldsymbol{\Sigma} - \frac{1}{2}\Sigma_{k=1}^{n}(\boldsymbol{x_k} - \boldsymbol{\mu})(\boldsymbol{x_k} - \boldsymbol{\mu})^T \tag{3}$$

$$\boldsymbol{\mu} = \frac{1}{n}\Sigma_{k=1}^{n}\boldsymbol{x_k}, \boldsymbol{\Sigma} = \frac{1}{n}\Sigma_{k=1}^{n}(\boldsymbol{x_k} - \boldsymbol{\mu})(\boldsymbol{x_k} - \boldsymbol{\mu})^T \tag{4}$$

```
print("Problem B, let we already know mean and covariance equation")
print("case x1, x2")

mean_mat = np.array([w1x1.mean(), w1x2.mean()]).T
print("mean matrix [mu1, mu2] = ", end="")
print(mean_mat)

covar_mat = np.zeros((2,2))
for idx in range(10):
    x_mu_sub_mat = np.array([[w1x1[idx], w1x2[idx]]]) - mean_mat.T
    covar_mat += np.matmul(x_mu_sub_mat.T, x_mu_sub_mat)
    # print(np.matmul(x_mu_sub_mat.T, x_mu_sub_mat))
covar_mat /= 10
print("covariance matrix = ", end="")
print(covar_mat)
```

```
Problem B, let we already know mean and covariance equation
case x1, x2
mean matrix [mu1, mu2] = [-0.0709 -0.6047]
covariance matrix = [[0.90617729 0.56778177]
 [0.56778177 4.20071481]]

case x1, x3
mean matrix [mu1, mu3] = [-0.0709 -0.911 ]
covariance matrix = [[0.90617729 0.3940801 ]
 [0.3940801  4.541949  ]]

case x3, x2
mean matrix [mu3, mu2] = [-0.911  -0.6047]
covariance matrix = [[4.541949   0.7337023 ]
 [0.7337023  4.20071481]]
```

Figure 2: Problem B's code and result

```
syms x11
syms x12
syms x21
syms x22
x1 = [0.42 -0.2 1.3 0.39 -1.6 -0.029 -0.23 0.27 -1.9 0.87]
x2 = [-0.087 -3.3 -0.32 0.71 -5.3 0.89 1.9 -0.3 0.76 -1]
A = [x11 x12; x21 x22]
B = A+[1 1; 1 1]
diff(A, x12)
mu = [mean(x1) mean(x2)]

summation = 0
for k = 1: 10
    summation = summation + A * 1/2 - ([x1(k) x2(k)]-mu)' * ([x1(k) x2(k)]-mu)/2
end
x11_result = double(solve(summation(1,1)==0, x11))
x12_result = double(solve(summation(1,2)==0, x12))
x21_result = double(solve(summation(2,1)==0, x21))
x22_result = double(solve(summation(2,2)==0, x22))
```

```
x11_result =

   0.906177290000000


x12_result =

   0.567781770000000


x21_result =

   0.567781770000000


x22_result =

   4.200714810000000
```

Figure 3: Problem B's matlab code and result

Equation 2와 Equation 3은 강의자료 및 교재에 미분을 통한 증명이 나오지 않아 추가하였다. Figure 2는 Problem B의 코드와 결과이다. 코딩 과정에서 Problem A처럼 Equation 3을 실제로 미분해보고자 하였으나 matrix에 대한 미분과 방정식 풀이는 지원하지 않았다. 그래서 mean과 variance를 Equation 4식을 통해 구하였다. 실제로 미분식을 통한 계산이 필요할 것으로 판단하여 matlab을 사용하여 Equation 4 식에 맞게 matlab code를 Figure 3과 같이 작성하였다. Dimension이 2개인 데이터를 이용했기 때문에 mean은 2x1, covariance는 2x2가 됨을 확인할 수 있다. 자세히 살펴보면, x1의 평균은 x2와 pair를 이루든, x3와 pair를 이루든 pair가 없든 항상 똑같은 값을 갖게된다. 이는 평균값에 다른 dimension의 data가 관여하지 않기 때문이다. 유사하게, covariance matrix의 diagonal elements를 확인해보면, x1의 분산은 x2, x3와 pair를 이루든, pair가 없든 항상 똑같은 값을 가짐을 확인할 수 있다. 실제로 수식을 통해 본다면 Equation 4을 전개해보면 diagonal elements에는 다른 pair의 data가 관여하지 않는 것을 쉽게 확인할 수 있다.

Problem C

Modify your program to apply to three-dimensional Gaussian data. Apply your data to the full three-dimensional data for w_1.

```python
print("Problem C")

mean_mat = np.array([w1x1.mean(), w1x2.mean(), w1x3.mean()]).T
print("mean matrix [mu1, mu2, mu3] = ", end="")
print(mean_mat)

covar_mat = np.zeros((3,3))
for idx in range(10):
    x_mu_sub_mat = np.array([[w1x1[idx], w1x2[idx], w1x3[idx]]]) - mean_mat.T
    covar_mat += np.matmul(x_mu_sub_mat.T, x_mu_sub_mat)
    # print(np.matmul(x_mu_sub_mat.T, x_mu_sub_mat))
covar_mat /= 10
print("covariance matrix = ", end="")
print(covar_mat)
```

```
Problem C
mean matrix [mu1, mu2, mu3] = [-0.0709 -0.6047 -0.911 ]
covariance matrix = [[0.90617729 0.56778177 0.3940801 ]
 [0.56778177 4.20071481 0.7337023 ]
 [0.3940801  0.7337023  4.541949  ]]
```

Figure 4: Problem C's code and result

```
syms x11
syms x12
syms x13
syms x21
syms x22
syms x23
syms x31
syms x32
syms x33
x1 = [0.42 -0.2 1.3 0.39 -1.6 -0.029 -0.23 0.27 -1.9 0.87];
x2 = [-0.087 -3.3 -0.32 0.71 -5.3 0.89 1.9 -0.3 0.76 -1];
x3 = [0.58 -3.4 1.7 0.23 -0.15 -4.7 2.2 -0.87 -2.1 -2.6];
A = [x11 x12 x13; x21 x22 x23; x31 x32 x33];
mu = [mean(x1) mean(x2) mean(x3)];

summation = 0
for k = 1: 10
    summation = summation + A * 1/2 - ([x1(k) x2(k) x3(k)]-mu)' * ([x1(k) x2(k) x3(k)]-mu)/2;
end
x11_result = double(solve(summation(1,1)==0, x11))
x12_result = double(solve(summation(1,2)==0, x12))
x13_result = double(solve(summation(1,3)==0, x13))
x21_result = double(solve(summation(2,1)==0, x21))
x22_result = double(solve(summation(2,2)==0, x22))
x23_result = double(solve(summation(2,3)==0, x23))
x31_result = double(solve(summation(3,1)==0, x31))
x32_result = double(solve(summation(3,2)==0, x32))
x33_result = double(solve(summation(3,3)==0, x33))
```

```
x11_result =

   0.906177290000000


x12_result =

   0.567781770000000


x13_result =

   0.394080100000000


x21_result =

   0.567781770000000


x22_result =

   4.200714810000000


x23_result =

   0.733702300000000


x31_result =

   0.394080100000000


x32_result =

   0.733702300000000


x33_result =

   4.541949000000000
```

Figure 5: Problem C's matlab code and result

Figure 4는 Problem C의 코드와 결과이다. Problem B와 동일하게 Equation 4을 통해 three-dimensional data에 대한 mean과 variance를 구하였다. 또한 Figure 5와 같이 matlab으로 값을 확인해 보았으며 동일한 값이 나옴을 확인할 수 있다. Dimension이 3개인 데이터를 이용했기 때문에 mean은 3x1, covariance는 3x3가 됨을 확인할 수 있다. 또한, mean과 variance의 diagonal elements가 Problem A, Problem B에서 구한 값과 정확히 일치함을 확인할 수 있다. 이는 Problem B에서와 동일한 이유로 mean과 variance의 diagonal elements는 다른 pair와는 독립적인 값이기 때문이다.

Problem D

Assume your three-dimensional model is separable, so that $\Sigma = diag(\sigma_1^2, \sigma_2^2, \sigma_3^2)$. Write a program to estimate the mean and the diagonal components of Σ. Apply your program to the data in w_2.

```
print("Problem D : same as Problem 1")

x, y, z = symbols('x y z')
mean_mat = np.array([w2x1.mean(), w2x2.mean(), w2x3.mean()])
w2x1_minus_mean = w2x1 - mean_mat[0]
w2x2_minus_mean = w2x2 - mean_mat[1]
w2x3_minus_mean = w2x3 - mean_mat[2]
sig1 = 1/10 * ((w2x1_minus_mean)**2).sum()
sig2 = 1/10 * ((w2x2_minus_mean)**2).sum()
sig3 = 1/10 * ((w2x3_minus_mean)**2).sum()
covar_mat = [[sig1,0,0],
             [0,sig2,0],
             [0,0,sig3]]

print(covar_mat)
```

```
Problem D : same as Problem 1
[[0.05392584, 0, 0], [0, 0.04597009, 0], [0, 0, 0.0072655055999999996]]
```

Figure 6: Problem D's code and result

```
syms x11
syms x12
syms x13
syms x21
syms x22
syms x23
syms x31
syms x32
syms x33
x1 = [0.42 -0.2 1.3 0.39 -1.6 -0.029 -0.23 0.27 -1.9 0.87];
x2 = [-0.087 -3.3 -0.32 0.71 -5.3 0.89 1.9 -0.3 0.76 -1];
x3 = [0.58 -3.4 1.7 0.23 -0.15 -4.7 2.2 -0.87 -2.1 -2.6];
A = [x11 0 0; 0 x22 0; 0 0 x33];
mu = [mean(x1) mean(x2) mean(x3)];

summation = 0
for k = 1: 10
    summation = summation + A * 1/2 - ([x1(k) x2(k) x3(k)]-mu)' * ([x1(k) x2(k) x3(k)]-mu)/2;
end
x11_result = double(solve(summation(1,1)==0, x11))
x12_result = double(solve(summation(1,2)==0, x12))
x13_result = double(solve(summation(1,3)==0, x13))
x21_result = double(solve(summation(2,1)==0, x21))
x22_result = double(solve(summation(2,2)==0, x22))
x23_result = double(solve(summation(2,3)==0, x23))
x31_result = double(solve(summation(3,1)==0, x31))
x32_result = double(solve(summation(3,2)==0, x32))
x33_result = double(solve(summation(3,3)==0, x33))
```

```
x11_result =

   0.906177290000000


x12_result =

  0×1 빈 있는 double 열 벡터


x13_result =

  0×1 빈 있는 double 열 벡터


x21_result =

  0×1 빈 있는 double 열 벡터


x22_result =

   4.200714810000000


x23_result =

  0×1 빈 있는 double 열 벡터


x31_result =

  0×1 빈 있는 double 열 벡터


x32_result =

  0×1 빈 있는 double 열 벡터


x33_result =

   4.541949000000000
```

Figure 7: Problem D's matlab code and result

Figure 6는 Problem D의 코드와 결과이다. Python으론 upper(lower) triangle value를 조절하여 확인하기 어렵기 때문에 Figure 7과 같이 matlab을 통해 실제로도 같은 값을 가짐을 확인했다. Variance matrix의 diagonal elements만 존재한다는 의미는 각 data들을 서로 독립적으로 생각하겠다는 의미와 같다. 이는 다시 말하면 Problem B에서 diagonal elements에 대해서 언급했듯 variance가 해당 data가 아닌 다른 pair의 data가

전혀 관여하지 못함을 뜻한다. 즉, Problem A에서와 동일한 방법으로 구한 Figure 6의 결과에서 각각의 variance가 $\sigma_1^2, \sigma_2^2, \sigma_3^2$임을 확인할 수 있다. 또한 diagonal elements와 upper triangle이 모두 존재하는 Problem C에서도 diagonal elements가 eigenvalue와 같은 역할을 하기 때문에 upper(lower) triangle 값이 0이라 하더라도 동일한 값을 가짐을 확인할 수 있다.

Problem E and F

Compare your results for the mean of each feature μ_i calculated in the above ways. Explain why they are the same of different.
Compare your results for the variance of each feature σ_i^2 calculated in the above ways. Explain why they are the same or different.

```
Problem E and F
solve equation and comparison for w2 x1
mean = -0.1126
real mean = -0.11259999999999999
sigma = 0.0539258400000000
real sigma = 0.05392584


solve equation and comparison for w2 x2
mean = 0.4299
real mean = 0.42990000000000006
sigma = 0.0459700900000000
real sigma = 0.045970089999999936


solve equation and comparison for w2 x3
mean = 0.00372
real mean = 0.003720000000000001
sigma = 0.00726550560000000
real sigma = 0.0072655056


solve equation and comparison for w3 x1
mean = 0.2747
real mean = 0.27470000000000006
sigma = 0.301860810000000
real sigma = 0.30186081000000003


solve equation and comparison for w3 x2
mean = 0.3001
real mean = 0.30010000000000003
sigma = 0.644964090000000
real sigma = 0.64496409


solve equation and comparison for w3 x3
mean = 0.6786
real mean = 0.6786000000000001
sigma = 1.26214164000000
real sigma = 1.26214164
```

Figure 8: Problem E and F's code and result

1차원 data의 경우와 2차원 이상의 경우 모두 μ와 σ^2는 Problem A, B, C, D에서 확인함과 같이 실제 평균, 분산의 식으로 구한 값과 동일할 것이다. 실제로 Problem A에 대해서 살펴보면 Figure 1과 Figure 8의 결과를 보에서 mean과 sigma는 MLE를 이용하여 구한 값이며 real_mean과 real_sigma는 일반적으로 알려져있는 평균과 분산을 구한 것이다.

$$m = \frac{1}{n}\Sigma_{k=1}^{n} x_i, \sigma^2 = \frac{1}{n}[\Sigma_{k=1}^{n} x_k^2 - \mu^2] \tag{5}$$

이는 Figure 1의 코드에서 하단 부분에 real mean과 real sigma를 print한 부분을 보면 내장 함수를 이용하여 Equation 5와 동일하게 구했음을 알 수 있다. Gaussian distribution을 갖는 data를 maximum likelihood

estimation을 통해 구한 평균과 분산이 익히 알고있는 평균과 분산을 구한 값과 동일함을 알 수 있으며
지금까지 특정 sample에서의 평균과 분산을 구하는 것은 gaussian distribution을 가정하고 구한 값이라는 것을
알 수 있다.