Problem 1

(1)

For each datapoint y_i, we introduce a latent variable z_i, which indicates the normal distribution y_i is from, *i.e.* $z_i = 0 \iff y_i$ is from $N(\mu_1, \sigma_1^2)$ and vice versa. The EM algorithm is derived below.

In E-step, we first derive $p(z_n = k|x_n, \theta)$ using Bayesian formula where θ denotes μ_i, σ_i^2, w.

$$\begin{aligned} p(z_n = k|y_n, \theta) &= \frac{p(z_n = k, y_i|\theta)}{\sum_k p(z_n = k, y_i|\theta)} \\ &= \frac{w_k N(y_i|\mu_k, \sigma_k^2)}{\sum_k w_k N(y_i|\mu_k, \sigma_k^2)} \\ &= \gamma_{n,k} \end{aligned} \tag{1}$$

where $\gamma_{n,k}$ can be seen as a soft label of latent variable z_n. The log likelihood function is then

$$\begin{aligned} l(\theta) &= \sum_n \sum_k p(z_k|x_n, \theta) \log p(z_k, x_n|\theta) \\ &= \sum_n \sum_k \gamma_{n,k} (\log w_k - \frac{1}{2} 2\pi - \frac{1}{2} \log \sigma_k^2) - \frac{(y_n - \mu_k)^2}{2\sigma_k^2} \end{aligned} \tag{2}$$

In M-step, we take the derivative of $l(\theta)$, and set it to zero w.r.t. w_k, μ_k and σ_k.

a) max $l(\theta)$ w.r.t. w_k, using Lagrange multiplier since $\sum w_k = 1$

$$\begin{aligned} \sum_n \sum_k \gamma_{n,k} \frac{1}{w_k} - \lambda = 0, \quad \forall k \\ w_k \propto \sum_n \gamma_{n,k} \\ w_k = \frac{\sum_n \gamma_{n,k}}{\sum_k \sum_n \gamma_{n,k}} \end{aligned} \tag{3}$$

b) max $l(\theta)$ w.r.t. μ_k

$$\begin{aligned} \sum_n \gamma_{n,k} \frac{(y_n - \mu_k)}{\sigma_k^2} = 0 \\ \mu_k = \frac{\sum_n \gamma_{n,k} y_n}{\sum_n \gamma_{n,k}} \end{aligned} \tag{4}$$

c) max $l(\theta)$ w.r.t. σ_k^2

$$\begin{aligned} \sum_n \gamma_{n,k} (-\frac{1}{2\sigma_k^2} + \frac{(y_n - \mu_k)^2}{2(\sigma_k^2)^2}) = 0 \\ \sigma_k^2 = \frac{\sum_n \gamma_{n,k} (y_n - \mu_k)^2}{\sum_n \gamma_{n,k}} \end{aligned} \tag{5}$$

The EM algorithm is shown below

Algorithm 1: EM for GMM

Result: μ, σ^2, w
begin
  initialize μ, σ^2, w
  while *not converge* **do**
    calculate $\gamma_{n,k}$
    update μ, σ^2, w
  end
end

(2)
EM algorithm does not always converge on this dataset. In my implementation, four modes are found.

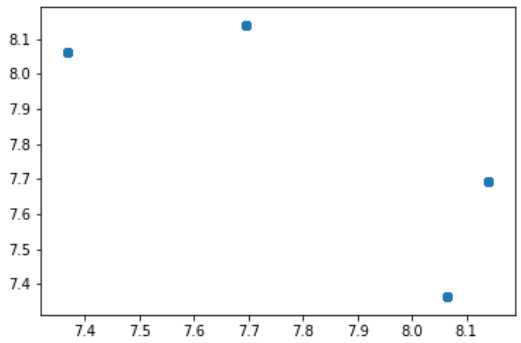


Figure 1: results of μ

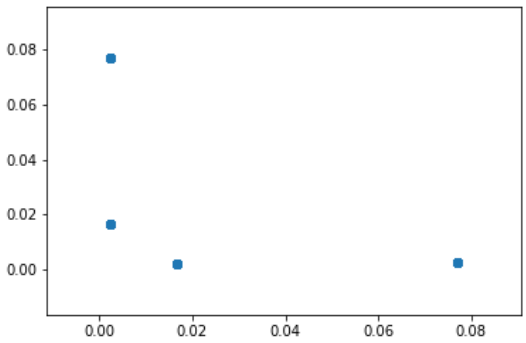


Figure 2: results of σ^2

Problem 2

(1)
The likelihood function for θ is

$$L(\theta) = \prod_n e^{-(x_n\theta + r_n)} \frac{(x_n\theta + r_n)^{y_n}}{y_n!} \tag{6}$$

(2)
Similar to the E-step in Problem1, we first calculate $E(z_{j1}|x_j, \theta)$. Notice that $z_{j1}|y_j \sim Binominal(y_j, \frac{x_j\theta}{x_j\theta + r_j})$. No need to calculate $E(z_{j2}|x_j, \theta)$ since it's irrelevant to θ in log likelihood function.

$$E(z_{j1}|x_j, \theta) = y_j \frac{x_j\theta}{x_j\theta + r_j} \tag{7}$$

In M-step, we derive its log likelihood function first

$$\begin{aligned} l(\theta) &= \sum_j -x_k + z_{j1} \log x_j\theta - \log z_{j1}! - r_j + z_{j2} \log r_j - \log z_{j2}! \\ \frac{dl(\theta)}{d\theta} &= \sum_j -x_j + z_{j1} \frac{1}{\theta} \end{aligned} \tag{8}$$

Substitute z_{j1} by $E(z_{j1}|x_j, \theta)$(since θ is irrelevant to z_{j1}) and set the derivative to zero, we have

$$\theta' = \frac{\theta}{\sum_j x_j} \sum_j \frac{y_j x_j}{x_j \theta + r_j} \tag{9}$$

which can be seen as a fix point iteration of true likelihood function.

Algorithm 2: EM for sum of Poission

```
Result: θ
begin
    initialize θ
    while not converge do
        calculate E(z_{j1}|x_j, θ)
        update θ
    end
end
```

(3)

The MLE is 5.606063396561341, which yields 1.78e-15 in true likelihood function.

(4)

By taking second derivative of log likelihood function in **(1)**, we have

$$\begin{aligned} -\frac{\partial^2 l(\theta)}{\partial \theta^2} &= \sum_n y_n \frac{x_n^2}{(x_n\theta + r_n)^2} \\ &= 2.423 \end{aligned} \tag{10}$$

, and the complete information being

$$\begin{aligned} -\frac{\partial^2 q(\theta)}{\partial \theta^2} &= \sum_n y_n \frac{x_n}{(x_n\theta + r_n)\theta} \\ &= 2.588 \end{aligned} \tag{11}$$

, so the fraction of missing information is 0.064

Problem 3

(1)

$$\begin{aligned} l(\pi) = &100 \log \pi_{11} + 50 \log \pi_{12} + 75 \log \pi_{21} + 75 \log \pi_{22} \\ &+ 28 \log(\pi_{11} + \pi_{12}) + 60 \log(\pi_{21} + \pi_{22}) + 30 \log(\pi_{11} + \pi_{21}) + 60 \log(\pi_{12} + \pi_{22}) \end{aligned} \tag{12}$$

The assumption under such a log likelihood includes

- outcomes of Y_1, Y_2 and that whether one of the outcomes is missing are independent, *i.e.* the fact that one of the outcomes is missing does not effect the prob. of that outcome

(2)

We derive the likelihood funtion for all data first

$$L(\pi) = \prod_n \pi_{11}^{y_{n1}=1, y_{n2}=1} \pi_{12}^{y_{n1}=1, y_{n2}=2} \pi_{21}^{y_{n1}=2, y_{n2}=1} \pi_{22}^{y_{n1}=2, y_{n2}=2} \tag{13}$$

E-step: For the observed 300 cases, all y_{n1}, y_{n2} are observed thus fixed, while for missing data, one of them need to be estimated, e.g. for the 28 datapoints with Y_1 missing, the estimation is simply from Bayesian formula

Problem 3 continued on next page...

$$p(y_{n1}=1,y_{n2}=1)=\frac{\pi_{11}}{\pi_{11}+\pi_{12}},\ p(y_{n1}=1,y_{n2}=2)=\frac{\pi_{12}}{\pi_{11}+\pi_{12}} \tag{14}$$

We do this for all missing data.
In M-step, we take the derivative of log likelihood function and set it to zero w.r.t. π_{ij}

$$\begin{aligned}\sum_n p(y_{n1}=i,y_{n2}=j)/\pi_{ij}-\lambda=0&\\ \pi_{ij}\propto\sum_n p(y_{n1}=i,y_{n2}=j)&\\ \pi_{ij}=\sum_n p(y_{n1}=i,y_{n2}=j)/\sum_{i,j}\sum_n p(y_{n1}=i,y_{n2}=j)&\end{aligned} \tag{15}$$

(3)
The MLE of π from EM algorithm is [0.27912927 0.17190756 0.24112718 0.30783599]
(4)
The odds ratio from complete data is 100 * 75 / (75 * 50) = 2.0, while that from MLE is 2.07, not equal.

```
import numpy as np
import scipy.stats
```

```
data = [8.1, 8.2, 8.1, 8.2, 8.2, 7.4, 7.3, 7.4, 8.1, 8.1, 7.9, 7.8, 8.2, 7.9, 7.9, 8.1,
8.1]
data = np.array(data)
```

```
def init():
    mu1 = np.random.random_sample() * 4 + 6.0
    mu2 = np.random.random_sample() * 4 + 6.0
    var1 = np.random.random_sample() * 2.0
    var2 = np.random.random_sample() * 2.0
    w = np.random.random_sample()
    return mu1, mu2, var1, var2, w
```

```
def get_prob(x, mu, var):
    return scipy.stats.norm(mu, np.sqrt(var)).pdf(x)
```

```
def e_step(mu1, mu2, var1, var2, w):
    soft_label = np.zeros(shape = data.shape)
    for i, x in enumerate(data):
        soft_label[i] = (w * get_prob(x, mu1, var1)) / (w * get_prob(x, mu1, var1) + (1
- w) * get_prob(x, mu2, var2))
    return soft_label
```

```
def m_step(mu1, mu2, var1, var2, w, soft_label):
    mu1 = np.sum(data * soft_label) / np.sum(soft_label)
    mu2 = np.sum(data * (1.0 - soft_label)) / np.sum(1.0 - soft_label)
    var1 = np.sum((data - mu1) * (data - mu1) * soft_label) / np.sum(soft_label)
    var2 = np.sum((data - mu2) * (data - mu2) * (1.0 - soft_label)) / np.sum(1.0 -
soft_label)
    w = np.sum(soft_label) / data.shape[0]
    return mu1, mu2, var1, var2, w
```

```
def EM():
    mu1, mu2, var1, var2, w = init()
    for i in range(100):
        soft_label = e_step(mu1, mu2, var1, var2, w)
        mu1, mu2, var1, var2, w = m_step(mu1, mu2, var1, var2, w, soft_label)
    return mu1, mu2, var1, var2, w
```

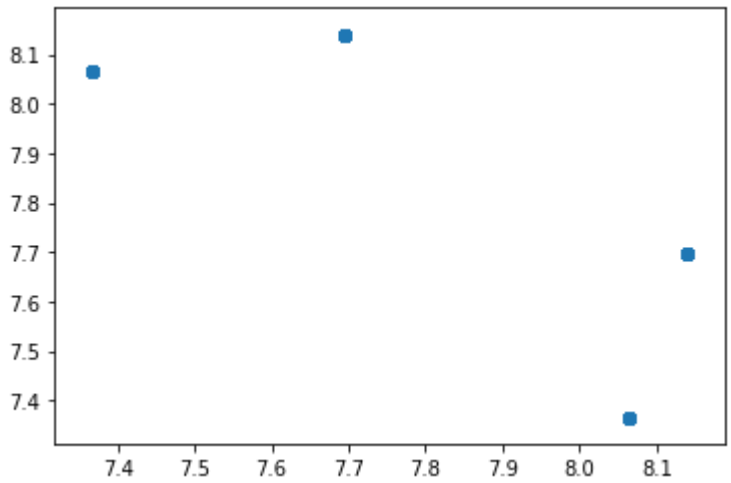
```
res_list = []
for i in range(100):
    res_list.append(EM())
```

```
res_conv_list = []
for res in res_list:
    if res[0] < 1e5:
        res_conv_list.append(res)
```

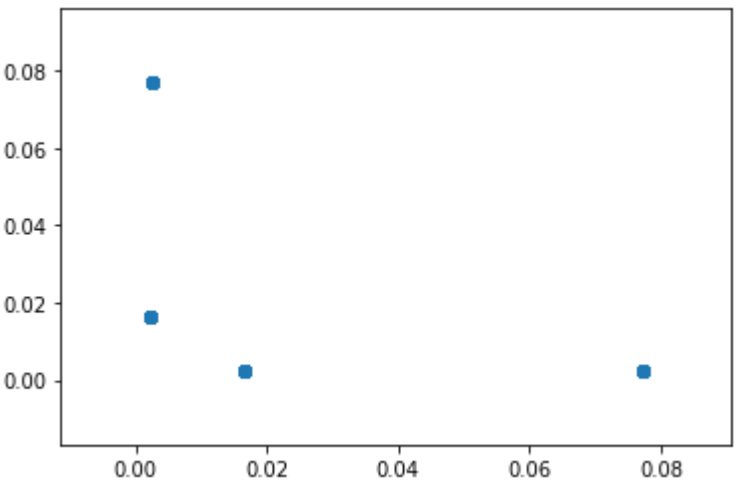
```
import matplotlib.pyplot as plt
```

```
res_conv_list = np.array(res_conv_list)
```

```
plt.scatter(res_conv_list[:, 0], res_conv_list[:, 1])
plt.savefig("mu_mode")
```




```
plt.scatter(res_conv_list[:, 2], res_conv_list[:, 3])
plt.savefig("sigma_mode")
```



```
import numpy as np
```

```
x = [1.41, 1.84, 1.64, 0.85, 1.32, 1.97, 1.70, 1.02, 1.84, 0.92]
```

```
r = [0.94, 0.70, 0.16, 0.38, 0.40, 0.57, 0.24, 0.27, 0.60, 0.81]
```

```
y = [13, 17, 6, 3, 7, 13, 8, 7, 5, 8]
```

```
def em_step(theta):
    sum_x = np.sum(x)
    tmp = 0.0
    for i in range(len(x)):
        tmp += y[i] * x[i] / (x[i] * theta + r[i])
    return theta * tmp / sum_x
```

```
def em_all():
    theta = np.random.random_sample() * 5
    for i in range(50):
        theta = em_step(theta)
    print(theta)
```

```
for i in range(100):
    em_all()
```

```
def get_l(theta):
    tmp = 0.0
    for i in range(len(x)):
        tmp += x[i] * y[i] / (x[i] * theta + r[i])
    sum_x = np.sum(x)
    return tmp - sum_x
```

```
get_l(5.606063396561341)
```

```
import numpy as np
```

```
def init(prob, pi):
    pi = np.random.random(4)
    pi /= np.sum(pi)
    for i in range(0, 100):
        prob[i][0] = 1.0
    for i in range(100, 150):
        prob[i][1] = 1.0
    for i in range(150, 225):
        prob[i][2] = 1.0
    for i in range(225, 300):
        prob[i][3] = 1.0
    return prob, pi
```

```
def e_step(prob, pi):
    for i in range(300, 300 + 28):
        prob[i][0] = pi[0] / (pi[0] + pi[1])
        prob[i][1] = pi[1] / (pi[0] + pi[1])
    for i in range(300 + 28, 300 + 28 + 60):
        prob[i][2] = pi[2] / (pi[2] + pi[3])
        prob[i][3] = pi[3] / (pi[2] + pi[3])
    for i in range(300 + 28 + 60, 300 + 28 + 60 + 30):
        prob[i][0] = pi[0] / (pi[0] + pi[2])
        prob[i][2] = pi[2] / (pi[0] + pi[2])
    for i in range(300 + 28 + 60 + 30, 300 + 28 + 60 + 30 + 60):
        prob[i][1] = pi[1] / (pi[1] + pi[3])
        prob[i][3] = pi[3] / (pi[1] + pi[3])
    return prob
```

```
def m_step(prob):
    pi_new = np.sum(prob, axis=0)
    return pi_new / np.sum(pi_new)
```

```
def em_step(prob, pi):
    prob = e_step(prob, pi)
    pi = m_step(prob)
    return prob, pi
```

```
prob = np.zeros(shape=[478, 4])
pi = np.zeros(shape=4)
prob, pi = init(prob, pi)
prob, pi = em_step(prob, pi)
```