Q4

May 19, 2019

```
In [1]: import numpy as np
        import pandas as pd
        from scipy.linalg import sqrtm
```

0.1 Calculate the pmf and Derivatives

FIrst we can make the pmf of y

$$p_y(y) = \pi \cdot I(y=0) + (1-\pi)\frac{e^{-\lambda}\lambda^y}{y!}$$

Let $\theta = (\lambda, \pi)$ and $Y = (y_1, \ldots, y_n)$ Then likelihood is,

$$\begin{aligned}
L(\theta|Y) &= \prod_{i=1}^{n}\left[\pi \cdot I(y_i=0) + (1-\pi)\frac{e^{-\lambda}\lambda^{y_i}}{y_i!}\right] \\
&= \prod_{y_i=0}\left[\pi + (1-\pi)\frac{e^{-\lambda}\lambda^{0}}{0!}\right]\prod_{y_i\neq 0}\left[(1-\pi)\frac{e^{-\lambda}\lambda^{y_i}}{y_i!}\right]
\end{aligned}$$

$N = \sum_{k=0}^{6} n_k$, Then log likelihood $l(\theta)$ is

$$\begin{aligned}
l(\theta) &= \sum_{y_i=0} log\left[\pi + (1-\pi)\frac{e^{-\lambda}\lambda^{0}}{0!}\right] + \sum_{y_i\neq 0} log\left[(1-\pi)\frac{e^{-\lambda}\lambda^{y_i}}{y_i!}\right] \\
&= n_0 log\left[\pi + (1-\pi)e^{-\lambda}\right] + (N-n_0)\left[log(1-\pi) - \lambda\right] + \sum_{y_i\neq 0}\left[y_i log\lambda - log(y_i!)\right]
\end{aligned}$$

Then, log likelihood function is

```
In [2]: def loglikelihood(lam,pi,Y):
            n0 = sum(Y==0)
            N = len(Y)
            out = n0 * np.log(pi + (1-pi)*np.exp(-lam)) \
                  + (N-n0)*(np.log(1-pi) - lam) \
                  +(Y*np.log(lam)\
                  - np.array(list(map(np.math.factorial,Y)))).sum()
            return(out)
```

First derivative is

$$
\begin{aligned}
\frac{\partial l(\theta)}{\partial \lambda} &= (-n_0)\cdot\frac{(1-\pi)e^{-\lambda}}{\pi+(1-\pi)e^{-\lambda}} - (N-n_0) + \sum_{y_i\neq 0}\left(\frac{y_i}{\lambda}\right)\\
&= (-n_0)\cdot\frac{(1-\pi)e^{-\lambda}}{\pi+(1-\pi)e^{-\lambda}} - (N-n_0) + \sum_{y_i}\left(\frac{y_i}{\lambda}\right)\\
\frac{\partial l(\theta)}{\partial \pi} &= n_0\cdot\frac{1-e^{-\lambda}}{\pi+(1-\pi)e^{-\lambda}} - (N-n_o)\cdot\frac{1}{1-\pi}
\end{aligned}
$$

Second Derivatice is

$$
\begin{aligned}
\frac{\partial^2 l\left(\theta\right)}{\partial \lambda^2} &= n_0\cdot\frac{\pi\left(1-\pi\right)e^{-\lambda}}{\left(\pi+\left(1-\pi\right)e^{-\lambda}\right)^2} - \sum_{y_i\neq 0}\frac{y_i}{\lambda^2}\\
&= n_0\cdot\frac{\pi\left(1-\pi\right)e^{-\lambda}}{\left(\pi+\left(1-\pi\right)e^{-\lambda}\right)^2} - \sum_{y_i}\frac{y_i}{\lambda^2}\\
\frac{\partial^2 l\left(\theta\right)}{\partial \pi^2} &= (-n_0)\cdot\frac{\left(1-e^{-\lambda}\right)^2}{\left(\pi+\left(1-\pi\right)e^{-\lambda}\right)^2} - (N-n_0)\,\frac{1}{\left(1-\pi\right)^2}\\
\frac{\partial^2 l\left(\theta\right)}{\partial \pi\partial \lambda} &= n_0\cdot\frac{e^{-\lambda}}{\left(\pi+\left(1-\pi\right)e^{-\lambda}\right)^2}
\end{aligned}
$$

1 (a) Derive Newton's Method and Fisher Scoring Method

1.1 Newton's method

Iterate

$$\theta^{(t+1)} = \theta^{(t)} - \left[l''(\theta^{(t)})\right]^{-1} l'(\theta^{(t)})$$

and Standard Error Estimate is

$$\sqrt{\left(-l''(\theta)\right)^{-1}}$$

We calculated $l'(\theta)$ and $l''(\theta)$, first make the dataset and derivative functions

```
In [3]: Y = np.concatenate([np.repeat(0,3062),np.repeat(1,587),np.repeat(2,284),
                            np.repeat(3,103),np.repeat(4,33),np.repeat(5,4),
                            np.repeat(6,2)])
        Y.shape
```

```
Out[3]: (4075,)
```

```
In [4]: def first_der_lam(lam,pi,y):
            n0 = sum(y==0)
            N = len(y)
            out = -n0 *(((1-pi)*np.exp(-lam))/(pi+(1-pi)*np.exp(-lam))) \
                  -(N-n0) + y.sum()/lam
            return(out)
```

```
In [5]: def first_der_pi(lam,pi,y):
            n0 = sum(y==0)
            N = len(y)
            out = n0*((1-np.exp(-lam))/(pi+(1-pi)*np.exp(-lam))) \
                  -(N-n0)/(1-pi)
            return(out)
```

```
In [6]: def second_der_lam2(lam,pi,y):
            n0 = sum(y==0)
            N = len(y)
            out = n0*((pi*(1-pi)*np.exp(-lam))/((pi + (1-pi)*np.exp(-lam))**2))\
                  -y.sum()/(lam**2)
            return(out)
```

```
In [7]: def second_der_pi2(lam,pi,y):
            n0 = sum(y==0)
            N = len(y)
            out = -n0*(((1-np.exp(-lam))**2)/((pi-(1-pi)*np.exp(-lam))**2)) \
                  -(N-n0)/((1-pi)**2)
            return(out)
```

```
In [8]: def second_der_pilam(lam,pi,y):
            n0 = sum(y==0)
            N = len(y)
            out = n0*((np.exp(-lam))/((pi+(1-pi)*np.exp(-lam))**2))
            return(out)
```

Set the initail value

$$\lambda_0 = 1$$
$$\pi_0 = 0.5$$

```
In [9]: lam , pi = 1, 0.5
        theta = np.array([lam,pi])
```

Make the function iterate until loglikelihood do not increase more than criteria

```
In [10]: def Newton(theta,Y,citeria = 10**(-7)):
             llikelst = [loglikelihood(theta[0],theta[1],Y)]
             thetalst = [theta]
             niter = 0
             while True:
                 niter = niter + 1
                 l1 = np.array([first_der_lam(theta[0],theta[1],Y)
                               ,first_der_pi(theta[0],theta[1],Y)])
                 l2 = np.reshape([second_der_lam2(theta[0],theta[1],Y),
                             second_der_pilam(theta[0],theta[1],Y),
                             second_der_pilam(theta[0],theta[1],Y),
                             second_der_pi2(theta[0],theta[1],Y)],(2,2))
                 theta = theta - np.linalg.inv(l2).dot(l1)
```

```
            thetalst.append(theta)
            llikelst.append(loglikelihood(theta[0],theta[1],Y))
            if (abs(llikelst[-1]-llikelst[-2]) < citeria):
                break
        out = pd.DataFrame({'lambda' : pd.DataFrame(thetalst)[0],
                            'pi': pd.DataFrame(thetalst)[1],
                            'logLikelihood':llikelst})
        stdm = sqrtm(-np.linalg.inv(l2))
        return(out,stdm)
```

In [11]:
```
N_result,N_stdm = Newton(theta,Y)
```

Then the result is

$$\hat{\theta}^{Newton} = (\hat{\lambda}^{Newton}, \hat{\pi}^{Newton}) = (1.037836, 0.615055)$$

it converges at 13 times

In [12]:
```
N_result
```

Out[12]:

	lambda	pi	logLikelihood
0	1.000000	0.500000	-10425.367474
1	0.866135	0.529549	-10396.762382
2	0.939669	0.562936	-10386.699016
3	0.990509	0.589546	-10381.765834
4	1.018312	0.604416	-10380.371074
5	1.030320	0.610933	-10380.109878
6	1.035009	0.613502	-10380.069499
7	1.036783	0.614476	-10380.063694
8	1.037446	0.614841	-10380.062881
9	1.037693	0.614976	-10380.062768
10	1.037785	0.615027	-10380.062752
11	1.037819	0.615046	-10380.062750
12	1.037832	0.615053	-10380.062750
13	1.037836	0.615055	-10380.062750

Standard error estimate is

In [13]:
```
pd.DataFrame(N_stdm)
```

Out[13]:

	0	1
0	0.036015	0.004470
1	0.004470	0.009601

1.2 Fisher Scoring method

Iterate

$$\theta^{(t+1)} = \theta^{(t)} + \left[I(\theta^{(t)})\right]^{-1} l'(\theta^{(t)})$$

where $I(\theta) = E\left[-l''(\theta)\right]$, and Standard Error Estimate is

$$\sqrt{\left[I(\theta^{(t)})\right]^{-1}}$$

We calculated $l'(\theta)$, $l''(\theta)$, we only need to calculate $I(\theta) = E\left[-l''(\theta)\right]$

Since

$$n_0, \ldots, n_6 \sim Multinomial\left(N, \left\{\pi + (1-\pi)e^{-\lambda}, (1-\pi)\frac{\lambda^1 e^{-\lambda}}{1!}, \ldots, \frac{\lambda^6 e^{-\lambda}}{6!}\right\}\right)$$

Expected value of n_k is

$$E[n_0] = N \cdot (\pi + (1-\pi)e^{-\lambda})$$
$$E[n_k] = N \cdot \left((1-\pi)\frac{\lambda^k e^{-\lambda}}{k!}\right) \text{ for i} = 1, \ldots .6$$

Then $I(\theta) = E\left[-l''(\theta)\right]$ is

$$\begin{aligned}
E\left[-\frac{\partial^2 l(\theta)}{\partial \lambda^2}\right] &= -E[n_0] \cdot \frac{\pi(1-\pi)e^{-\lambda}}{\left(\pi + (1-\pi)e^{-\lambda}\right)^2} + E\left[\sum_{y_i \neq 0} \frac{y_i}{\lambda^2}\right] \\
&= -E[n_0] \cdot \frac{\pi(1-\pi)e^{-\lambda}}{\left(\pi + (1-\pi)e^{-\lambda}\right)^2} + E\left[\sum_{k=1}^{6} \frac{k \cdot n_k}{\lambda^2}\right] \\
&= -E[n_0] \cdot \frac{\pi(1-\pi)e^{-\lambda}}{\left(\pi + (1-\pi)e^{-\lambda}\right)^2} + \sum_{k=1}^{6} \frac{k \cdot E[n_k]}{\lambda^2} \\
E\left[-\frac{\partial^2 l(\theta)}{\partial \pi^2}\right] &= E[n_0] \cdot \frac{\left(1-e^{-\lambda}\right)^2}{\left(\pi + (1-\pi)e^{-\lambda}\right)^2} - (N - E[n_0])\frac{1}{(1-\pi)^2} \\
E\left[-\frac{\partial^2 l(\theta)}{\partial \pi \partial \lambda}\right] &= -E[n_0] \cdot \frac{e^{-\lambda}}{\left(\pi + (1-\pi)e^{-\lambda}\right)^2}
\end{aligned}$$

Make the function of expectation of n_k's where $k \neq 0$

```
In [14]: def E_nk(N,k,lam,pi):
             return (N*(1-pi)*(lam**k)*np.exp(-lam))/(np.math.factorial(k))
```

```
In [15]: def E_second_der_lam2(lam,pi,y):
             N = len(y)
             n0 = N*(pi + (1-pi)*np.exp(-lam))
             summ = 0
             for k in range(7):
                 summ  = summ + k*E_nk(N,k,lam,pi)
             out = n0*((pi*(1-pi)*np.exp(-lam))/((pi + (1-pi)*np.exp(-lam))**2))\
                   -(summ)/(lam**2)
             return(out)
```

```
In [16]: def E_second_der_pi2(lam,pi,y):
             N = len(y)
             n0 = N*(pi + (1-pi)*np.exp(-lam))
             out = -n0*(((1-np.exp(-lam))**2)/((pi-(1-pi)*np.exp(-lam))**2)) \
                   -(N-n0)/((1-pi)**2)
             return(out)
```

In [17]:
```
def E_second_der_pilam(lam,pi,y):
    N = len(y)
    n0 = N*(pi + (1-pi)*np.exp(-lam))
    out = n0*((np.exp(-lam))/((pi+(1-pi)*np.exp(-lam))**2))
    return(out)
```

Set the initail value

$$\lambda_0 = 1$$
$$\pi_0 = 0.5$$

In [18]:
```
lam , pi = 1, 0.5
theta = np.array([lam,pi])
```

In [19]:
```
def Fisher(theta,Y,citeria = 10**(-7)):
    llikelst = [loglikelihood(theta[0],theta[1],Y)]
    thetalst = [theta]
    niter = 0
    while True:
        niter = niter + 1
        l1 = np.array([first_der_lam(theta[0],theta[1],Y)
                      ,first_der_pi(theta[0],theta[1],Y)])
        l2 = np.reshape([E_second_der_lam2(theta[0],theta[1],Y),
                    E_second_der_pilam(theta[0],theta[1],Y),
                    E_second_der_pilam(theta[0],theta[1],Y),
                    E_second_der_pi2(theta[0],theta[1],Y)],(2,2))
        theta = theta - np.linalg.inv(l2).dot(l1)
        thetalst.append(theta)
        llikelst.append(loglikelihood(theta[0],theta[1],Y))
        if (abs(llikelst[-1]-llikelst[-2]) < citeria):
            break
    out = pd.DataFrame({'lambda' : pd.DataFrame(thetalst)[0],
                      'pi': pd.DataFrame(thetalst)[1],
                      'logLikelihood':llikelst})
    stdm = sqrtm(-np.linalg.inv(l2))
    return(out,stdm)
```

In [20]:
```
F_result,F_stdm = Fisher(theta,Y)
```

Then the result is

$$\hat{\theta}^{Fisher} = (\hat{\lambda}^{Fisher}, \hat{\pi}^{Fisher}) = (1.037836, 0.615055)$$

it converges at 13 times

In [21]:
```
F_result
```

Out[21]:

	lambda	pi	logLikelihood
0	1.000000	0.500000	-10425.367474
1	0.915897	0.538016	-10393.735425

```
2    0.950009   0.568840   -10385.365635
3    0.992221   0.591847   -10381.488303
4    1.018426   0.605006   -10380.339002
5    1.030359   0.611067   -10380.106903
6    1.035041   0.613536   -10380.069205
7    1.036799   0.614487   -10380.063659
8    1.037453   0.614844   -10380.062876
9    1.037696   0.614978   -10380.062767
10   1.037786   0.615027   -10380.062752
11   1.037819   0.615046   -10380.062750
12   1.037832   0.615053   -10380.062750
13   1.037836   0.615055   -10380.062750
```

Standard error estimates is

```
In [22]: pd.DataFrame(F_stdm)
```

```
Out[22]:          0         1
         0  0.036041  0.004474
         1  0.004474  0.009601
```

2 (b) EM algorithn

Let Z be random varible

$$Z \sim Bernoulli(\pi)$$

and $\theta = (\lambda, \pi)$ Then,

$$\begin{aligned} L(\theta|Y,Z) &= P(Y,Z|\theta) \\ &= P(Y|Z,\theta)P(Z|\theta) \\ &= \prod_i I(y_i=0)^{z_i} \left(\frac{e^{-\lambda}\lambda^{y_i}}{y_i!}\right)^{1-z_i} \prod_i \pi^{z_i}(1-\pi)^{1-z_i} \\ &= \prod_i [\pi \cdot I(y_i=0)]^{z_i} \left[(1-\pi)\left(\frac{e^{-\lambda}\lambda^{y_i}}{y_i!}\right)\right]^{1-z_i} \end{aligned}$$

log likelihood is

$$l(\theta|Y,Z) = \sum_i [z_i log(\pi \cdot I(y_i=0)) + (1-z_i)\{log(1-\pi) - \lambda + y_i log(\lambda) - log(y_i!)\}]$$

2.0.1 E-step

$$\begin{aligned} Q(\theta|\theta^{(t)}) &= E[l(\theta|Y_{com})|Y_{obs},\theta^{(t)}] \\ &= \sum_i \left[E[z_i|Y,\theta^{(t)}] log(\pi \cdot I(y_i=0)) + (1-E[z_i|Y,\theta^{(t)}])\{log(1-\pi) - \lambda + y_i log(\lambda) - log(y_i!)\}\right] \end{aligned}$$

Where

$$
\begin{aligned}
E[z_i|Y,\theta^{(t)}] &= P(z_i = 1|Y,\theta^{(t)}) \\
&= \frac{P(y_i|z_i = 1,\theta^{(t)})P(z_i = 1)}{P(y_i|z_i = 0,\theta^{(t)})P(z_i = 0) + P(y_i|z_i = 1,\theta^{(t)})P(z_i = 1)} \\
&= \begin{cases} 0 & \text{when } y_i \neq 0 \\ \frac{\pi^{(t)}}{\pi^{(t)} + (1-\pi^{(t)})e^{-\lambda^{(t)}}} & \text{when } y_i = 0 \end{cases}
\end{aligned}
$$

2.0.2 M-step

First partial derivative for λ,

$$
\begin{aligned}
&\frac{\partial Q(\theta|\theta^{(t)})}{\partial \lambda} =^{let} 0 \\
\rightarrow &\sum_i (1 - E[z_i|Y,\theta^{(t)}])(-1 + \frac{y_i}{\lambda}) = 0 \\
\rightarrow &\lambda^{(t+1)} = \frac{\sum_i (1 - E[z_i|Y,\theta^{(t)}]) \cdot y_i}{\sum_i (1 - E[z_i|Y,\theta^{(t)}])}
\end{aligned}
$$

Second partial derivative for π,

$$
\begin{aligned}
&\frac{\partial Q(\theta|\theta^{(t)})}{\partial \pi} =^{let} 0 \\
\rightarrow &\sum_i \frac{E[z_i|Y,\theta^{(t)}]}{\pi} - \sum_i \frac{1 - E[z_i|Y,\theta^{(t)}]}{1-\pi} = 0 \\
\rightarrow &\pi^{(t+1)} = \frac{1}{N} \sum_i E[z_i|Y,\theta^{(t)}]
\end{aligned}
$$

```
In [23]: def Estep(lam,pi,Y):
             if Y ==0:
                 out = pi/(pi + (1-pi)*np.exp(-lam))
             else:
                 out = 0
             return(out)
```

```
In [24]: def Mstep(lam,pi,Y):
             lam = sum(list(map(lambda y : (1-Estep(lam,pi,y))*y,Y)))\
                   /sum(list(map(lambda y : (1-Estep(lam,pi,y)),Y)))
             pi = sum(list(map(lambda y : Estep(lam,pi,y),Y)))/len(Y)
             return(lam,pi)
```

```
In [25]: def iterateEM(theta,Y,citeria = 10**(-7)):
             llikelst = [loglikelihood(theta[0],theta[1],Y)]
             thetalst = [theta]
             while True:
                 theta = Mstep(theta[0],theta[1],Y)
                 thetalst.append(theta)
```

```
        llikelst.append(loglikelihood(theta[0],theta[1],Y))
        if (abs(llikelst[-1]-llikelst[-2]) < citeria):
            break
    out = pd.DataFrame({'lambda' : pd.DataFrame(thetalst)[0],
                        'pi': pd.DataFrame(thetalst)[1],
                        'logLikelihood':llikelst})
    return(out)
```

Set the initail value

$$\lambda_0 = 1$$
$$\pi_0 = 0.5$$

In [26]:
```
lam , pi = 1, 0.5
theta = np.array([lam,pi])
```

In [27]:
```
EM_result = iterateEM(theta,Y)
```

In [28]:
```
EM_result
```

Out[28]:

	lambda	pi	logLikelihood
0	1.000000	0.500000	-10425.367474
1	0.886469	0.532120	-10395.553509
2	0.890872	0.552198	-10390.295073
3	0.915914	0.567316	-10386.553374
4	0.941596	0.579144	-10383.965771
5	0.963787	0.588331	-10382.322126
6	0.981853	0.595353	-10381.333153
7	0.996074	0.600637	-10380.760983
8	1.007004	0.604565	-10380.439774
9	1.015254	0.607456	-10380.263591
10	1.021396	0.609568	-10380.168651
11	1.025921	0.611102	-10380.118167
12	1.029230	0.612212	-10380.091588
13	1.031635	0.613013	-10380.077695
14	1.033376	0.613590	-10380.070472
15	1.034633	0.614005	-10380.066731
16	1.035538	0.614303	-10380.064799
17	1.036188	0.614516	-10380.063804
18	1.036656	0.614669	-10380.063291
19	1.036991	0.614779	-10380.063028
20	1.037231	0.614858	-10380.062892
21	1.037404	0.614914	-10380.062823
22	1.037527	0.614955	-10380.062787
23	1.037616	0.614984	-10380.062769
24	1.037679	0.615004	-10380.062760
25	1.037724	0.615019	-10380.062755
26	1.037757	0.615030	-10380.062752
27	1.037780	0.615038	-10380.062751
28	1.037797	0.615043	-10380.062750

```
29  1.037809  0.615047  -10380.062750
30  1.037818  0.615050  -10380.062750
31  1.037824  0.615052  -10380.062750
```

Then the result is

$$\hat{\theta}^{EM} = (\hat{\lambda}^{EM}, \hat{\pi}^{EM}) = (1.037824, 0.615052)$$

it converges at 31 times

3 (c) compare the result

Newton's method result is

$$\hat{\theta}^{Fisher} = (\hat{\lambda}^{Fisher}, \hat{\pi}^{Fisher}) = (1.037836, 0.615055)$$

it converges at 13 times

Fisher scoring method result is

$$\hat{\theta}^{Fisher} = (\hat{\lambda}^{Fisher}, \hat{\pi}^{Fisher}) = (1.037836, 0.615055)$$

it converges at 13 times

EM result is

$$\hat{\theta}^{EM} = (\hat{\lambda}^{EM}, \hat{\pi}^{EM}) = (1.037824, 0.615052)$$

it converges at 31 times

Estimated value of θ is very similar but EM algorithm converges slower than Newton's and Fisher scoring method

Q7

(a) Derive Newton's Method

$$
\begin{aligned}
L(\theta|Y) &= \prod_{i=1}^{n} P(y_i|\theta) \\
&= \prod_{i=1}^{n} \frac{(\theta+1)^{y_i} e^{-(\theta+1)}}{y_i!} \\
l(\theta|Y) &= log(\theta+1)\sum_{i=1}^{n} y_i - n(\theta+1) - \sum_{i=1}^{n} log(y_i!)
\end{aligned}
$$

First and Second derivative is

$$
\begin{aligned}
\frac{\partial l(\theta|Y)}{\partial \theta} &= \frac{\sum_{i=1}^{n} y_i}{\theta+1} - n \\
\frac{\partial^2 l(\theta|Y)}{\partial \theta^2} &= -\frac{\sum_{i=1}^{n} y_i}{(\theta+1)^2}
\end{aligned}
$$

So we can optimize θ by iteratation

$$
\begin{aligned}
\theta^{(t+1)} &= \theta^{(t)} - \left[\frac{\partial^2 l(\theta^{(t)}|Y)}{\partial {\theta^{(t)}}^2}\right]^{-1} \frac{\partial l(\theta^{(t)}|Y)}{\partial \theta^{(t)}} \\
&= 1 + 2\theta^{(t)} - \frac{n(\theta+1)^2}{\sum_{i=1}^{n} y_i}
\end{aligned}
$$

(b) Scoring method

Since $y_i \sim Poisson(\theta+1)$, $E[y_i] = \theta + 1$. Then,

$$
\begin{aligned}
I(\theta) = E\left[-\frac{\partial^2 l(\theta|Y)}{\partial \theta^2}\right] &= \frac{\sum_{i=1}^{n} E[y_i]}{(\theta+1)^2} \\
&= \frac{n}{\theta+1}
\end{aligned}
$$

So we can optimize θ by iteratation

$$
\begin{aligned}
\theta^{(t+1)} &= \theta^{(t)} - \left[I(\theta^{(t)})\right]^{-1} \frac{\partial l(\theta^{(t)}|Y)}{\partial \theta^{(t)}} \\
&= \frac{\sum_{i=1}^{n} y_i}{n} - 1
\end{aligned}
$$

(c) Derive EM

Treat s_i's as missing data

$$
\begin{aligned}
L(\theta|S,Y) &= \prod_{i=1}^{n} P(S,Y|\theta) \\
&= \prod_{i=1}^{n} P(Y|S,\theta)P(S|\theta) \\
&= \prod_{i=1}^{n} \frac{e^{-1}}{(y_i - s_i)!} \frac{\theta^{s_i} e^{-\theta}}{s_i!} \\
&= e^{-n(1+\theta)} \cdot \theta^{\sum_{i=1}^{n} s_i} \cdot \prod_{i=1}^{n} \frac{1}{(y_i - s_i)! \cdot s_i!}
\end{aligned}
$$

Then Log likelihood is

$$
l(\theta|S,Y) = -n(1+\theta) + \sum_{i=1}^{n} s_i log(\theta)
$$

0.1 E-step

$$
\begin{aligned}
Q(\theta|\theta^{(t)}) &= E[l(\theta|Y_{com})|Y_{obs}, \theta^{(t)}] \\
&= -n(1+\theta) + \sum_{i=1}^{n} E[s_i|Y_{obs}, \theta^{(t)}] log(\theta)
\end{aligned}
$$

PMF of $s_i|Y_{obs}, \theta^{(t)}$ is

$$
\begin{aligned}
P(s_i|Y_{obs} = y_i, \theta^{(t)}) &= \frac{P(s_i, y_i|\theta^{(t)})}{P(y_i|\theta^{(t)})} \\
&= \frac{{\theta^{(t)}}^{s_i} e^{-\theta^{(t)}}}{s_i!} \cdot \frac{e^{-1}}{(y_i - s_i)!} \cdot \left[\frac{(\theta^{(t)}+1)^{y_i} e^{-(\theta^{(t)}+1)}}{y_i!} \right]^{-1} \\
&= \frac{y_i!}{(y_i - s_i)! s_i!} \left(\frac{\theta^{(t)}}{\theta^{(t)}+1} \right)^{s_i} \left(\frac{1}{\theta^{(t)}+1} \right)^{y_i - s_i}
\end{aligned}
$$

Thus $s_i|Y_{obs} = y_i, \theta^{(t)} \sim Binomial\left(y_i, \frac{\theta^{(t)}}{\theta^{(t)}+1}\right)$, Then $E[s_i|Y_{obs}, \theta^{(t)}] = y_i \cdot \frac{\theta^{(t)}}{\theta^{(t)}+1}$

0.2 M-step

$$
Q(\theta|\theta^{(t)}) = -n(1+\theta) + log\theta \cdot \sum_{i=1}^{n} y_i \cdot \frac{\theta^{(t)}}{\theta^{(t)}+1}
$$

Maximize the Qfunction by letting the first derivative 0

$$\begin{aligned}\frac{\partial Q(\theta|\theta^{(t)})}{\partial\theta} &=^{let} 0 \\ &= -n + \frac{\sum_{i=1}^{n} y_i \cdot \frac{\theta^{(t)}}{\theta^{(t)}+1}}{\theta} \\ \theta^{(t+1)} &= \frac{\sum_{i=1}^{n} y_i}{n} \cdot \frac{\theta^{(t)}}{\theta^{(t)}+1}\end{aligned}$$

We can get the $\hat{\theta}$ by iterate the E-step and M-step

(d) EM algorithm has unique solution

EM algorithm converges when $\theta^{(t+1)} = \theta^{(t)}$

$$\begin{aligned}\hat{\theta} &= \frac{\sum_{i=1}^{n} y_i}{n} \cdot \frac{\hat{\theta}}{\hat{\theta}+1} \\ \hat{\theta} &= \frac{\sum_{i=1}^{n} y_i}{n} - 1\end{aligned}$$

which does not depend on $\theta^{(t)}$. it means that regardless inital value EM-algorithm converge to unique solution

(e) find the convergence rate of the EM

as $M(\theta) = \frac{\sum_{i=1}^{n} y_i}{n} \cdot \frac{\theta}{\theta+1}$

$$\begin{aligned}DM(\theta) &= \frac{\partial M(\theta)}{\partial\theta} \\ &= \frac{\sum_{i=1}^{n} y_i}{n \cdot (\theta+1)^2}\end{aligned}$$

(f) Compute $V_{obs}(\hat{\theta})$ using Louis' method

from (c) $s_i|Y_{obs} = y_i, \theta^{(t)} \sim Binomial\left(y_i, \frac{\theta^{(t)}}{\theta^{(t)}+1}\right)$. Then,

$$\begin{aligned}E[s_i|Y_{obs}, \theta] &= y_i \cdot \frac{\theta}{\theta+1} \\ V[s_i|Y_{obs}, \theta] &= y_i \cdot \frac{\theta}{(\theta+1)^2}\end{aligned}$$

First compute the $I_{com} = E[-l''(\theta|Y_{com})|Y_{obs}, \theta]$

$$
\begin{aligned}
E[-l''(\theta|Y_{com})|Y_{obs}, \theta] &= \frac{\partial^2(n(1+\theta) - log\theta \sum_{i=1}^{n} E[s_i|Y_{obs}, \theta]))}{\partial\theta^2} \\
&= \frac{1}{\theta^2}\sum_{i=1}^{n} y_i \cdot \frac{\theta}{\theta+1} \\
&= \frac{1}{\theta(\theta+1)}\sum_{i=1}^{n} y_i
\end{aligned}
$$

Second compute $Var(l'(\theta|Y_{com})|Y_{obs}, \theta)$

$$
\begin{aligned}
Var(l'(\theta|Y_{com})|Y_{obs}, \theta) &= Var\left(-n + \frac{1}{\theta}\sum_{i=1}^{n} s_i|Y_{obs}, \theta\right) \\
&= \frac{1}{\theta^2}\sum_{i=1}^{n} Var(s_i|Y_{obs}, \theta) \\
&= \frac{1}{\theta^2}\sum_{i=1}^{n} y_i \cdot \frac{\theta}{(\theta+1)^2} \\
&= \frac{1}{\theta(\theta+1)}\sum_{i=1}^{n} y_i
\end{aligned}
$$

as $I_{obs} = I_{com} - Var(l'(\theta|Y_{com})|Y_{obs}, \theta)$

$$
\begin{aligned}
I_{obs} = I_{com} - Var(l'(\theta|Y_{com})|Y_{obs}, \theta) &= \frac{1}{\theta(\theta+1)}\sum_{i=1}^{n} y_i - \frac{1}{\theta(\theta+1)}\sum_{i=1}^{n} y_i \\
&= \frac{1}{(\theta+1)^2}\sum_{i=1}^{n} y_i
\end{aligned}
$$

Thus,

$$
V_{obs} = I_{obs}^{-1} = \frac{(\theta+1)^2}{\sum_{i=1}^{n} y_i}
$$

Then,

$$
V_{obs}(\hat{\theta}) = \frac{(\hat{\theta}+1)^2}{\sum_{i=1}^{n} y_i}
$$

where $\hat{\theta} = \frac{\sum_{i=1}^{n} y_i}{n} - 1$

$$
\begin{aligned}
V_{obs}(\hat{\theta}) &= \frac{(\hat{\theta}+1)^2}{\sum_{i=1}^{n} y_i} \\
&= \frac{1}{n^2}\sum_{i=1}^{n} y_i
\end{aligned}
$$

(f) Compute $V_{obs}(\hat{\theta})$ using SEM

$V_{obs}(\hat{\theta}) = \Big[(1 - DM(\hat{\theta}))I_{com}\Big]^{-1}$ from (e) and (f) we already computed $DM(\hat{\theta})$ and I_{com}

$$
\begin{aligned}
V_{obs}(\hat{\theta}) &= \Big[(1 - DM(\hat{\theta}))I_{com}\Big]^{-1} \\
&= \left[\left(1 - \frac{\sum_{i=1}^n y_i}{n(1+\hat{\theta})^2}\right)\frac{1}{\hat{\theta}(\hat{\theta}+1)}\sum_{i=1}^n y_i\right]^{-1}
\end{aligned}
$$

where $\hat{\theta} = \frac{\sum_{i=1}^n y_i}{n} - 1$. Then,

$$
\begin{aligned}
V_{obs}(\hat{\theta}) &= \left(1 - \frac{n}{\sum_{i=1}^n y_i}\right)\frac{n}{\frac{\sum_{i=1}^n y_i}{n} - 1} \\
&= \left(\frac{\sum_{i=1}^n y_i - n}{\sum_{i=1}^n y_i}\right)\frac{n^2}{\sum_{i=1}^n y_i - n} \\
&= \frac{n^2}{\sum_{i=1}^n y_i}
\end{aligned}
$$

PMF of $s_i|Y_{obs}, \Theta^{(t)}$ is

$$
\begin{aligned}
P(s_i|Y_{obs} = y_i, \Theta^{(t)}) &= \frac{P(s_i, y_i|\Theta^{(t)})}{P(y_i|\Theta^{(t)})} \\
&= \frac{{\theta^{*(t)}}^{s_i} e^{-\theta^{*(t)}}}{s_i!} \cdot \frac{(1-\alpha)^{y_i - s_i} e^{-(1-\alpha)}}{(y_i - s_i)!} \cdot \left[\frac{(\theta^{*(t)} + 1 - \alpha)^{y_i} e^{-(\theta^{*(t)}+1-\alpha)}}{y_i!}\right]^{-1} \\
&= \frac{y_i!}{(y_i - s_i)! s_i!}\left(\frac{\theta^{*(t)}}{\theta^{*(t)} + 1 - \alpha}\right)^{s_i}\left(\frac{1}{\theta^{*(t)} + 1 - \alpha}\right)^{y_i - s_i}
\end{aligned}
$$

Q8

(a) Complete data log likelihood under under expanded parameter space $\Theta = (\theta^*, \alpha)$

$$
\begin{aligned}
L(\Theta|S,Y) &= \prod_{i=1}^{n} P(S,Y|\Theta) \\
&= \prod_{i=1}^{n} P(Y|S,\Theta)P(S|\Theta) \\
&= \prod_{i=1}^{n} \frac{e^{-(1-\alpha)}(1-\alpha)^{y_i-s_i}}{(y_i-s_i)!}\frac{\theta^{*s_i}e^{-\theta^*}}{s_i!} \\
&= e^{-n(1-\alpha+\theta^*)} \cdot \theta^{*\sum_{i=1}^{n} s_i} \cdot \prod_{i=1}^{n} \frac{(1-\alpha)^{y_i-s_i}}{(y_i-s_i)! \cdot s_i!} \\
l(\Theta|S,Y) &= -n(1-\alpha) - n\theta^* + log(1-\alpha)\sum_{i=1}^{n}(y_i-s_i) + log(\theta^*)\sum_{i=1}^{n} s_i - \sum_{i=1}^{n} log\left[(y_i-s_i)!s_i!\right]
\end{aligned}
$$

(b) show that observed data and complete data model preserved under the expanded parameter space

Model preserved when $\theta = \theta^* - \alpha$. Then,

$$R(\Theta) = \theta^* - \alpha$$

Observed-data model is preserved

$$Y_{obs}|\Theta \sim P(Y_{obs}|\theta = R(\Theta))$$

Complete data model is preserved at $\alpha = \alpha_0$

$$
\begin{aligned}
P_x(Y_{com}|\Theta = (\theta^*, \alpha_0)) &= e^{-n(1-\alpha_0+\theta^*)} \cdot \theta^{*\sum_{i=1}^{n} s_i} \cdot \prod_{i=1}^{n} \frac{(1-\alpha_0)^{y_i-s_i}}{(y_i-s_i)! \cdot s_i!} \\
P(Y_{com}|\theta = \theta^*) &= e^{-n(1+\theta^*)} \cdot \theta^{*\sum_{i=1}^{n} s_i} \cdot \prod_{i=1}^{n} \frac{1}{(y_i-s_i)! \cdot s_i!}
\end{aligned}
$$

$P_x(Y_{com}|\Theta = (\theta^*, \alpha_0)) = P(Y_{com}|\theta = \theta^*)$ when $\alpha_0 = 0$, Complete data model is preserved at $\alpha_0 = 0$

(c) Derive PX-EM and show that converges in one iteration

0.1 E-step

$$
\begin{aligned}
Q(\Theta|\Theta^{(t)}) &= E[l(\Theta|Y_{com})|Y_{obs}, \Theta^{(t)}] \\
&= -n(1-\alpha) - n\theta^* + log(1-\alpha)\sum_{i=1}^{n}(y_i - E\left[s_i|Y, \Theta^{(t)}\right]) + log(\theta^*)\sum_{i=1}^{n} E\left[s_i|Y, \Theta^{(t)}\right]
\end{aligned}
$$

need to compute pmf of $s_i|Y_{obs}, \Theta^{(t)}$ is

$$
\begin{aligned}
P(s_i|Y_{obs} = y_i, \Theta^{(t)}) &= \frac{P(s_i, y_i|\Theta^{(t)})}{P(y_i|\Theta^{(t)})} \\
&= \frac{{\theta^{*(t)}}^{s_i} e^{-\theta^{*(t)}}}{s_i!} \cdot \frac{(1-\alpha)^{y_i - s_i} e^{-(1-\alpha)}}{(y_i - s_i)!} \cdot \left[\frac{(\theta^{*(t)} + 1 - \alpha)^{y_i} e^{-(\theta^{*(t)}+1-\alpha)}}{y_i!}\right]^{-1} \\
&= \frac{y_i!}{(y_i - s_i)! s_i!}\left(\frac{\theta^{*(t)}}{\theta^{*(t)} + 1 - \alpha}\right)^{s_i}\left(\frac{1}{\theta^{*(t)} + 1 - \alpha}\right)^{y_i - s_i}
\end{aligned}
$$

Thus $s_i|Y_{obs} = y_i, \Theta^{(t)} \sim Binomial\left(y_i, \frac{\theta^{*(t)}}{\theta^{*(t)}+1-\alpha}\right)$, Then $E[s_i|Y_{obs}, \theta^{(t)}] = y_i \cdot \frac{\theta^{*(t)}}{\theta^{*(t)}+1-\alpha}$

0.2 M-step

$$
Q(\Theta|\Theta^{(t)}) = -n(1-\alpha) - n\theta^* + log(1-\alpha)\sum_{i=1}^{n}(y_i - y_i \cdot \frac{\theta^{*(t)}}{\theta^{*(t)} + 1 - \alpha}) + log\theta^* \cdot \sum_{i=1}^{n} y_i \cdot \frac{\theta^{*(t)}}{\theta^{*(t)} + 1 - \alpha}
$$

Maximize the Qfunction by letting the first derivative 0

$$
\begin{aligned}
\frac{\partial Q(\Theta|\Theta^{(t)})}{\partial \theta^*} &=^{let} 0 \\
&= -n + \frac{\sum_{i=1}^{n} y_i \cdot \frac{\theta^{*(t)}}{\theta^{*(t)}+1-\alpha}}{\theta^*} \\
\theta^{*(t+1)} &= \frac{1}{n} \cdot \sum_{i=1}^{n} y_i \cdot \frac{\theta^{*(t)}}{\theta^{*(t)} + 1 - \alpha} \\
\frac{\partial Q(\Theta|\Theta^{(t)})}{\partial \alpha} &=^{let} 0 \\
&= n - \frac{\sum_{i=1}^{n}\left(y_i - y_i \cdot \frac{\theta^{*(t)}}{\theta^{*(t)}+1-\alpha}\right)}{1-\alpha} \\
&= n - \frac{\sum_{i=1}^{n} y_i\left(1 - \frac{\theta^{*(t)}}{\theta^{*(t)}+1-\alpha}\right)}{1-\alpha} \\
\hat{\alpha} &= 1 - \frac{\sum_{i=1}^{n} y_i\left(1 - \frac{\theta^{*(t)}}{\theta^{*(t)}+1-\alpha}\right)}{n}
\end{aligned}
$$

First iteration is,

$$\begin{aligned}
\theta^{(t+1)} &= R(\Theta^{(t+1)}) \\
&= \hat{\theta}^* - \hat{\alpha} \\
&= \frac{1}{n} \cdot \sum_{i=1}^{n} y_i \cdot \frac{\theta^{*(t)}}{\theta^{*(t)} + 1 - \alpha} - 1 + \frac{\sum_{i=1}^{n} y_i \left(1 - \frac{\theta^{*(t)}}{\theta^{*(t)}+1-\alpha}\right)}{n} \\
&= \frac{\sum_{i=1}^{n} y_i}{n} - 1
\end{aligned}$$

Which is same value in Q7 (unique solution of EM-algorithm), it converges in first iteration.

Q9

May 21, 2019

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import scipy.stats as stats
        from scipy.special import digamma
        from scipy.stats import gamma,norm
```

0.1 prior setting

Let $(\sigma^2)^{-1} = \tau$ Then,

$$\begin{aligned} y_i|\mu,\tau &\sim N(\mu,\tau^{-1}) \\ \tau &\sim Gamma(a_0,b_0) \\ \mu|\tau &\sim N(0,(\tau/k)^{-1}) \end{aligned}$$

Thus

$$p(\mu,\tau|Y) = \frac{p(\mu,\tau)p(Y|\mu,\tau)}{p(Y)} = \frac{p(\mu|\tau)p(\tau)p(Y|\mu,\tau)}{p(Y)}$$

1 (a) Derive MF variational distribution of (μ,τ) and ELBO

Let $\theta = (a_0,b_0,k)$, $p = p(\mu,\tau|Y,\theta)$ and $q = q(\mu,\tau)$

$$p(\mu,\tau|Y,\theta) \approx q(\mu,\tau) = q_1(\mu)q_2(\tau)$$

Thus, mean-field variational distribution is

$$q(\mu,\tau) = q_1(\mu|t,u)q_2(\tau|v,w)$$

where $q_1 \sim N(t,u^{-1})$ and $q_2 \sim Gamma(v,w)$, variational parameters $\lambda = (t,u,v,w)$. ELBO is

$$\begin{aligned} ELBO(\lambda) =& E_q[\log p(\mu,\tau,Y)|\lambda] - E_q[\log q(\mu,\tau)|\lambda] \\ =& \sum_{i=1}^{n} E_q[\log p(y_i|\mu,\tau)|\lambda] + E_q[\log p(\mu|\tau)|t,u] + E_q[\log p(\tau)|w,v] \\ & - E_{q_1}[logq_1(\mu|t,u)|t,u] - E_{q_2}[logq_2(\tau)|v,w] \end{aligned}$$

2 (b) derive the coordinate descent algorithm

$$
\begin{aligned}
\log q_1^*(\mu) &\propto E_{q_2}\left[\log p(\mu,\tau,Y)\right] \propto E_{q_2}\left[\log p(Y|\mu,\tau)+\log p(\mu|\tau)\right]\\
&\propto -\frac{1}{2}E_{q_2}[\tau]\cdot\sum_{i=1}^{n}(y_i-\mu)^2-\frac{E_{q_2}[\tau]}{2k}\mu^2\\
&\propto -\frac{1}{2}\left[E_{q_2}[\tau]\left(n+\frac{1}{k}\right)\mu^2-2E_{q_2}[\tau]\mu\sum_{i=1}^{n}y_i\right]
\end{aligned}
$$

Thus

$$
t=\frac{k\sum_{i=1}^{n}y_i}{nk+1},u=\frac{E_{q2}[\tau](nk+1)}{k}
$$

as $q_1^*(\mu)\sim N(t,u^{-1})$

$$
\begin{aligned}
E_{q_1}[\mu] &= t=\frac{k\sum_{i=1}^{n}y_i}{nk+1}\\
E_{q_1}[\mu^2] &= u^{-1}+t^2=\left[\frac{k}{E_{q2}[\tau](nk+1)}\right]+\left[\frac{k\sum_{i=1}^{n}y_i}{nk+1}\right]^2
\end{aligned}
$$

where $E_{q2}[\tau]=v/w$

$$
\begin{aligned}
\log q_2^*(\tau) &\propto E_{q_1}\left[\log p(\mu,\tau,Y)\right] \propto E_{q_1}\left[\log p(Y|\mu,\tau)+\log p(\mu|\tau)+\log p(\tau)\right]\\
&\propto (a_0-1)\log\tau-b_0\tau+\frac{n}{2}\log\tau-\frac{1}{2}\tau E_{q_1}\left[\sum_{i=1}^{n}(y_i-\mu)^2\right]+\frac{1}{2}\log\tau-\frac{\tau}{2k}E_{q_1}\left[\mu^2\right]\\
&\propto \log(\tau)\left(a_0-1+\frac{n+1}{2}\right)-\tau\left(b_0+\frac{1}{2}E_{q_1}\left[\sum_{i=1}^{n}(y_i-\mu)^2\right]+\frac{1}{2k}E_{q_1}\left[\mu^2\right]\right)
\end{aligned}
$$

Thus

$$
v=a_0+\frac{n+1}{2},w=b_0+\frac{1}{2}E_{q_1}\left[\sum_{i=1}^{n}(y_i-\mu)^2\right]+\frac{1}{2k}E_{q_1}\left[\mu^2\right]
$$

where,

$$
E_{q_1}\left[\sum_{i=1}^{n}(y_i-\mu)^2\right]=\sum y_i^2-2\sum y_iE[\mu]+E[\mu^2]
$$

$q_2^*(\tau)\sim Gamma(v,w)$

3 (c) write down python code

input data

```
In [2]: Y = np.array([1.64,1.70,1.72,1.74,1.82,1.82,1.82,
                      1.90,2.08,1.78,1.86,1.96,1.96,2.00,2.00])
```

setting the initail value

In [45]:
```
def mfvb(Y,maxiter = 100):
    a_0=2
    b_0=60
    k=2
    n= len(Y)
    expected_mu = 0
    expected_mu2 = 1

    v = a_0 +(n+1)/2
    param_out=[]
    i=0
    while i<maxiter:
        w = b_0 + 0.5*((Y**2).sum() -2 *Y.sum() * expected_mu
                       + expected_mu2) + expected_mu2/(2*k)
        expected_tau = v/w
        t = (k*Y.sum())/(n*k+1)
        u = (expected_tau * (n*k+1))/k
        expected_mu = t
        expected_mu2 = 1/u + t**2
        param_out.append([v,w,t,u])
        i= i+1
        if i>1:
            if param_out[-1] == param_out[-2]:
                break
    out = pd.DataFrame(param_out)
    out.columns = ['v','w','t','u']
    return(out)
```

In [46]:
```
mfvb(Y)
```

Out[46]:

	v	w	t	u
0	10.0	86.625200	1.793548	1.789318
1	10.0	38.846321	1.793548	3.990082
2	10.0	38.615133	1.793548	4.013970
3	10.0	38.614014	1.793548	4.014087
4	10.0	38.614009	1.793548	4.014087
5	10.0	38.614009	1.793548	4.014087
6	10.0	38.614009	1.793548	4.014087
7	10.0	38.614009	1.793548	4.014087
8	10.0	38.614009	1.793548	4.014087

In [13]:
```
ksigma2 = 1/4.014087
sigma2 = ksigma2/2
print(ksigma2,sigma2)
```

```
0.24912265229926506 0.12456132614963253
```

$$q_2(\tau) \sim Gamma(10, 38.614009)$$
$$q_1(\mu) \sim N(1.793548, 0.249123)$$

4 (d) draw contour of the target posterior distribution and compare it with MFVB

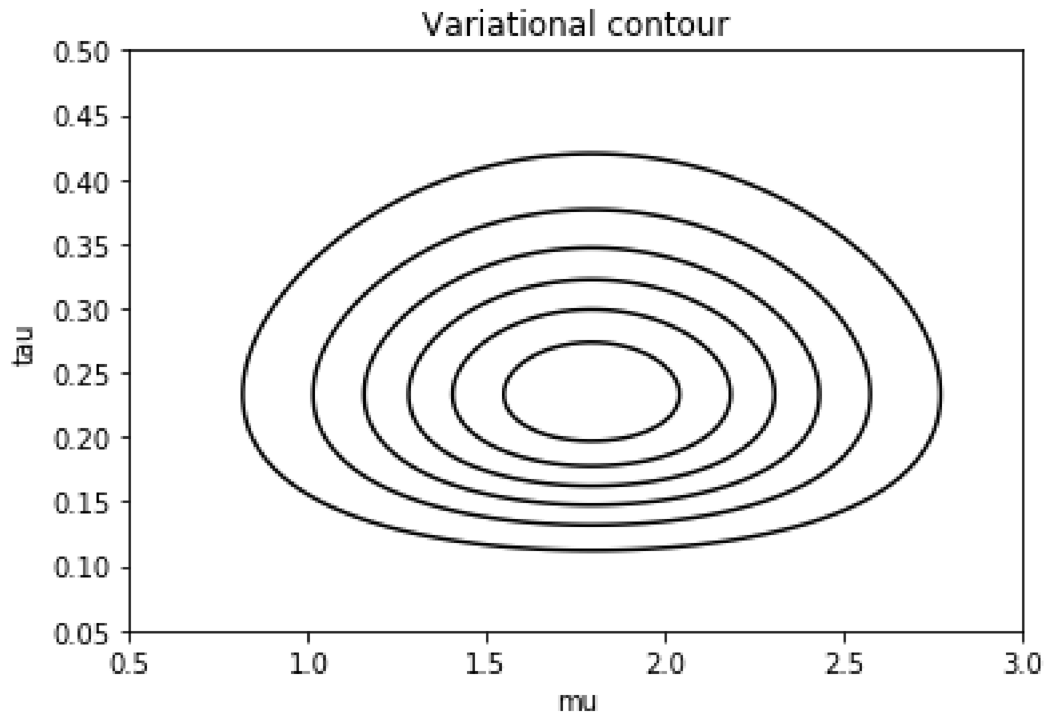
4.1 VI Contour

$$q_2(\tau) \sim Gamma(10, 38.614009)$$
$$q_1(\mu) \sim N(1.793548, 0.249123)$$

```
In [5]: def q(mu,tau):
            out = gamma.pdf(tau,a = 10,scale = 1/38.614009)*\
                  norm.pdf(mu,loc = 1.793548,
                           scale = np.sqrt(0.249123))
            return(out)
```

```
In [28]: tau = np.linspace(0.05, 0.5, 100)
         mu = np.linspace(0.5, 3, 100)

         Mu, Tau = np.meshgrid(mu, tau)
         vi = q(Mu, Tau)
         plt.contour( Mu,Tau, vi, colors='black')
         plt.title('Variational contour')
         plt.xlabel('mu')
         plt.ylabel('tau')
         plt.show()
```



4.2 Posterior contour

$$p(\mu, \tau | Y) \propto p(\mu, \tau, Y) \propto p(Y|\mu, \tau)p(\mu|\tau)p(\tau)$$

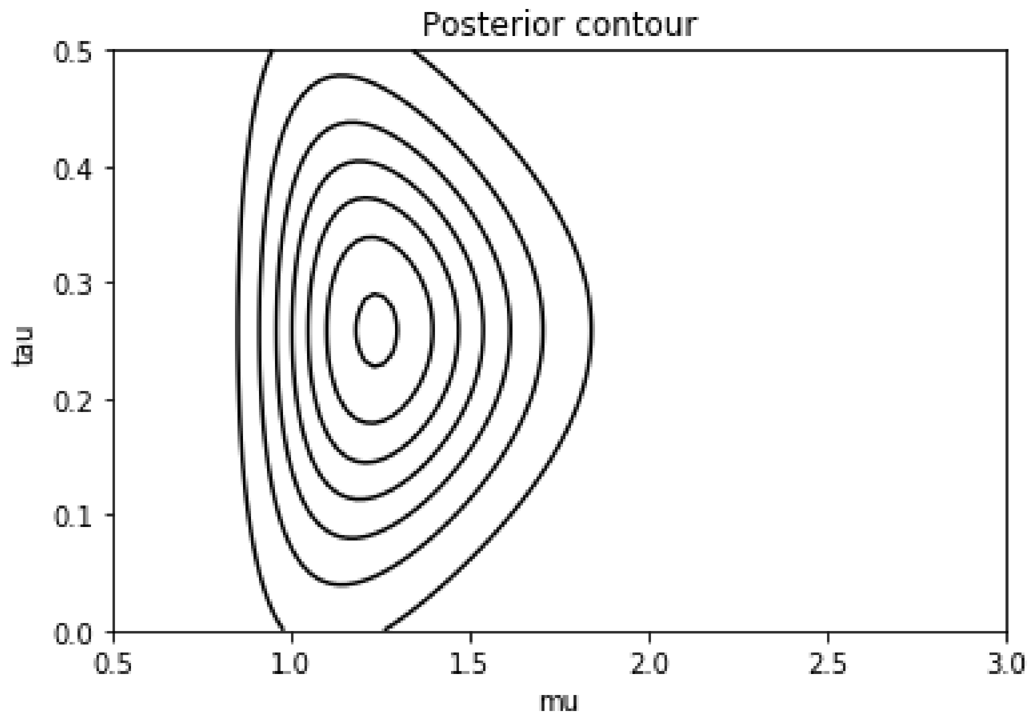
Thus

where $a_0 = 2, b_0 = 60$ and $k = 2$

```
In [8]: def p(mu,tau,Y):
            out = np.prod(norm.pdf(Y,loc = mu,
                           scale = np.sqrt(tau**(-1))))*\
            norm.pdf(mu,loc = 0,
                     scale = np.sqrt(2*(tau**(-1))))*\
            gamma.pdf(tau,a = 2,scale = 1/60)
            return(out)
```

```
In [29]: post = np.zeros([100,100])
         for i in range(100):
             for j in range(100):
                 post[i,j] = p(mu[i], tau[j],Y)
```

```
In [27]: plt.contour(Mu, Tau, post, colors='black')
         plt.title('Posterior contour')
         plt.xlabel('mu')
         plt.ylabel('tau')
         plt.show()
```

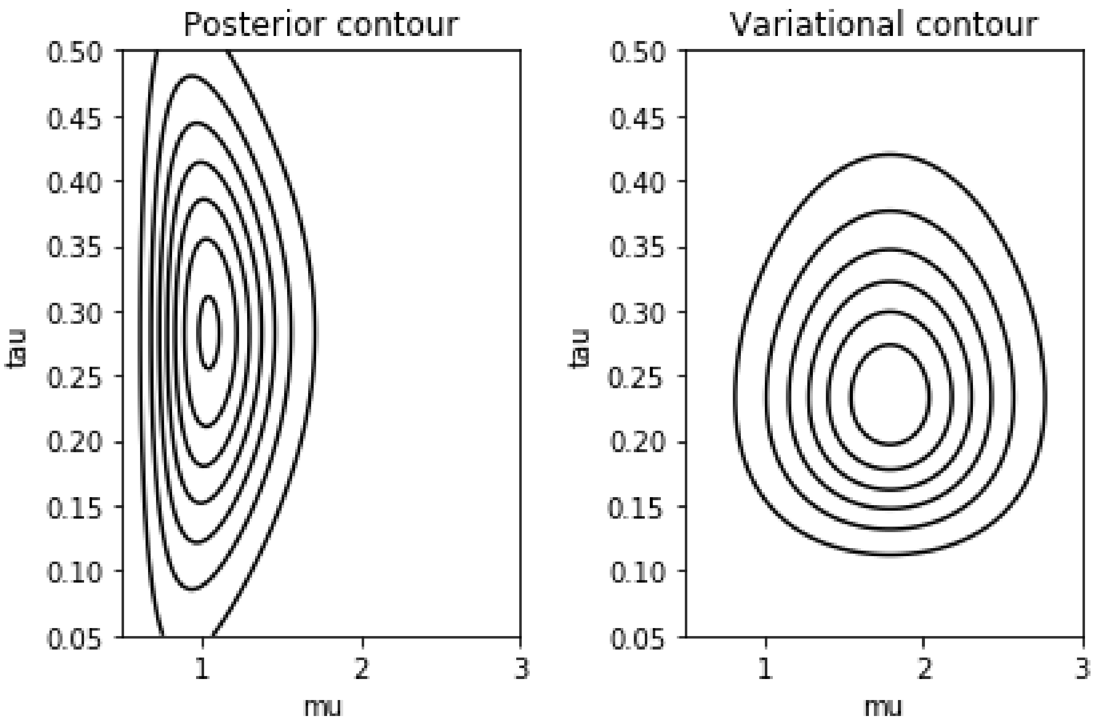


4.3 Compare

```
In [40]: def posteriorplot(ax):
             ax.contour(Mu, Tau, post, colors='black')
             ax.set_xlabel('mu')
             ax.set_ylabel('tau')
             ax.set_title('Posterior contour')
```

```
In [41]: def viplot(ax):
             ax.contour( Mu,Tau, vi, colors='black')
             ax.set_xlabel('mu')
             ax.set_ylabel('tau')
             ax.set_title('Variational contour')
```

```
In [42]: fig, ((ax1), (ax4)) = plt.subplots(nrows=1, ncols=2)
         posteriorplot(ax1)
         viplot(ax4)
         plt.tight_layout()
```



It seems that mean-field vairational distribution overestimate variance of μ and underestimate variance of τ