Student: Quoc Tuan Vinh, Ngo
ID: 704526

CS-E4820 – Machine Learning: Advanced Probabilistic Methods Homework Assignment 3

Problem 1. *"Poisson-Gamma"*

Suppose we have N i.i.d. observations $\boldsymbol{x} = \{x_i\}_{i=1}^{N}$ from a Poisson (λ) distribution with parameter λ that has a conjugate prior

$$\lambda \sim \text{Gamma}(a,b)$$

For a Poisson distribution, we know the formula:

$$p(x_i;\lambda) = \frac{e^{-\lambda}\lambda^{x_i}}{x_i!}, for\ x_{\boldsymbol{i}} \in \boldsymbol{x}$$

We can derive the posterior distribution as follows:

$$p(\lambda|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|\lambda)p(\lambda)}{p(\boldsymbol{x})} \propto p(\boldsymbol{x}|\lambda)p(\lambda) = \prod_{i=1}^{N}\frac{e^{-\lambda}\lambda^{x_i}}{x_i!} \times \frac{1}{\Gamma(a)}b^a\lambda^{a-1}e^{-b\lambda}$$

$$\propto e^{-N\lambda}\lambda^{\sum_{i=1}^{N}x_i} \times \lambda^{a-1}e^{-b\lambda} = e^{-\lambda(N+b)}\lambda^{a+\sum_{i=1}^{N}x_i-1} \propto Gamma(\lambda|a_n,b_n)$$

with:

$$a_n = a + \sum_{i=1}^{N}x_i$$

$$b_n = b + N$$

Problem 2. *"Multivariate Gaussian"*

Suppose we have N i.i.d. observations $\boldsymbol{X} = \{x_i\}_{i=1}^{N}$ from a multivariate Gaussian distribution

$$x_i|\ \mu \sim \mathcal{N}(\mu,\Sigma)$$

with unknown μ and known Σ. Also, the mean parameter μ also follows Gaussian distribution as

$$\mu \mid \mathcal{N}(m_0,S_0)$$

a. *Derive the posterior distribution* $p(\mu|X)$ *of the mean* μ as follows:

$$p(\mu|\boldsymbol{X}) = \frac{p(\boldsymbol{X}|\mu)p(\mu)}{p(\boldsymbol{X})} \propto p(\boldsymbol{X}|\mu)p(\mu)$$

$$= \prod_{i=1}^{N}(2\pi)^{\frac{-D}{2}}|\Sigma|^{\frac{-1}{2}}e^{\frac{-1}{2}(x_i-\mu)^T\Sigma^{-1}(x_i-\mu)} \times (2\pi)^{\frac{-D}{2}}|S_0|^{\frac{-1}{2}}e^{\frac{-1}{2}(\mu-m_0)^TS_0{}^{-1}(\mu-m_0)}$$

$$\propto e^{\frac{-1}{2}\sum_{i=1}^{N}(x_i - \mu)^T\Sigma^{-1}(x_i-\mu) - \frac{1}{2}(\mu - m_0)^T {S_0}^{-1}(\mu - m_0)}$$

with D as the dimension. For easier visualization, we can take the natural logarithm from both sides of the above expansion as below:

$$\begin{aligned}
&\ln p(\mu|\boldsymbol{X}) \\
&\propto \frac{-1}{2}\sum_{i=1}^{N}(x_i - \mu)^T\Sigma^{-1}(x_i - \mu) - \frac{1}{2}(\mu - m_0)^T {S_0}^{-1}(\mu - m_0) \\
&\propto \frac{-1}{2}N\mu^T\Sigma^{-1}\mu + \sum_{i=1}^{N}\mu^T\Sigma^{-1}x_i - \frac{1}{2}\mu^T S_0^{-1}\mu + \mu^T S_0^{-1}m_0 \\
&= -\left[\frac{1}{2}\mu^T(N\Sigma^{-1} + S_0^{-1})\,\mu - \mu^T\left(\sum_{i=1}^{N}\Sigma^{-1}x_i + S_0^{-1}m_0\right)\right]
\end{aligned}$$

Continue by applying completing the square method:

$$\begin{aligned}
&\ln p(\mu|\boldsymbol{X}) \\
&\propto -\left[\frac{1}{2}\left(\mu - (N\Sigma^{-1} + S_0^{-1})^{-1}\left(\Sigma^{-1}\sum_{i=1}^{N}x_i + S_0^{-1}m_0\right)\right)^T (N\Sigma^{-1} + S_0^{-1})\left(\mu \right.\right. \\
&\qquad\left.\left. - (N\Sigma^{-1} + S_0^{-1})^{-1}\left(\Sigma^{-1}\sum_{i=1}^{N}x_i + S_0^{-1}m_0\right)\right)\right]
\end{aligned}$$

Therefore, returning to our original posterior function, we have:

$$\begin{aligned}
&p(\mu|\boldsymbol{X}) \\
&\propto e^{-\left[\frac{1}{2}\left(\mu - (N\Sigma^{-1} + S_0^{-1})^{-1}\left(\Sigma^{-1}\sum_{i=1}^{N}x_i + S_0^{-1}m_0\right)\right)^T (N\Sigma^{-1} + S_0^{-1})\left(\mu - (N\Sigma^{-1} + S_0^{-1})^{-1}\left(\Sigma^{-1}\sum_{i=1}^{N}x_i + S_0^{-1}m_0\right)\right)\right]} \\
&\propto \mathcal{N}(\mu_n, \Sigma_n) \text{ (which is also multivariate Gaussian distribution)}
\end{aligned}$$

with:

$$\mu_n = (N\Sigma^{-1} + S_0^{-1})^{-1}\left(\Sigma^{-1}\sum_{i=1}^{N}x_i + S_0^{-1}m_0\right)$$

$$\Sigma_n = (N\Sigma^{-1} + S_0^{-1})^{-1}$$

b. Comparison between Bayesian estimate & bivariate Gaussian

We solve the problem by the code below:

```
#Problem 2
import numpy as np
import random
```

```
N = 10
np.random.seed(5)

#Generate 10 observations
mean = [0,0]
sigma = [[1,0],[0,1]]
data = np.random.multivariate_normal(mean, sigma, 10) #numpy provides this
function that it generates the variables matrix
print('The simulated variables are: \n',data)

#Maximum Likelihood
print('\n1. Maximum Likelihood:')
ML_mean = 1/N * np.sum(data, axis = 0)
print('Mean: \n', ML_mean)

_data = data.copy()
_data -= _data.mean(axis = 0)
ML_sigma = 1/ N * _data.T @ _data
print('Covariance Matrix: \n', ML_sigma)

#Bayesian Posterior
print('\n2. Bayesian Posterior:')
mean = np.array([[0,0]]).T
sigma = np.array([[1,0],[0,1]])
m0 = np.array([[0,0]]).T
S0 = np.array([[0.1,0],[0,0.1]])

pos_sigma = np.linalg.inv((N * np.linalg.inv(sigma) + np.linalg.inv(S0)))
pos_mean = pos_sigma @ (np.linalg.inv(sigma) @ np.reshape(np.sum(data, axis =
0), (np.sum(data, axis = 0).shape[0],1)) + np.linalg.inv(S0) @ m0)

print('Mean: \n',pos_mean.T)
```

The above code's results are:

```
The simulated variables are:
 [[ 0.44122749 -0.33087015]
 [ 2.43077119 -0.25209213]
 [ 0.10960984  1.58248112]
 [-0.9092324  -0.59163666]
 [ 0.18760323 -0.32986996]
 [-1.19276461 -0.20487651]
 [-0.35882895  0.6034716 ]
 [-1.66478853 -0.70017904]
 [ 1.15139101  1.85733101]
 [-1.51117956  0.64484751]]

1. Maximum Likelihood:
Mean:
 [-0.13161913  0.22786068]
```

```
2. Bayesian Posterior:
Mean:
 [[-0.06580957  0.11393034]]
```

The estimate is the mean in each case. Note that Maximum Likelihooh estimate is calculated due to the formula from slide 27, lecture 03, and the Bayesian estimate is calculated according to Problem 2a.

Comparing between the two means, one can easily see that Bayesian estimate (posterior mean) has closer mean to the true value $\mu = [0,0]^T$ than the maximum likelihood.

Problem 3. *"Wishart distribution"*

Suppose we have a multivariate distribution $\mathcal{N}(\mu,\wedge^{-1})$, where $\wedge$ is the precision matrix.

a. The Wishart distribution is a distribution for nonnegative-definite matrix-valued random variables

$$\wedge \sim \mathcal{W}(\wedge|W, v)$$

Then, the mean and variance are respectively as follows:
$E(\wedge) = vW$
$Var(\wedge_{ij}) = n(w_{ij}^2 + w_{ii}w_{jj})$

b.
"Specify parameters"

We know that parameters of Wishart distribution are W, v .

According to the question, we know that expected value of $\wedge$ would be equal to A:
$E(\wedge) = vW = A$, which is also the mean of the Wishart distribution.

Since the only requirement that we have for these parameters is that v needs to be higher than p (=2). Therefore, at this point, let us pick 2 as v degree of freedom value (value of 2 satisfies the requirement of v, which is more than (p – 1 = 1)). From A, we can calculate W as follows:

$W = \frac{A}{v}$, which results as the matrix $W = \begin{bmatrix} 1 & 0.15 \\ 0.15 & 0.25 \end{bmatrix}$

"Simulation"
Simulate samples by function *scipy.stats.wishart.rvs(df, scale, size)*, with df is degree of freedom v, scale is the scale matrix W and size is 1, 10, 1000. The results below show the average matrix of each sample size case.

```
Simulation of Wishart Distribution: The mean matrix

1. Varying the sample size (degree of freedom fixed = 2):
```

```
For 1 samples:
[[0.46315593 0.21269774]
 [0.21269774 0.99929394]]

For 10 samples:
[[1.43274053 0.42763495]
 [0.42763495 0.44792232]]

For 1000 samples:
[[2.03866512 0.33413889]
 [0.33413889 0.51349856]]
```

One easily sees again that as the number of samples increases, the average matrix converges to A from sample size 1 → 10 → 1000.

c. *"Effect of Parameter Values"*

Since $E(\wedge) = vW$ is fixed as A, by changing v, we can also change the scale matrix accordingly.

Speaking of v, we know that "*the least informative, proper Wishart prior is obtained by setting $v = p$*". Therefore, by increasing degree of freedom, we can arbitrarily make the matrices converge to A.

To show that, the value of degree of freedom is adjusted to be [2, 20, 200, 2000] differently, and again, the average matrices of each case are shown here:

```
2. Varying degree of freedom (sample size fixed = 50) :

For degree of freedom of 2:
[[2.20493495 0.32515555]
 [0.32515555 0.41953387]]

For degree of freedom of 20:
[[2.04055759 0.3200758 ]
 [0.3200758  0.51223015]]

For degree of freedom of 200:
[[2.00881455 0.30718032]
 [0.30718032 0.50497034]]

For degree of freedom of 2000:
[[2.00239093 0.3023514 ]
 [0.3023514  0.5017859 ]]
```

From the results, we can easily the above-mentioned affirmation.

```
#Problem 3
#Wishart distribution

import scipy.stats
```

```
A = np.array([[2, 0.3],[0.3, 0.5]]) #precision matrix of the multivariate normal
distribution

#b
v = 2 #more than p (=2) - 1
W = A / v
print('The scale matrix:')
print(W)

print('\nSimulation of Wishart Distribution: The mean matrix')
print('\n1. Varying the sample size (degree of freedom fixed = 2):')
sizes = [1,10,1000]
for size in sizes:
    print('\nFor %s samples:' % size)
    wishart = scipy.stats.wishart.rvs(v,W,size, random_state = 5)
    if size != 1:
        print(np.mean(wishart, axis = 0))
    elif size == 1:
        print(wishart)

#c
print('\n2. Varying degree of freedom (sample size fixed = 50) :')
vs = [2,20,200,2000]
size = 50
for v in vs:
    W = A / v
    print('\nFor degree of freedom of %s:' % v)
    wishart = scipy.stats.wishart.rvs(v,W,size, random_state = 5)
    print(np.mean(wishart, axis = 0))
```