INF367A Exercise 5

Naphat Amundsen

February 13, 2020

1 Introduction

This exercise is about Bayesian inference on sample datasets. The functions are continuous, and we are supposed to utilize conjugate priors to estimate parameters.

In case future self forgets: Normally, when you want to fit a probability density function to your data you find the MLEs for the model parameters. Now, we don't really care what the maximum likelihood actually is, as long as we maximize it. So when deriving MLEs you can basically remove every factor that does not depend on the parameter you want to optimize for since they are monotonic transformations (preserves extrema). This gives us much freedom when deriving.

When doing stuff the Bayesian way, we actually include the prior into the equation. So instead of just maximizing likelihood, we maximize the likelihood times the prior: $P(X|\theta)P(\theta)$. In this exercise (and often generally) we assume that the parameter itself is distributed with some function. When simplifying $P(X|\theta)P(\theta)$ we can sometimes get the equation into a form that is proportional to a probability density function. If you do manage to get $P(X|\theta)P(\theta)$ in a form of another pdf, you have found a conjugate prior. You can then easily use Monte Carlo approximation to estimate the optimal parameter value since most known distributions are probably implemented by someone already.

2 Poisson-Gamma

1. **Poisson-Gamma**
   Poisson distribution can be used to model count data. Suppose you have n i.i.d. observations $X = \{x_i\}_{i=1}^n$ from a Poisson(λ) distribution with a rate parameter λ that has a conjugate prior $\lambda \sim \Gamma a, b$ with the shape and rate hyperparameters a and b.

   (a) Derive the posterior distribution $P(\lambda|x)$.

   (b) Load the data set exercise5_1.txt (100 one-dimensional data points). Compute the posterior of λ given this data.

   (c) Estimate the predictive distribution using Monte Carlo approximation. Plot the histogram of the predictive samples. Compare it to the original data. Do your results make sense? (If you are inclined to answer no then you have probably done something wrong along the way.)

   (d) Compute 50% and 95% credible intervals for the predictive distribution. Hint: You should start by sorting your samples.

Note: There are several different ways to parameterize the gamma distribution. If you use the gamma distribution from scipy, then Gamma(a0,b0) distribution is specified by gamma(a=a0, scale=1/b0).

Solution:
1a) Poisson distribution is:

$$P(K = k) = \frac{\lambda^k e^{-\lambda}}{k!} \tag{1}$$

Then assuming that data points are i.i.d

$$\begin{aligned}
P(\lambda|x) &= \frac{P(\lambda)P(x|\lambda)}{P(x)} \propto P(\lambda)P(x|\lambda) &(2)\\
&= \frac{1}{\Gamma(a)} b^a \lambda^{a-1} e^{-b\lambda} \left[\frac{\lambda^{x_1} e^{-\lambda}}{x_1!} \cdot \frac{\lambda^{x_2} e^{-\lambda}}{x_2!} \cdot \ldots \cdot \frac{\lambda^{x_n} e^{-\lambda}}{x_n!}\right] &(3)\\
&\propto \lambda^{a-1} e^{-b\lambda} \left(\lambda^{x_1} e^{-\lambda} \cdot \lambda^{x_2} e^{-\lambda} \cdot \ldots \cdot \lambda^{x_n} e^{-\lambda}\right) &(4)\\
&= \lambda^{a-1+\sum_{i=1}^{n} x_i} e^{-(n+b)\lambda} &(5)\\
&\propto \underline{\underline{\text{Gamma}(a - 1 + \sum_{i=1}^{n} x_i, n + b)}} &(6)
\end{aligned}$$

1b) The prior distribution for λ is plotted against the posterior distribution. The prior is Gamma$(2, 2)$.

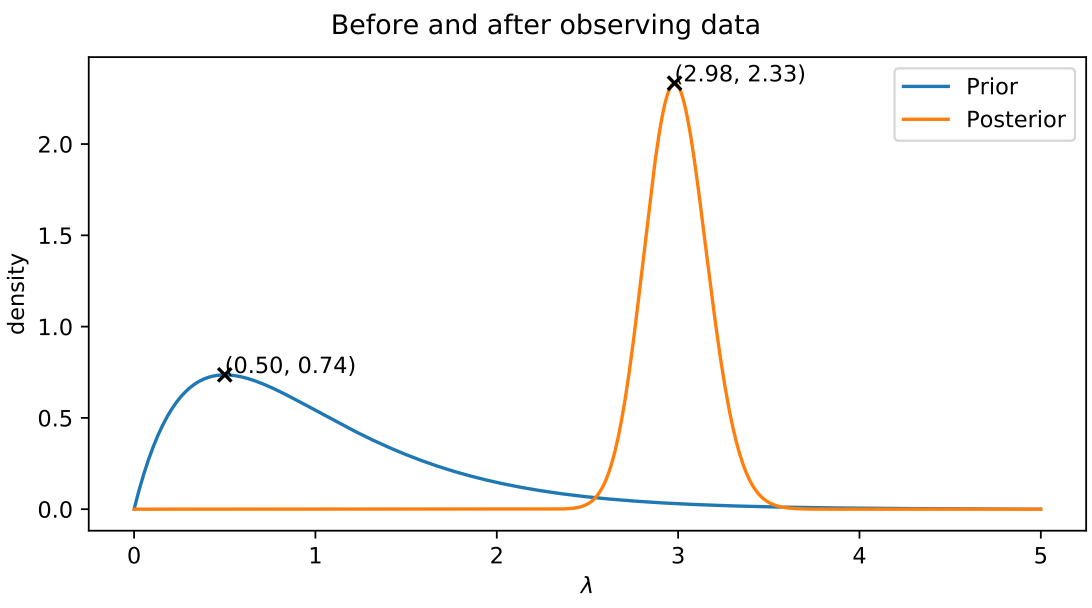

Figure 1: We can see that the posterior distribution centers around $\lambda \approx 3$

1c) Then using the posterior approximated λ, we try to sample from the predictive function (Poisson(λ)) to see if the distribution looks alike.

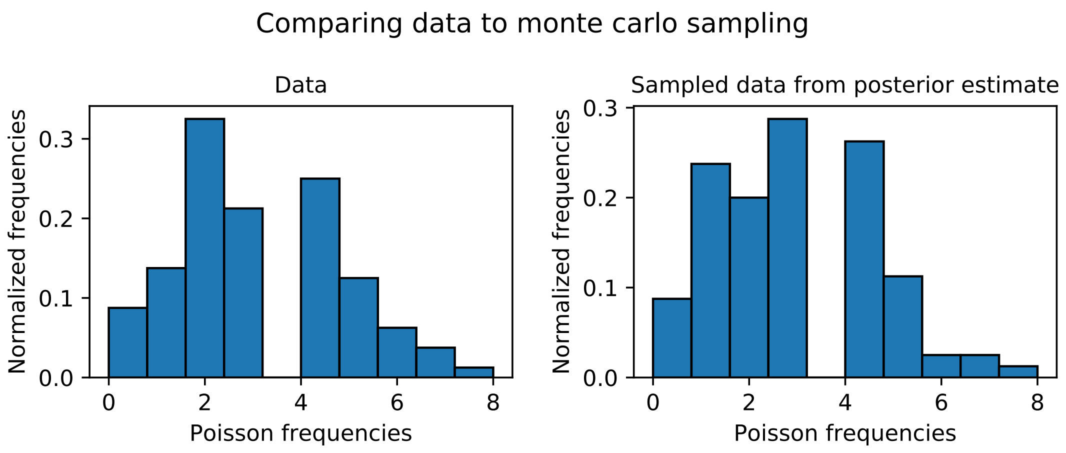

Figure 2: The data and the monte carlo sampled points seem to come from the same distributions.

1d)
50% Credibility interval: (1, 3)
95% Credibility interval: (0, 6)

3 Multivariate Gaussian

2. **Multivariate Gaussian**
   Suppose we have N i.i.d observations $X = x_{i i=1}^{N}$ from a multivariate Gaussian distribution

$$\boldsymbol{x}_i|\boldsymbol{\mu} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

   with unknown mean parameter μ and a known covariance matrix Σ. As prior information on the mean parameter we have

$$\boldsymbol{\mu} \sim \mathcal{N}(\boldsymbol{m}_0, \boldsymbol{S}_0).$$

   (a) Derive the posterior distribution $P(\mu|X)$ of the mean parameter μ. Hints: It may be more comfortable to work with the log-posterior $\log P(\mu|X)$. You can drop all terms that do not depend on μ. The log-posterior of a multivariate Gaussian is a second degree polynomial so you can use "completing the square".

   (b) Load the data set exercise5_2.txt(100 two-dimensional data points). Compute the posterior of μ given this data. Compute the posterior of μ given this data.

   (c) (Optional) Plot the posterior in 3D. Do your results make sense?

2a) Multivariate normal distribution probably density function:

$$f_N(x_i|\mu, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right] \tag{7}$$

Then to calculate the posterior:

$$\begin{aligned} P(\mu|X) &= \frac{P(\mu)P(X|\mu)}{P(X)} \propto P(\mu)P(X|\mu) & (8)\\ &= \exp\left[-\frac{1}{2}(\mu-m_0)^T S_0^{-1}(\mu-m_0)\right]\prod_{i=1}^{N}\exp\left[-\frac{1}{2}(x_i-\mu)^T\Sigma^{-1}(x_i-\mu)\right] & (9)\\ & & (10) \end{aligned}$$

Solve for the logarithm of the posterior instead in order to make math easier. The goal here is to get the terms in quadratic form:

$$\frac{1}{2}\theta^T A\theta + b^T\theta \tag{11}$$

which will implies that you have a Gaussian conjugate (if can get the quadratic form).

$$\log P(\mu|X) = \log \exp\left[-\frac{1}{2}(\mu - m_0)^T S_0^{-1}(\mu - m_0)\right] \tag{12}$$

$$+ \log \prod_{i=1}^{N} \exp\left[-\frac{1}{2}(x_i - \mu)^T \Sigma^{-1}(x_i - \mu)\right] \tag{13}$$

$$= -\frac{1}{2}(\mu - m_0)^T S_0^{-1}(\mu - m_0) + \sum_{i=1}^{N} \log \exp\left[-\frac{1}{2}(x_i - \mu)^T \Sigma^{-1}(x_i - \mu)\right] \tag{14}$$

$$= -\frac{1}{2}(\mu - m_0)^T S_0^{-1}(\mu - m_0) + \sum_{i=1}^{N} -\frac{1}{2}(x_i - \mu)^T \Sigma^{-1}(x_i - \mu) \tag{15}$$

$$= -\frac{1}{2}\left[(\mu - m_0)^T S_0^{-1}(\mu - m_0) + \sum_{i=1}^{N}(x_i - \mu)^T \Sigma^{-1}(x_i - \mu)\right] \tag{16}$$

$$= -\frac{1}{2}\big[\mu^T S_0^{-1}\mu - m_0^T S_0^{-1}\mu - \mu^T S_0^{-1} m_0 - m_0^T S_0^{-1} m_0 \tag{17}$$

$$+ \sum_{i=1}^{N} x_i^T \Sigma^{-1} x_i - x_i^T \Sigma^{-1}\mu - \mu^T \Sigma^{-1} x_i + \mu^T \Sigma^{-1}\mu\big] \tag{18}$$

$$\propto -\frac{1}{2}\big[\mu^T S_0^{-1}\mu - m_0^T S_0^{-1}\mu - \mu^T S_0^{-1} m_0 \tag{19}$$

$$+ \sum_{i=1}^{N} -x_i^T \Sigma^{-1}\mu - \mu^T \Sigma^{-1} x_i + \mu^T \Sigma^{-1}\mu\big] \tag{20}$$

$$\tag{21}$$

When we have a symmetric A, then this relation holds: $x^T Ay = y^T Ax$

$$\Rightarrow \log P(\mu|X) \propto -\frac{1}{2}\big[\mu^T S_0^{-1}\mu - m_0^T S_0^{-1}\mu - \mu^T S_0^{-1} m_0 \tag{22}$$

$$+ \sum_{i=1}^{N} -x_i^T \Sigma^{-1}\mu - \mu^T \Sigma^{-1} x_i + \mu^T \Sigma^{-1}\mu\big] \tag{23}$$

$$= -\frac{1}{2}\big[\mu^T S_0^{-1}\mu - m_0^T S_0^{-1}\mu - m_0^T S_0^{-1}\mu \tag{24}$$

$$+ \sum_{i=1}^{N} -x_i^T \Sigma^{-1}\mu - x_i^T \Sigma^{-1}\mu + \mu^T \Sigma^{-1}\mu\big] \tag{25}$$

$$= -\frac{1}{2}\big[\mu^T S_0^{-1}\mu - 2m_0^T S_0^{-1}\mu + \sum_{i=1}^{N} -2x_i^T \Sigma^{-1}\mu + \mu^T \Sigma^{-1}\mu\big] \tag{26}$$

$$= -\frac{1}{2}\big[\mu^T S_0^{-1}\mu - 2m_0^T S_0^{-1}\mu - N\mu^T \Sigma^{-1}\mu - 2\sum_{i=1}^{N} x_i^T \Sigma^{-1}\mu\big] \tag{27}$$

$$= -\frac{1}{2}\big[\mu^T S_0^{-1}\mu + N\mu^T \Sigma^{-1}\mu - 2N\bar{X}^T \Sigma^{-1}\mu - 2m_0^T S_0^{-1}\mu\big] \tag{28}$$

$$= -\frac{1}{2}\left[\mu^T \left(S_0^{-1} + N\Sigma^{-1}\right)\mu - 2\left(N\bar{X}^T\Sigma^{-1} + m_0^T S_0^{-1}\right)\mu\right] \tag{29}$$

$$= -\frac{1}{2}\mu^T \left(N\Sigma^{-1} + S_0^{-1}\right)\mu + \left(N\bar{X}^T\Sigma^{-1} + m_0^T S_0^{-1}\right)\mu \tag{30}$$

We have now obtained quadratic form. Let

$$A = \left(N\Sigma^{-1} + S_0^{-1}\right), b^T = \left(N\bar{X}^T\Sigma^{-1} + m_0^T S_0^{-1}\right) \tag{31}$$

Then by completing the square, we can obtain the parameters μ_N, Σ_N for a Gaussian (see lecture slides)

$$\Sigma_N = A^{-1} = \left(N\Sigma^{-1} + S_0^{-1}\right)^{-1} \tag{32}$$

$$\mu_N = A^{-1}b = \left(N\Sigma^{-1} + S_0^{-1}\right)^{-1}\left(N\Sigma^{-1}\bar{X} + S_0^{-1} m_0\right) \tag{33}$$

$$\underline{\underline{\Rightarrow P(\mu|X) = N(\mu|\mu_N, \Sigma_N)}} \tag{34}$$

2b,c) Doing another monte carlo sampling, we can compare the prior and posterior distributions for μ.

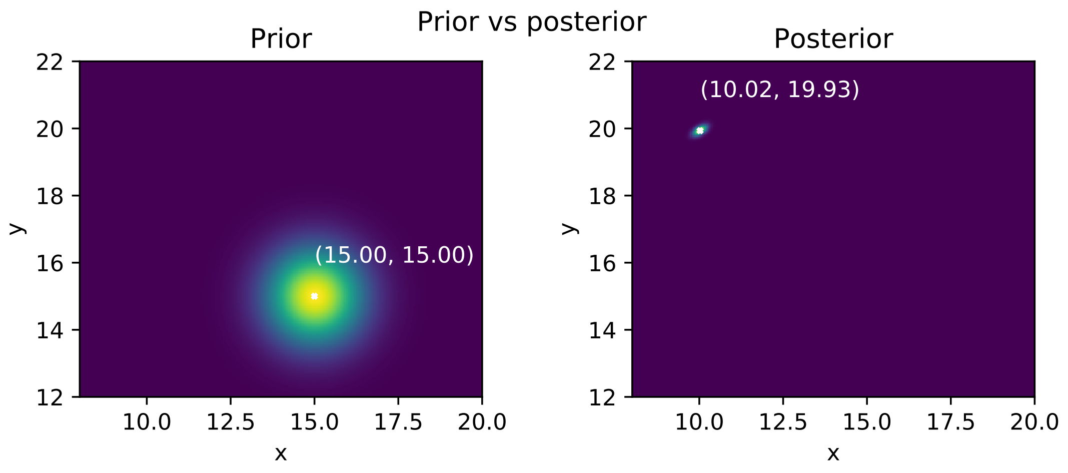

Figure 3: Heatmaps representing the densities on the $\mathbb{R}^2$ input space. The prior has variance 1 and is centered at (15,15). The posterior shrunk in size and huddled around $\approx (10, 20)$. The real μ is probably $(10, 20)$ since they are so nice numbers.

A Task 1 Code

```
import pandas as pd
import numpy as np
from scipy.stats import gamma, poisson, sem, bayes_mvs
from matplotlib import pyplot as plt

np.random.seed(69)

def HDI_from_MCMC(posterior_samples, credible_mass):
        # https://stackoverflow.com/questions/22284502/highest-posterior-
                                                      density-region-and-central-
                                                      credible-region
        # Computes highest density interval from a sample of
                                                      representative values,
        # estimated as the shortest credible interval
        # Takes Arguments posterior_samples (samples from posterior) and
                                                      credible mass (normally .95)
        sorted_points = sorted(posterior_samples)
        ciIdxInc = np.ceil(credible_mass * len(sorted_points)).astype('int
                                                      ')
        nCIs = len(sorted_points) - ciIdxInc

        ciWidth = [None]*nCIs
        for i in range(nCIs):
            ciWidth[i] = sorted_points[i + ciIdxInc] - sorted_points[i]

        HDImin = sorted_points[ciWidth.index(min(ciWidth))]
        HDImax = sorted_points[ciWidth.index(min(ciWidth))+ciIdxInc]
        return (HDImin, HDImax)
```

```
def maxima(ax, x, y):
    best_idx = np.argmax(y)
    bestx = x[best_idx]
    besty = y[best_idx]

    ax.scatter(bestx, besty, color='k', marker='x',zorder=10)
    ax.text(bestx, besty + 0.01, f'({bestx:.2f}, {besty:.2f})')

if __name__ == '__main__':
    def task1b(show=True):
        '''Compute posterior'''
        df = pd.read_csv('exercise5_1.txt', header=None)
        X = df.values.ravel()

        a0 = 2
        b0 = 2
        n = len(X)

        a = a0 + X.sum()
        b = n + b0

        xrange = np.linspace(0, 5, 10000)

        # Prior
        y_before = gamma.pdf(xrange, a=a0, scale=1/b0)

        # Posterior
        y_after = gamma.pdf(xrange, a=a, scale=1/b)

        # print(y)

        lambda_approx = xrange[np.argmax(y_after)]

        if show:
            fig, ax = plt.subplots(figsize=(7,4))
            fig.suptitle('Before and after observing data')
            ax.plot(xrange, y_before, label='Prior')
            ax.plot(xrange, y_after, label='Posterior')
            ax.set_xlabel(r'$\lambda$')
            ax.set_ylabel('density')

            ax.legend()
            maxima(ax, xrange, y_after)
            maxima(ax, xrange, y_before)
            fig.tight_layout(rect=[0, 0.03, 1, 0.95])
            # plt.savefig('images/1b.pdf')
            plt.show()

            print(lambda_approx)

        return lambda_approx

    def task1c(show=True):
        '''MC sanity check'''
        df = pd.read_csv('exercise5_1.txt', header=None)
        X = df.values.ravel()

        lambda_approx = task1b(False)
        y = poisson.rvs(lambda_approx, size=100)

        if show:
            histkwargs = dict(edgecolor='black', density=True)
            fig, (ax1, ax2) = plt.subplots(1,2, sharex=True, figsize=(7,3)
                                                )
            fig.suptitle('Comparing data to monte carlo sampling')
            ax1.set_title('Data', fontsize=10)
            ax1.hist(X, **histkwargs)
            ax2.set_title('Sampled data from posterior estimate', fontsize
                                                =10)
            ax2.hist(y, **histkwargs)

            plt.setp([ax1, ax2], xlabel='Poisson frequencies', ylabel='
                                                Normalized frequencies')

            fig.tight_layout(rect=[0, 0.03, 1, 0.90])
            # plt.savefig('images/1c.pdf')
            plt.show()

        return y

    def task1d(show=True):
        '''Credibility interval'''
        y = task1c(False)

        stringy = f'95% credibility interval is ()'
        def print_ci(y: np.ndarray, cm: float):
            left, right = HDI_from_MCMC(y, cm)
            print(f'{cm*100:.0f}% Credibility interval: ({left}, {right})'
                                                )

        print_ci(y, 0.5)
        print_ci(y, 0.95)


    # task1b()
    # task1c(True)
    # task1d()
```

B Task 2 Code

```
import pandas as pd
import numpy as np
from scipy.stats import gamma, poisson, sem, bayes_mvs,
                                        multivariate_normal
```

```
from matplotlib import pyplot as plt
np.random.seed(69)

def maxima2d(ax, x, z):
    best_idx = np.argmax(z)
    bestx = x[best_idx]
    bestz = z[best_idx]

    ax.scatter(bestx[0], bestx[1], color='w', marker='x',zorder=10, s=5)
    ax.text(bestx[0], bestx[1] + 1, f'({bestx[0]:.2f}, {bestx[1]:.2f})',
                                        color='w')

if __name__ == '__main__':
    def task2b(show=True):
        '''Calculate posterier and plot'''
        df = pd.read_csv('exercise5_2.txt', header=None)
        X = df.values

        sigma = np.array([[1, 0.8],[0.8, 0.2]])
        s0 = np.array([[1, 0],[0, 1]])
        m0 = np.array([15,15])

        N = len(X)

        precmat = np.linalg.inv(sigma)
        precmat_s = np.linalg.inv(s0)

        Xsum = X.sum(axis=0)

        A = N*precmat + precmat_s
        b = precmat@Xsum + precmat_s@m0

        sigma_n = np.linalg.inv(A)
        mu_n = sigma_n @ b

        sigma_n = sigma_n + np.eye(sigma_n.shape[0]) * 1e-2

        if show:
            resolution = 2000
            xrange = np.linspace(8,20,resolution)
            yrange = np.linspace(12,22,resolution)

            Xmesh, Ymesh = np.meshgrid(xrange, yrange)

            points = np.array([Xmesh.ravel(), Ymesh.ravel()]).T

            z_pri = multivariate_normal.pdf(points, m0, s0)
            z_post = multivariate_normal.pdf(points, mu_n, sigma_n)

            fig, (ax1, ax2) = plt.subplots(1,2, figsize=(7,3))
            fig.suptitle('Prior vs posterior')
            ax1.set_title('Prior')
            ax1.imshow(z_pri.reshape((resolution,resolution)), extent=[8,
                                                       20, 12, 22], origin='
                                                       lower')

            ax2.set_title('Posterior')
            ax2.imshow(z_post.reshape((resolution, resolution)), extent=[8
                                                       , 20, 12, 22], origin='
                                                       lower')

            plt.setp([ax1, ax2], xlabel='x', ylabel='y')

            fig.tight_layout(rect=[0, 0.03, 1, 0.95])

            maxima2d(ax2, points, z_post)
            maxima2d(ax1, points, z_pri)

            # plt.savefig('images/2c.pdf')
            plt.show()

    # task2b()
```