

CS-E5710 Bayesian Data Analysis

Assignment 4

October 6, 2019

1 Bioassay model and importance sampling

a) Report the mean and covariance of the bivariate normal distribution

```
1 mu_a = 0
2 sigma_a = 2
3 mu_b = 10
4 sigma_b = 10
5 corr = 0.5
6
7 mean = np.array([mu_a, mu_b])
8 print('mean:', mean)
9
10 cov = np.array([[sigma_a**2, corr * sigma_a * sigma_b],
11                 [corr * sigma_a * sigma_b, sigma_b**2]])
12 print('covariance:', cov)
```

```
1 mean: [ 0 10]
2 covariance: [[ 4. 10.]
3              [10. 100.]
```

b) Implement a function for computing the logarithm of the density of the prior distribution

```
1 def p_log_prior(alpha, beta):
2
3     prior = stats.multivariate_normal(mean, cov)
4     pos = np.dstack((alpha, beta))
5     log_prior = prior.logpdf(pos)
6
7     return log_prior
8
9 print('test prior:', p_log_prior(3, 9))
```

```
1 test prior: -6.296434970404113
```

c) Implement a function for computing the logarithm of the density of the posterior

$$\log(\text{posterior}) = \log(\text{prior} \times \text{likelihood}) = \log(\text{prior}) + \log(\text{likelihood}) \quad (1)$$

where $\log(\text{likelihood})$ is given in part c), $\log(\text{likelihood})$ is given in the BDA repository on GitHub. According to the Chapter3.7 Example: analysis of a bioassay experiment, we could implement $p_{\log\text{-posterior}}$ as following:

```

1 dose = np.array([-0.86, -0.30, -0.05, 0.72])
2 deaths = np.array([0, 1, 3, 5])
3 animals = np.array([5, 5, 5, 5])
4
5 def p_log_posterior(alpha, beta, x, y, n):
6     #logarithm of the density of the prior distribution
7     prior= stats.multivariate_normal(mean, cov)
8     pos = np.dstack((alpha, beta))
9     log_prior = prior.logpdf(pos)
10
11     #logarithm of the density of the likelihood distribution
12     alpha = np.expand_dims(alpha, axis=-1)
13     beta = np.expand_dims(beta, axis=-1)
14     t = alpha + beta*x
15     et = np.exp(t)
16     z = et/(1.+et)
17     log_likelihood = np.sum(y*np.log(z)+ (n-y)*np.log(1.0-z), axis=-1)
18     ##logarithm of the density of the posterior distribution
19     log_posterior = log_prior + log_likelihood
20     return log_posterior
21
22 print('test posterior',p_log_posterior(3, 9, dose, deaths, animals))

```

```

1 test posterior: -15.788012556775058

```

d) Plot the posterior density in a grid of points $\alpha \in [-4, 4]$, $\beta \in [-10, 30]$

```

1 alpha, beta = np.meshgrid(np.linspace(-4,4,100),np.linspace(-10,30,100))
2 posterior = np.exp(p_log_posterior(alpha, beta, dose, deaths, animals))
3
4 plt.contourf(alpha, beta,posterior, cmap=plt.cm.Greys)
5 plt.xlabel('alpha')
6 plt.ylabel('beta')
7 plt.title('Posterior Distribution')
8 plt.grid(linewidth=0.8, alpha=0.2)
9 plt.colorbar(plt.contourf(alpha, beta, posterior, cmap=plt.cm.Greys))
10 plt.savefig('./log_posterior.png')
11 plt.show()

```

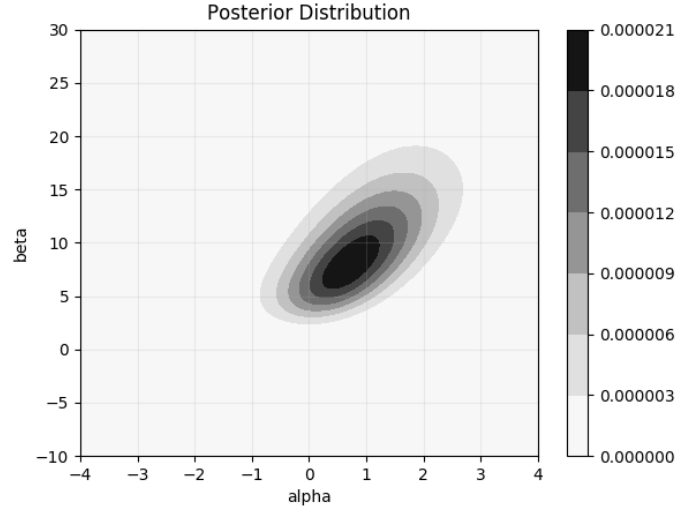


Figure 1: Heatmap of the posterior density

e) Sample draws of alpha and beta from the prior distribution

```
1 prior = stats.multivariate_normal(mean, cov)
2 samples = prior.rvs(10000)
```

Compute the importance ratios (importance weights) for each draw when the target distribution is the posterior distribution.

$$w = \theta_i^{y_i} (1 - \theta_i)^{n_i - y_i}$$

$$\theta_i = \frac{1}{1 + \exp(-\alpha + \beta x_i)} \quad (2)$$

Normalize the weights so that they sum to 1.

$$\tilde{\mathbf{w}}(\theta^s) = \frac{\mathbf{w}(\theta^s)}{\sum_{s'=1}^S \mathbf{w}(\theta^{s'})} \quad (3)$$

```
1 theta = 1 / (1 + np.exp(-(samples[:, 0, None] + samples[:, 1, None] * dose)))
2 weights = np.prod(
3     theta**deaths * (1 - theta)**(animals - deaths), axis=1)
4
5 #Normalize the weights
6 weights_norm = (weights) / np.sum(weights)
```

f) Compute the posterior mean using importance sampling and draws from e)

```

1 posterior_mean = sum(weights[ : , None] * samples) / sum(weights)
2 print('posterior mean of alpha : ', posterior_mean[0])
3 print('posterior mean of beta : ', posterior_mean[1])

```

```

1 posterior mean of alpha: 0.96466253
2 posterior mean of beta: 10.48528471

```

g) Compute the effective sample size

$$S_{eff} = \frac{1}{\sum_{s=1}^S (\tilde{\mathbf{w}}(\theta^s))^2} \quad (4)$$

$$\tilde{\mathbf{w}}(\theta^s) = \frac{\mathbf{w}(\theta^s)}{\sum_{s'=1}^S \mathbf{w}(\theta^{s'})}$$

```

1 s_eff = 1 / np.sum(weights_norm**2)
2 print('effective sample size: ', s_eff)

```

```

1 effective sample size: 2750.282727444427

```

h) Use importance resampling without replacement to obtain a posterior sample of size 1000 of alpha and beta

```

1 scode = np.random.choice(a=10000, size=1000, replace=False, p=weights_norm)
2 resamples = samples[scode]
3 print('mean of resampled alpha: ', np.mean(resamples[:, 0]))
4 print('mean of resampled beta: ', np.mean(resamples[:, 1]))

```

```

1 mean of resampled alpha: 0.9247278915535617
2 mean of resampled beta: 10.650671142369971

```

Plot a scatterplot of the obtained posterior sample

```

1 plt.xlim([-4, 4])
2 plt.ylim([-10, 30])
3 plt.xlabel('alpha')
4 plt.ylabel('beta')
5 plt.grid(linewidth=0.8, alpha=0.2)
6 plt.scatter(resamples[:, 0], resamples[:, 1], 8, color='grey')
7 plt.title('Posterior Samples')
8 plt.savefig('./posterior_samples.png')
9 plt.show()
10
11 plt.xlim([-4, 4])
12 plt.ylim([-10, 30])
13 plt.xlabel('alpha')
14 plt.ylabel('beta')
15 plt.grid(linewidth=0.8, alpha=0.2)
16 plt.contourf(alpha, beta, posterior, cmap=plt.cm.Greys)
17 plt.colorbar(plt.contourf(alpha, beta, posterior, cmap=plt.cm.Greys))

```

```

18 plt.scatter(resamples[:, 0], resamples[:, 1], 8, alpha=.15, color='grey')
19 plt.title('Contourf & Cscatter Comparision')
20 plt.savefig('./contourf_scatter.png')
21 plt.show()

```

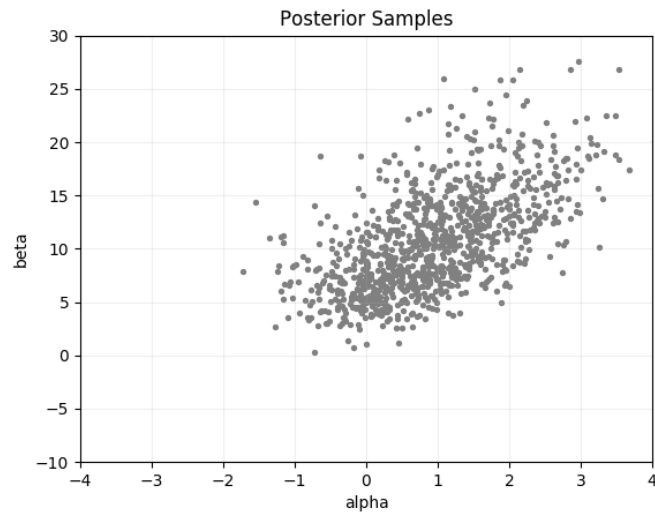


Figure 2: Scatterplot of the obtained posterior sample

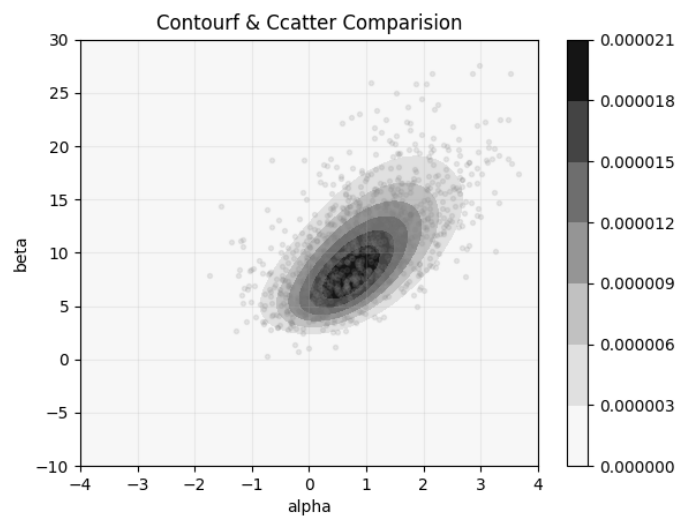


Figure 3: Comparison of Scatterplot and Heatmap

i) Using the posterior sample obtained via importance resampling, report an estimate for $p(\beta > 0|x, n, y)$, i.e., the probability that the drug is harmful.

```
1 beta_resample = resamples[:, 1]
2 alpha_resample = resamples[:, 0]
3 pos = beta_resample > 0
4 p_harmful = (beta_resample[pos].size / (beta_resample.size + 1))
5 print('Probability that the drug is harmful:', p_harmful)
```

```
1 Probability that the drug is harmful: 0.999000999000999
```

j) Using the posterior sample obtained via importance resampling, draw a histogram of the draws from the posterior distribution of the LD50 conditional on $\beta > 0$

LD50 is $x_i = -\frac{\alpha}{\beta}$

```
1 ld50 = - alpha_resample[pos] / beta_resample[pos]
2 y = np.arange(-0.4, 0.4, 0.01)
3 plt.hist(ld50, y, ec='white', color='grey')
4
5 plt.xlabel('LD50')
6 plt.title('Histogram')
7 plt.savefig('./histogram.png')
8 plt.show()
```

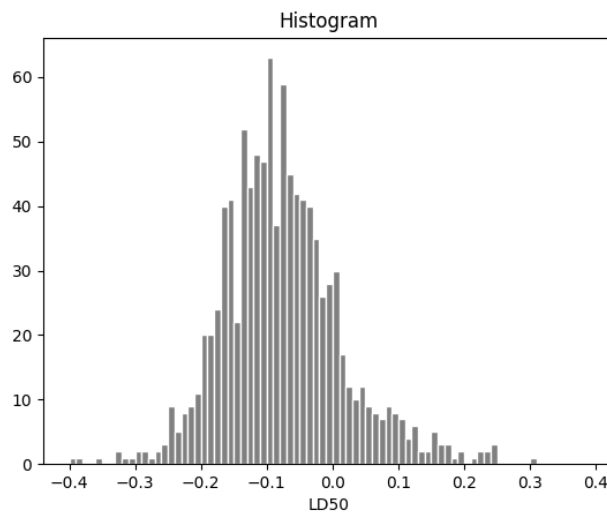


Figure 4: Histogram

A Code

```
1 import matplotlib.pyplot as plt
2 from mpl_toolkits.mplot3d import Axes3D
3 from scipy import stats
4 import numpy as np
5
6 def bioassaylp(a, b, x, y, n):
7     # last axis for the data points
8     a = np.expand_dims(a, axis=-1)
9     b = np.expand_dims(b, axis=-1)
10    # these help using chain rule in derivation
11    t = a + b*x
12    et = np.exp(t)
13    z = et/(1.+et)
14    # negative log posterior (error function to be minimized)
15    lp = np.sum(y*np.log(z) + (n-y)*np.log(1.0-z), axis=-1)
16    return lp
17
18 a)
19 '''
20 mu_a = 0
21 sigma_a = 2
22 mu_b = 10
23 sigma_b = 10
24 corr = 0.5
25
26 mean = np.array([mu_a, mu_b])
27 print('mean:', mean)
28
29 cov = np.array([[sigma_a**2, corr * sigma_a * sigma_b],
30 [corr * sigma_a * sigma_b, sigma_b**2]])
31 print('covariance:', cov)
32
33 alpha, beta = np.meshgrid(np.linspace(-4, 4, 100), np.linspace(-10, 30, 100))
34
35 '''
36 b)
37 '''
38
39 def p_log_prior(alpha, beta):
40
41     prior= stats.multivariate_normal(mean, cov)
42     pos = np.dstack((alpha, beta))
43     log_prior = prior.logpdf(pos)
44
45     return log_prior
46 #print('test', p_log_prior(3, 9))
47
48 dose = np.array([-0.86, -0.3, -0.05, 0.72])
49 deaths = np.array([0, 1, 3, 5])
50 animals = np.array([5, 5, 5, 5])
51
52 '''
53 c)
54 '''
```

```

55
56 def p_log_posterior(alpha, beta, x, y, n):
57     prior= stats.multivariate_normal(mean, cov)
58     pos = np.dstack((alpha, beta))
59     log_prior = prior.logpdf(pos)
60
61     alpha = np.expand_dims(alpha, axis=-1)
62     beta = np.expand_dims(beta, axis=-1)
63     t = alpha + beta*x
64     et = np.exp(t)
65     z = et/(1.+et)
66     log_likelihood = np.sum(y*np.log(z)+ (n-y)*np.log(1.0-z), axis=-1)
67
68     log_posterior = log_prior + log_likelihood
69
70     return log_posterior
71 #print('testposterior',p_log_posterior(3, 9, dose, deaths, animals))
72
73 '''
74 d)
75 '''
76 alpha,beta = np.meshgrid(np.linspace(-4,4,100), np.linspace(-10,30,100))
77 posterior = np.exp(p_log_posterior(alpha, beta, dose, deaths, animals))
78 plt.contourf(alpha, beta,posterior, cmap=plt.cm.Greys)
79 plt.xlabel('alpha')
80 plt.ylabel('beta')
81 plt.title('Posterior Distribution')
82 plt.grid(linewidth=0.8, alpha=0.2)
83 plt.colorbar(plt.contourf(alpha, beta, posterior, cmap=plt.cm.Greys))
84 plt.savefig('./log_posterior.png')
85 plt.show()
86
87 '''
88 e) 2. Sample draws of alpha and beta from the prior distribution.
89 '''
90 prior = stats.multivariate_normal(mean, cov)
91 samples = prior.rvs(10000)
92 #print('Shape of the samples from prior: ', samples.shape)
93
94 theta = 1 / (1+np.exp(-(samples[:,0,None] + samples[:,1,None] * dose)))
95 weights = np.prod(
96     theta**deaths * (1 - theta)**(animals - deaths), axis=1)
97
98 weights_norm = (weights) / np.sum(weights)
99 #print('Shape of the weights of the likelihood: ', weights.shape)
100
101 '''
102 f)
103 '''
104 posterior_mean = sum(weights[ :, None] * samples) / sum(weights)
105 print('posterior mean of alpha : ', posterior_mean[0])
106 print('posterior mean of beta : ', posterior_mean[1])
107
108 '''
109 g)
110 '''
111 s_eff = 1 / np.sum(weights_norm**2)

```



```

112 print('effective sample size: ', s_eff)
113
114 '''
115 h)
116 '''
117 scode = np.random.choice(a=10000, size=1000, replace=False, p=weights_norm)
118 resamples = samples[scode]
119 print('mean of resampled alpha: ', np.mean(resamples[:, 0]))
120 print('mean of resampled beta: ', np.mean(resamples[:, 1]))
121
122 plt.xlim([-4, 4])
123 plt.ylim([-10, 30])
124 plt.xlabel('alpha')
125 plt.ylabel('beta')
126 plt.grid(linewidth=0.8, alpha=0.2)
127 plt.scatter(resamples[:, 0], resamples[:, 1], 8, color='grey')
128 plt.title('Posterior Samples')
129 plt.savefig('./posterior_samples.png')
130 plt.show()
131
132 plt.xlim([-4, 4])
133 plt.ylim([-10, 30])
134 plt.xlabel('alpha')
135 plt.ylabel('beta')
136 plt.grid(linewidth=0.8, alpha=0.2)
137 plt.contourf(alpha, beta, posterior, cmap=plt.cm.Greys)
138 plt.colorbar(plt.contourf(alpha, beta, posterior, cmap=plt.cm.Greys))
139 plt.scatter(resamples[:, 0], resamples[:, 1], 8, alpha=.15, color='grey')
140 plt.title('Contourf & Scatter Comparision')
141 plt.savefig('./contourf_scatter.png')
142 plt.show()
143
144 '''
145 i)
146 '''
147 beta_resample = resamples[:, 1]
148 alpha_resample = resamples[:, 0]
149 pos = beta_resample > 0
150 p_harmful = (beta_resample[pos].size / (beta_resample.size + 1))
151 print('Probability that the drug is harmful:', p_harmful)
152
153 '''
154 j)
155 '''
156 ld50 = - alpha_resample[pos] / beta_resample[pos]
157 y = np.arange(-0.4, 0.4, 0.01)
158 plt.hist(ld50, y, ec='white', color='grey')
159 plt.xlabel('LD50')
160 plt.title('Histogram')
161 plt.savefig('./histogram.png')
162 plt.show()

```