HW3

November 4, 2022

[1]:
```python
import sympy
from sympy import solve
from sympy.abc import a, b
import numpy as np
import scipy
from scipy.stats import norm, multivariate_normal, invgamma, bernoulli
import matplotlib.pyplot as plt
from copy import deepcopy
```

1 Problem 1(a)

1.1 Constant definition and data simulation

[2]:
```python
# define constants and seed
loc_gt = 50
scale_gt = np.sqrt(10)
N = 100
mu_0 = 0
sigma2_0 = 100
alpha = 2
beta = 10
np.random.seed(100)
```

[3]:
```python
y = norm.rvs(loc_gt, scale_gt, N)
```

[4]:
```python
y
```

[4]:
```
array([44.46675573, 51.08365058, 53.64621936, 49.20172716, 53.1032088 ,
       51.62610275, 50.69943153, 46.61622588, 49.40076157, 50.80638537,
       48.5515915 , 51.37610778, 48.15451041, 52.58309725, 52.12732998,
       49.66982297, 48.31994393, 53.25630067, 48.61449351, 46.46356719,
       55.11966954, 54.8749836 , 49.20348823, 47.33598428, 50.58349933,
       52.96331411, 52.31162606, 54.30561852, 48.96834467, 50.17606302,
       50.70328931, 45.43614714, 47.608204  , 52.58185428, 52.3731147 ,
       48.55816922, 53.76191592, 44.65380018, 45.71068959, 46.10269987,
       48.2783322 , 47.88705544, 50.02313068, 48.06171753, 54.1101643 ,
       44.51947043, 46.89050044, 51.13053878, 44.89741675, 54.6508056 ,
```

```
46.24315849, 48.26154989, 47.02731302, 47.38184798, 50.34425651,
51.60583492, 47.27339772, 53.95117025, 49.74824714, 47.18642201,
47.21150865, 50.05894153, 50.75213073, 50.04284427, 44.82800192,
46.69791843, 51.93859916, 52.3280853 , 53.24741073, 45.47101563,
44.17765137, 51.15768843, 48.95082858, 47.82050138, 56.43399404,
48.25848812, 52.3731418 , 45.86692732, 51.83593409, 46.5071913 ,
52.18235571, 52.17213711, 45.04569902, 52.86177945, 52.46285267,
51.35419124, 50.34428346, 50.08944071, 48.16959203, 46.20700227,
44.60530608, 51.16739893, 55.93424623, 48.80812696, 55.79309055,
50.00954196, 49.75959269, 50.01251501, 49.41493401, 42.13493626])
```

1.2 Gradient of log posterior

[5]:
```
# let a = \mu, b = \sigma**2
d_logp_d_mu = -(N*a - np.sum(y))/b - (a-mu_0)/sigma2_0
d_logp_d_simga2 = -(N/2+alpha+1)/b + ((N*(a**2) - 2*a*np.sum(y)+np.sum(y**2))/2␣
↪+ beta)/(b**2)
```

1.3 Solve μ and σ^2 numerically

[6]:
```
MAP = sympy.solve([d_logp_d_mu, d_logp_d_simga2])
```

[7]:
```
mu_map = MAP[0][a]
sigma2_map = MAP[0][b]
```

[8]:
```
MAP[0]
```

[8]:
```
{a: 49.6256255571400, b: 9.06247322659448}
```

[9]:
```
d2_logp_d_mu_2 = -N/(sigma2_map) - 1/sigma2_0
d2_logp_d_mu_d_sigma2 = (N*mu_map - np.sum(y)) / (sigma2_map**2)
d2_logp_d_sigma2_2 = (N/2+alpha+1) / (sigma2_map**2) - (np.sum((mu_map - y)**2)␣
↪+ 2*beta) / (sigma2_map**3)
```

[10]:
```
cov_mat = -np.linalg.inv(np.array([[d2_logp_d_mu_2, d2_logp_d_mu_d_sigma2],␣
↪[d2_logp_d_mu_d_sigma2, d2_logp_d_sigma2_2]], dtype=np.float32))
```

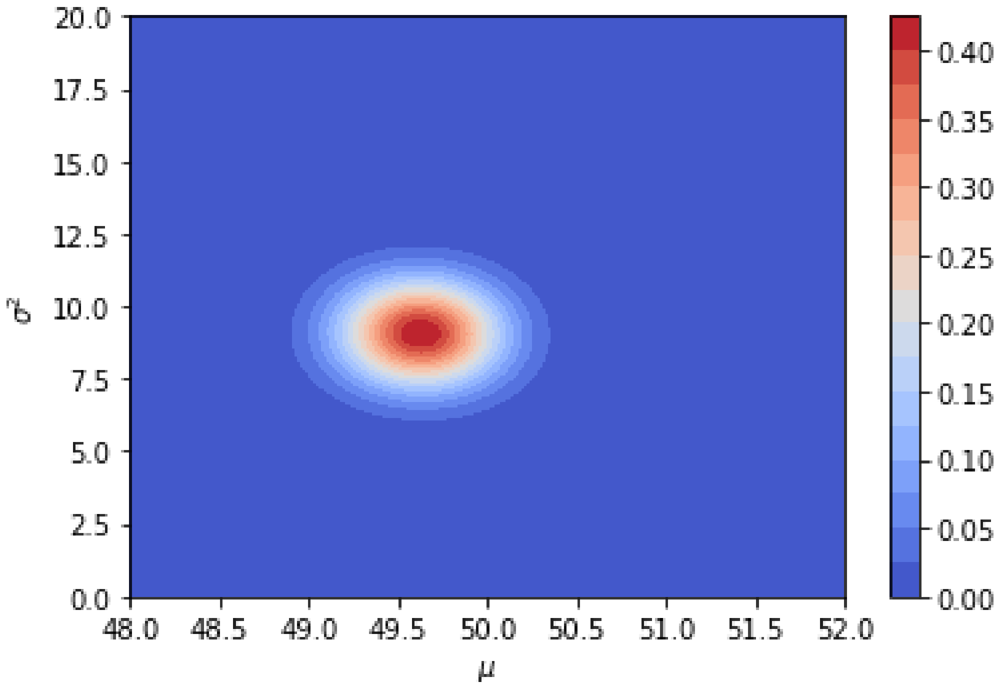
[11]:
```
cov_mat
```

[11]:
```
array([[ 0.09058078, -0.00768622],
       [-0.00768622,  1.550245  ]], dtype=float32)
```

[12]:
```
increment = 200
X    = np.linspace(48, 52, increment)
Y    = np.linspace(0, 20, increment)
X, Y = np.meshgrid(X, Y)
```

```
pos   = np.dstack((X, Y))
rv    = multivariate_normal([mu_map, sigma2_map], cov_mat)
Z     = rv.pdf(pos)

plt.contourf(X, Y, Z, 20, cmap='coolwarm')
plt.colorbar();
plt.xlabel('$\mu$')
plt.ylabel('$\sigma^2$')
plt.savefig("./p1a.pdf", format="pdf", bbox_inches="tight")
plt.show()
```



2 Problem 1(b)

2.1 Initialize m, v^2, b

[13]:
```
m = mu_0
v = np.sqrt(sigma2_0)
bb = beta # distinguish with symbol used in 1(a)
aa = 1 # distinguish with symbol used in 1(a)
```

[14]:
```
m, v, bb, mu_0, sigma2_0, alpha, beta
```

[14]: (0, 10.0, 10, 0, 100, 2, 10)

[15]:
```
total_steps = 1000
for l in range(total_steps):
    m = (np.sum(y) + bb/aa*(mu_0/sigma2_0)) / (N+bb/(aa*sigma2_0))
    v = np.sqrt((bb/aa) / (N+bb/(aa*sigma2_0)))
    bb = aa*(N*v**2 + np.sum((y-m)**2) + 2*beta) / (2*alpha + N)
```

[16]:
```
m, v, bb
```

[16]: (49.624316725225576, 0.3052502279129732, 9.326460345486584)

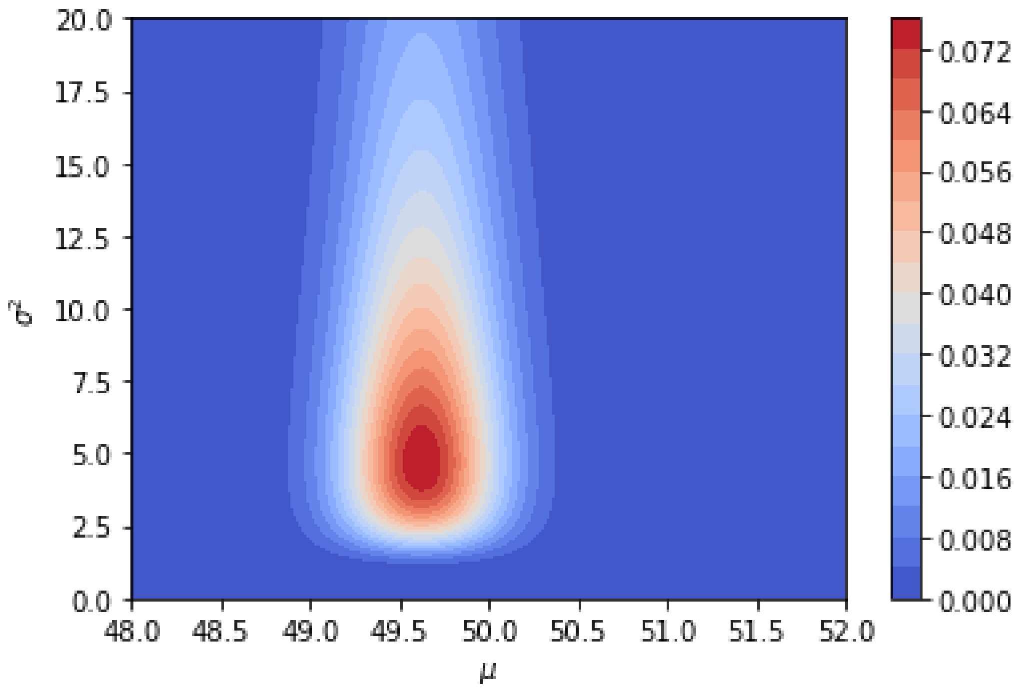
[17]:
```
q_u = norm(loc=m, scale=v)
q_sigma2 = invgamma(aa, scale=bb)
```

[18]:
```
increment = 200
X    = np.linspace(48, 52, increment)
Y    = np.linspace(0, 20, increment)
XX, YY = np.meshgrid(X, Y)
pos  = np.dstack((XX, YY))

MU = q_u.pdf(X)
SIMGA2 = q_sigma2.pdf(Y)

Z = np.outer(MU, SIMGA2).T

plt.contourf(X, Y, Z, 20, cmap='coolwarm')
plt.colorbar();
plt.xlabel('$\mu$')
plt.ylabel('$\sigma^2$')
plt.savefig("./p1b.pdf", format="pdf", bbox_inches="tight")
plt.show()
```



3 Problem 2(a)

3.1 Constant definition and data simulation

[19]:
```
N = 200
theta_0 = np.array([5, 5])
```

[20]:
```
w = bernoulli.rvs(0.5, size=N)
```

[21]:
```
y_from_mix1 = multivariate_normal.rvs(mean=theta_0, cov=np.eye(2), size=sum(w))
y_from_mix2 = multivariate_normal.rvs(mean=np.zeros((2, )), cov=10*np.eye(2),
↪size=N-sum(w))
```

[22]:
```
y = np.concatenate([y_from_mix1, y_from_mix2], axis=0)
```

[23]:
```
## Initialize
m = np.zeros((2, ))
v = 100
s = 1
```

[24]:
```
for n in range(N):
    z = (1-1/2)*multivariate_normal.pdf(y[n], mean=m, cov=(v+1)*np.eye(2)) + \
        1/2*multivariate_normal.pdf(y[n], mean=np.zeros((2, )), cov=10*np.
↪eye(2))
```

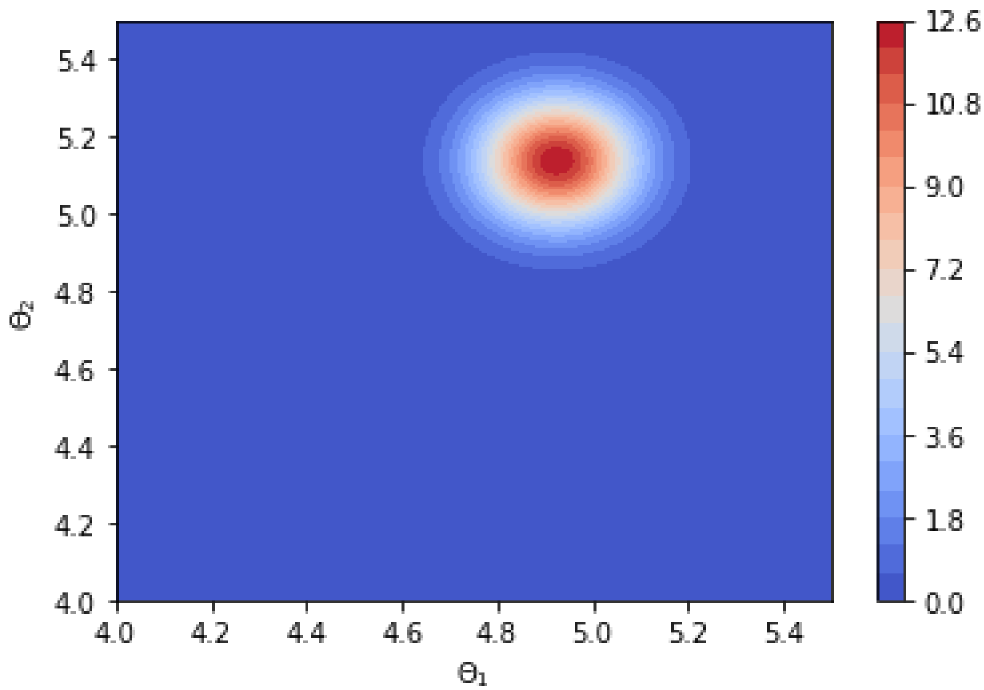
```
    s_new = s * z
    pi = 1 - 1/2 / z * multivariate_normal.pdf(y[n], mean=np.zeros((2, )),␣
↪cov=10*np.eye(2))
    m_new = m + v*pi*(y[n]-m)/(v+1)
    v_new = v - pi*(v**2)/(v+1) + pi*(1-pi)*(v**2*np.sum((y[n]-m)**2)) /␣
↪(2*(v+1)**2)
    m, v, s = m_new, v_new, s_new
```

[25]:
```
m_new, v_new
```

[25]: (array([4.92566649, 5.13535363]), 0.012727630711702492)

[26]:
```
increment = 200
X     = np.linspace(4, 5.5, increment)
Y     = np.linspace(4, 5.5, increment)
X, Y = np.meshgrid(X, Y)
pos   = np.dstack((X, Y))
rv    = multivariate_normal(m_new, v_new)
Z     = rv.pdf(pos)

plt.contourf(X, Y, Z, 20, cmap='coolwarm')
plt.colorbar();
plt.xlabel('$\Theta_1$')
plt.ylabel('$\Theta_2$')
plt.savefig("./p2a.pdf", format="pdf", bbox_inches="tight")
plt.show()
```



4 Problem 2(b)

[27]:
```
# initialize prior twems
v0 = 100
m0 = np.zeros((2, ))
s0 = (2*np.pi*v0)**(-2/2)
```

[28]:
```
# initialize data terms
v_data = np.ones((N, )) * np.inf
m_data = np.zeros((N, 2))
s_data = np.ones((N, ))
```

[29]:
```
# initialize target variables
m_new = m0
v_new = v0
```

[30]:
```
m_new
```

[30]:
```
array([0., 0.])
```

[31]:
```
delta = 1e-4
iter_num = 0
while True:
    # save old
    for n in range(N):
        # (a)
        m_new_copy = deepcopy(m_new)
        v_new_copy = deepcopy(v_new)
        v = 1/(1/v_new-1/v_data[n])
        if (v_data[n] < 0 or v_data[n] == np.inf) and iter_num > 1:
            v_data[n] = 1e10
            v_new = v
        m = m_new + v/v_data[n] * (m_new-m_data[n])
        # (b)
        # print(m, m_new, m_data[n], v, v_data[n])
        z = (1-1/2)*multivariate_normal.pdf(y[n], mean=m, cov=(v+1)*np.eye(2))␣
 ↪+ \
        1/2*multivariate_normal.pdf(y[n], mean=np.zeros((2, )), cov=10*np.
 ↪eye(2))
        pi = 1 - 1/2 / z * multivariate_normal.pdf(y[n], mean=np.zeros((2, )),␣
 ↪cov=10*np.eye(2))
        m_new = m + v*pi*(y[n]-m)/(v+1)
```

```python
        v_new = v - pi*(v**2)/(v+1) + pi*(1-pi)*(v**2*np.sum((y[n]-m)**2)) /␣
↪(2*(v+1)**2)
        # (c)
        v_data[n] = 1/(1/v_new - 1/v)
        m_data[n] = m + (v_data[n] + v)/v * (m_new-m)
        # s_data[n] = z / (2*np.pi*v_data[n]* multivariate_normal.
↪pdf(m_data[n], mean=m, cov=(v_data[n]+v)*np.eye(2)))
    # check convergence
        # print(np.sum(np.abs(m_new-m_new_copy)))
        if np.sum(np.abs(m_new-m_new_copy)) < delta:
            break
    if np.sum(np.abs(m_new-m_new_copy)) < delta:
            break
```

```
/var/folders/72/fk532lhj0vl3nqydc2n31h4c0000gn/T/ipykernel_64003/3967665427.py:2
2: RuntimeWarning: divide by zero encountered in double_scalars
  v_data[n] = 1/(1/v_new - 1/v)
/var/folders/72/fk532lhj0vl3nqydc2n31h4c0000gn/T/ipykernel_64003/3967665427.py:2
3: RuntimeWarning: invalid value encountered in multiply
  m_data[n] = m + (v_data[n] + v)/v * (m_new-m)
```

[32]:

```python
m_new, v_new
```

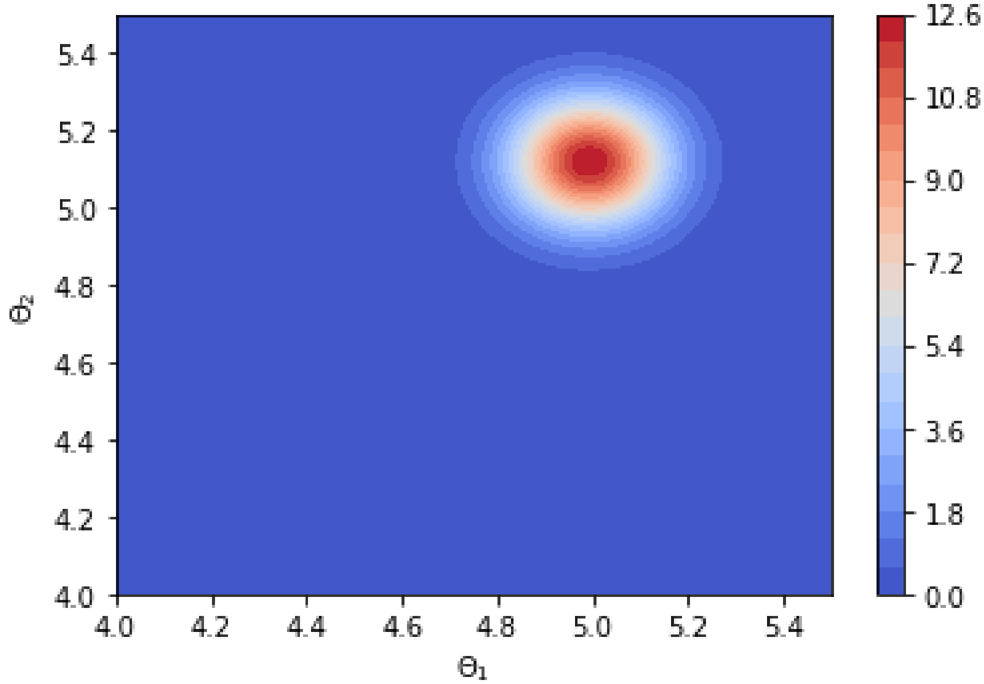
[32]:

```
(array([4.99337862, 5.11811095]), 0.012629772155207391)
```

[33]:

```python
increment = 200
X     = np.linspace(4, 5.5, increment)
Y     = np.linspace(4, 5.5, increment)
X, Y = np.meshgrid(X, Y)
pos   = np.dstack((X, Y))
rv    = multivariate_normal(m_new, v_new)
Z     = rv.pdf(pos)

plt.contourf(X, Y, Z, 20, cmap='coolwarm')
plt.colorbar();
plt.xlabel('$\Theta_1$')
plt.ylabel('$\Theta_2$')
plt.savefig("./p2b.pdf", format="pdf", bbox_inches="tight")
plt.show()
```


5.4
5.2
5.0
4.8
4.6
4.4
4.2
4.0
Θ2
4.0
4.2
4.4
4.6
4.8
5.0
5.2
5.4
Θ1
12.6
10.8
9.0
7.2
5.4
3.6
1.8
0.0