1. 《Machine Learning A Probabilistic Perspective》这本书的练习题: Exercise 4.1(20分)
2. Exercise 4.2(20分)
3. Exercise 4.7(20分)
4. Exercise 4.15(20分)
5. Exercise 4.22(20分)

1 Ex4.1 解：

$\rho(X,Y)=\frac{Cov(X,Y)}{\sqrt{var(X)var(Y)}}$,其中 $Cov(X,Y)=E[XY]-E[X]E[Y]=E[X^3]-E[X]E[Y]=\int_{-1}^{1}x^3p(x)dx-E[X^2]E[X]=0.\quad\Rightarrow\rho(X,Y)=0.$

2 Ex4.2 解：

a) $p(y)=p(wx)=p(sign(w)x)$，由于 $x\sim\mathcal{N}(0,1)$，那么 $p(x)=p(-x)$，所以有 $p(y)=p(sign(w)x)=p(x)$，故 $y\sim\mathcal{N}(0,1)$.

b) $E[XY]=E[E[XY|W]]=0.5E[XY|W=-1]+0.5E[XY|W=1]=0.5(E[-X^2]+E[X^2])=0.5E[X^2-X^2]=0$，所以 $Cov(X,Y)=E[XY]-E[X]E[Y]=0\quad\Rightarrow\quad\rho(X,Y)=0.$

3 Ex4.7 解：

a) 由于 x_1,x_2 都是二元正态分布，所以 $p(x_2|x_1)$ 也是正态分布.

$$p(x_2|x_1)=\mathcal{N}(x_2|\mu_{2|1},\Sigma_{2|1})$$
$$\mu_{2|1}=\mu_2+\Sigma_{21}\Sigma_{11}^{-1}(x-1-\mu_1)=\mu_2+\rho\frac{\sigma_2}{\sigma_1}(x-1-\mu_1)$$
$$\Sigma_{2|1}=\Sigma_{22}-\Sigma_{21}\Sigma_{11}^{-1}\Sigma_{12}=\sigma_2^2-\frac{\sigma_{21}\sigma_{12}}{\sigma_1^2}=\sigma_2^2(1-\rho^2)$$

由此，

$$p(x_2|x_1)=\mathcal{N}(x_2|\mu_2+\rho\frac{\sigma_2}{\sigma_1}(x_1-\mu_1),\sigma_2^2(1-\rho^2))$$

b) 当 $\sigma_1=\sigma_2=1$ 时，

$$p(x_2|x_1)=\mathcal{N}(x_2|\mu_2+\rho(x_1-\mu_1),1-\rho^2)$$

4 Ex4.15 解：

a) 要证

$$C_{n+1}=\frac{n-1}{n}C_n+\frac{1}{n+1}(x_{n+1}-m_n)(x_{n+1}-m_n)^T$$

即证

$$nC_{n+1}-(n-1)C_n=\frac{n}{n+1}(x_{n+1}-m_n)(x_{n+1}-m_n)^T$$

由于

$$C_n=\frac{1}{n-1}\sum_{i=1}^{n}(x_i-m_n)(x_i-m_n)^T$$

那么

$$nC_{n+1}=\sum_{j=1}^{n+1}(x_i-m_{n+1})(x_i-m_{n+1})^T=\sum_{j=1}^{n+1}x_jx_j^T-(n+1)m_{n+1}m_{n+1}^T$$
$$(n-1)C_n=\sum_{j=1}^{n}(x_i-m_n)(x_i-m_n)^T=\sum_{j=1}^{n}x_jx_j^T-nm_nm_n^T$$

$\Rightarrow$

$$nC_{n+1}-(n-1)C_n=x_{n+1}x_{n+1}^T-(n+1)m_{n+1}m_{n+1}^T+nm_nm_n^T$$

又 $m_{n+1}=\frac{x_{n+1}+nm_n}{n+1}$，那么

$$\begin{aligned}nC_{n+1}-(n-1)C_n&=x_{n+1}x_{n+1}^T-\frac{(n+1)}{(n+1)^2}(nm_n+x_{n+1})(nm_n+x_{n+1})^T+nm_nm_n^T\\&=x_{n+1}x_{n+1}^T-\frac{1}{n+1}(n^2m_nm_n^T+nm_nx_{n+1}^T+nx_{n+1}m_n^T+x_{n+1}x_{n+1}^T)+nm_nm_n^T\\&=\frac{n}{n+1}x_{n+1}x_{n+1}^T-\frac{n}{n+1}(m_nx_{n+1}^T+x_{n+1}m_n^T)+\frac{n}{n+1}m_nm_n^T\\&=\frac{n}{n+1}(x_{n+1}-m_n)(x_{n+1}-m_n)^T\end{aligned}$$

则有 $C_{n+1} = \frac{n-1}{n}C_n + \frac{1}{n+1}(x_{n+1}-m_n)(x_{n+1}-m_n)^T$ 成立.

b) 假设第 n 步的 m_n 已经得到，那么 $m_{n+1} = \frac{m_n + x_{n+1}}{n+1} : O(1)$，$(x_{n+1}-m_n)(x_{n+1}-m_n)^T : O(d^2)$，其他计算 $O(1)$，总共的时间复杂度为 $O(d^2)$.

c)

$$\begin{aligned}
C_{n+1}^{-1} &= \left[\frac{n-1}{n}C_n + \frac{1}{n+1}(x_{n+1}-m_n)(x_{n+1}-m_n)^T\right]^{-1} \\
&= \frac{n}{n-1}C_n^{-1} - \frac{\frac{n}{n-1}C_n^{-1}\frac{1}{n+1}(x_{n+1}-m_n)(x_{n+1}-m_n)^T\frac{n}{n-1}C_n^{-1}}{1+\frac{1}{n+1}(x_{n+1}-m_n)^T\frac{n}{n-1}C_n^{-1}(x_{n+1}-m_n)} \\
&= \frac{n}{n-1}\left[C_n^{-1} - \frac{nC_n^{-1}(x_{n+1}-m_n)(x_{n+1}-m_n)^T C_n^{-1}}{n^2-1+n(x_{n+1}-m_n)^T C_n^{-1}(x_{n+1}-m_n)}\right] \\
&= \frac{n}{n-1}\left[C_n^{-1} - \frac{C_n^{-1}(x_{n+1}-m_n)(x_{n+1}-m_n)^T C_n^{-1}}{\frac{n^2-1}{n}+(x_{n+1}-m_n)^T C_n^{-1}(x_{n+1}-m_n)}\right]
\end{aligned}$$

d) 对于 $C_n^{-1}(x_{n+1}-m_n)(x_{n+1}-m_n)^T C_n^{-1}$，

$$A = C_n^{-1}(x_{n+1}-m_n):\ O(d^2)$$

$$B = (x_{n+1}-m_n)^T C_n^{-1}:\ O(d^2)$$

$$C_n^{-1}(x_{n+1}-m_n)(x_{n+1}-m_n)^T C_n^{-1} = AB:\ O(d^2)$$

故时间复杂度为 $O(d^2)$.

5 Ex4.22 解：编写代码如下：

```python
from matplotlib import pyplot as plt
import numpy as np
from scipy.stats import multivariate_normal

plt.rcParams['figure.figsize'] = 10, 8

# Declare the parameters of the distributions and the data points
mu_1 = np.array([0, 0])
mu_2 = np.array([1, 1])
mu_3 = np.array([-1, 1])

Sigma_1 = np.array([[0.7, 0], [0, 0.7]])
Sigma_2 = np.array([[0.8, 0.2], [0.2, 0.8]])
Sigma_3 = np.array([[0.8, 0.2], [0.2, 0.8]])

x_1 = np.array([-0.5, 0.5])
x_2 = np.array([0.5, 0.5])

x = np.asarray([mu_1[0], mu_2[0], mu_3[0], x_1[0], x_2[0]])
y = np.asarray([mu_1[1], mu_2[1], mu_3[1], x_1[1], x_2[1]])

# Initial visualization
fig, ax = plt.subplots()
text = ['$\mu_1$', '$\mu_2$', '$\mu_3$', '$x_1$', '$x_2$']
text_loc_x = x + 0.05
text_loc_y = y + 0.05
ax.scatter(
x,
y,
color=['k', 'b', 'g', 'r', 'c'],
s=100)
for i, txt in enumerate(text):
ax.annotate(txt, (text_loc_x[i], text_loc_y[i]), size=30)
ax.grid()
ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_xlim([-1.2, 1.2])
ax.set_ylim([-1.2, 1.2])
ax.set_title('Scatter plot')

# Classification
score_x1 = [
multivariate_normal.pdf(x_1, mu_1, Sigma_1),
multivariate_normal.pdf(x_1, mu_2, Sigma_2),
multivariate_normal.pdf(x_1, mu_3, Sigma_3)
]
class_x1 = score_x1.index(max(score_x1)) + 1
print('Class x_1 = {}'.format(class_x1))

score_x2 = [
multivariate_normal.pdf(x_2, mu_1, Sigma_1),
multivariate_normal.pdf(x_2, mu_2, Sigma_2),
multivariate_normal.pdf(x_2, mu_3, Sigma_3)
]
class_x2 = score_x2.index(max(score_x2)) + 1
print('Class x_2 = {}'.format(class_x2))

# Visualization of the result
fig, ax = plt.subplots()

text = ['$\mu_1$', '$\mu_2$', '$\mu_3$', '$x_1$', '$x_2$']
text_loc_x = x + 0.05
text_loc_y = y + 0.05
ax.scatter(
x,
y,
color=['k', 'b', 'g', 'r', 'c'],
s=100)
for i, txt in enumerate(text):
ax.annotate(txt, (text_loc_x[i], text_loc_y[i]), size=30)
ax.grid()
ax.set_xlabel('x1')
ax.set_ylabel('x2')
ax.set_xlim([-2.0, 2.0])
ax.set_ylim([-1.2, 2.0])
ax.set_title('Scatter plot')

x_pos, y_pos = np.mgrid[-2.0:2.0:.01, -2.0:2.0:.01]
pos = np.empty(x_pos.shape + (2,))
pos[:, :, 0] = x_pos
pos[:, :, 1] = y_pos
rv1 = multivariate_normal(mu_1, Sigma_1)
rv2 = multivariate_normal(mu_2, Sigma_2)
rv3 = multivariate_normal(mu_3, Sigma_3)
contour_levels= [0.15, 0.2]
cs1 = ax.contour(
x_pos, y_pos, rv1.pdf(pos), colors='r', alpha=1, levels=contour_levels)
cs2 = ax.contour(
x_pos, y_pos, rv2.pdf(pos), colors='y', alpha=1, levels=contour_levels)
cs3 = ax.contour(
x_pos, y_pos, rv3.pdf(pos), colors='b', alpha=1, levels=contour_levels)
ax.clabel(cs1, fontsize=20)
ax.clabel(cs2, fontsize=20)
ax.clabel(cs3, fontsize=20)
```

得到分类结果为：$Class\ x_1 = 1$，$Class\ x_2 = 2$. 如下图所示：

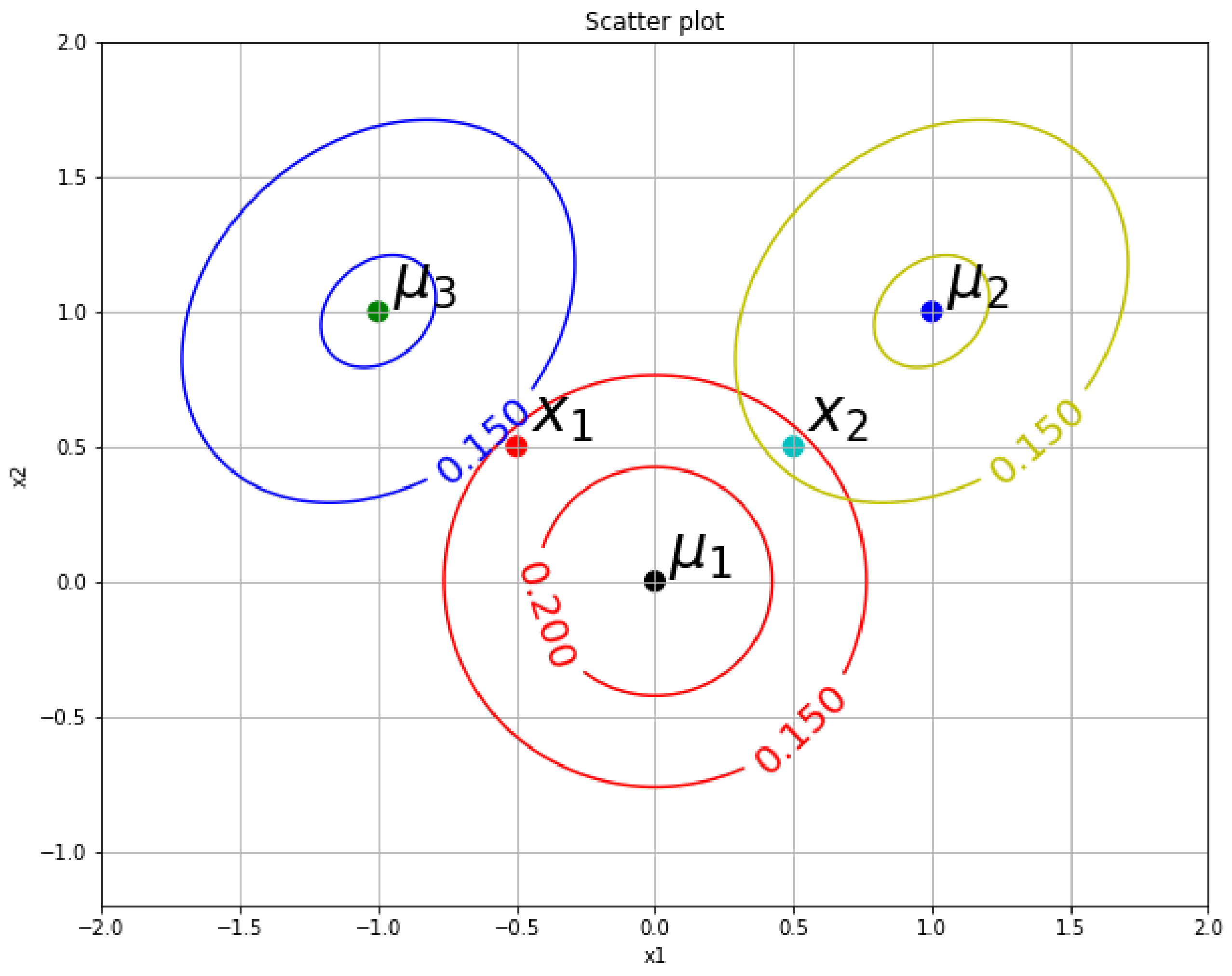

图 1: 分类结果