第二章作业

谈昊　2020E8013282037

Question

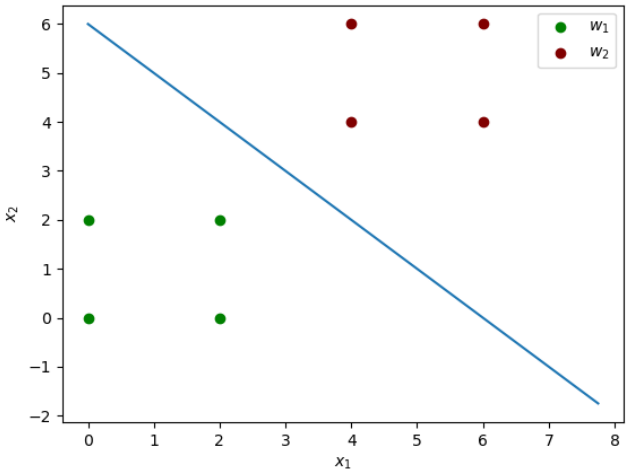
- 设以下模式类别具有正态概率密度函数：
  - $\omega_1: (0,0)^T, (2,0)^T, (2,2)^T, (0,2)^T$
  - $\omega_2: (4,4)^T, (6,4)^T, (6,6)^T, (4,6)^T$

  (1) 设 $P(\omega_1) = P(\omega_2) = 1/2$，求这两类模式之间的贝叶斯判别界面的方程式。

  (2) 绘出判别界面。
- 编写两类正态分布模式的贝叶斯分类程序。（可选例题或上述作业题为分类模式）

Answer

- 判别方程式为：$-4.0 * x_1 - 4.0 * x_2 + 24.0 = 0$
- 判别界面：



- 代码如下：

```
import numpy as np


class byers():
    def __init__(self):
        self.w_1 = np.array([[0, 0], [2, 0], [2, 2], [0, 2]])
        self.w_2 = np.array([[4, 4], [6, 4], [6, 6], [4, 6]])
        self.mean_1 = self._get_mean(self.w_1)
        self.mean_2 = self._get_mean(self.w_2)
        self.cov = self._get_cov(self.w_1, self.mean_1)

        self.w, self.b = self._get_line()
        self._plot()

    def _get_mean(self, x):
        return np.mean(x, axis=0)

    def _get_cov(self, x, m):
        return np.matmul((x - m).T, x - m) / x.shape[0]

    def _matmul(self, x, y, z):
        return np.matmul(np.matmul(x, y), z)

    def _get_line(self):
        cov_ = np.linalg.inv(self.cov)
        b = 0.5 * (self._matmul(self.mean_2.T, cov_, self.mean_2) - self._matmul(self.mean_1.T, cov_, self.mean_1))
        w = np.matmul((self.mean_1 - self.mean_2).T, cov_)
        line = ''
```

```
29                      for i, item in enumerate(w.data):
30                          flag = '+' if item > 0 else ''
31                          line += flag + str(item) + '*x_' + str
                                (i + 1)
32                      flag = '+' if b > 0 else ''
33                      line += flag + str(b)
34                      print(line)
35                      return w, b
36
37                  def _plot(self):
38                      import matplotlib.pyplot as plt
39                      x_1 = [x[0] for x in self.w_1.data.obj]
40                      y_1 = [x[1] for x in self.w_1.data.obj]
41
42                      x_2 = [x[0] for x in self.w_2.data.obj]
43                      y_2 = [x[1] for x in self.w_2.data.obj]
44
45                      fig = plt.figure()
46
47
48                      X = np.arange(0, 8, 0.25)
49
50                      a1 = self.w.data[0]
51                      a2 = self.w.data[1]
52                      Y = (-a1 * X - self.b) / a2
53
54
55                      plt.plot(X, Y)
56                      plt.xlabel(r"$x_1$")
57                      plt.ylabel(r"$x_2$")
58                      plt.scatter(x_1, y_1, label=r'$w_1$',
                            color=(0., 0.5, 0.))
59                      plt.scatter(x_2, y_2, label=r'$w_2$',
                            color=(0.5, 0., 0.))
60                      plt.legend()
61                      plt.show()
```

```
62
63
64              if __name__ == '__main__':
65                  byr = byers()
```