模式识别作业三

上机练习

数据科学与计算机学院　17大数据与人工智能

17341015　陈鸿峥

问题 1 (§2.5 Q1). 下面的几道题可能会用到如下的程序：

(a) 写一个程序产生服从d维正态分布$N(\boldsymbol{\mu}, \Sigma)$的随机样本

(b) 写一个程序计算一给定正态分布及先验概率$P(\omega_i)$的判别函数（式*(49)*中所给的形式）。

(c) 写一个程序计算任意两个点间的欧式距离。

(d) 在给定协方差矩阵Σ的情况条件下，写一个程序计算任意一点$\mathbf{x}$到均值$\boldsymbol{\mu}$间的*Mahalanobis*距离。

一、 实验过程及代码

原理分析如下

(a) d维多元正态密度的形式如下

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left[-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right]$$

对Σ进行乔列斯基分解，得到$\Sigma = LL^{\mathrm{T}}$，故随机样本为$L\mathbf{v}+\boldsymbol{\mu}$，其中$\mathbf{v}$为$d$维的随机数，用以表示偏移量。

(b) 正态分布的判别函数为

$$g(\mathbf{x}) = -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}) - \frac{d}{2}\ln 2\pi - \frac{1}{2}\ln|\Sigma| + \ln P(\omega)$$

(c) 即求两个点$\mathbf{v}_1$和$\mathbf{v}_2$的L2范数。

(d) 由Mahalanbois距离的定义

$$d_M^2 = (\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})$$

程序如下，采用Python进行编写，并且利用numpy包进行矩阵运算。有些是numpy中有现成函数，但我也写了两个版本互相进行验证（一种基于基本的矩阵运算，一种直接使用numpy封装好的包）。

```
import numpy as np

def normal_distribution(mu,sigma,size=10):
    """
```

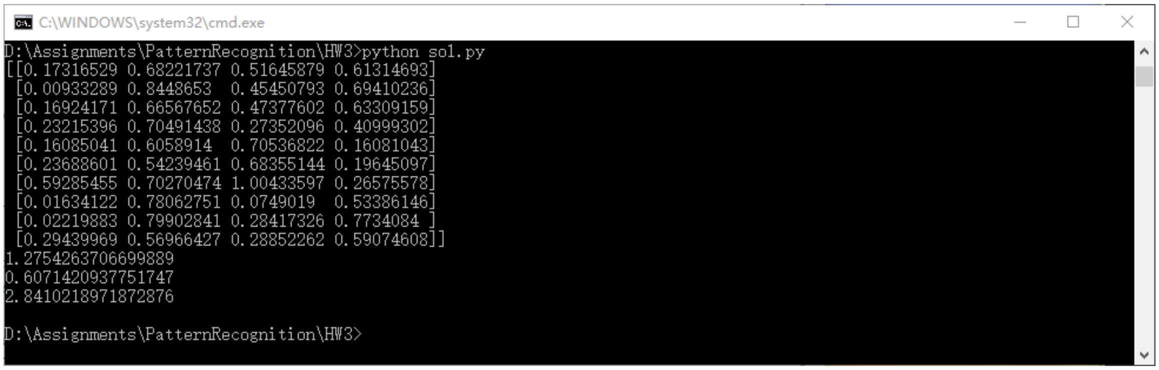
```
    Generate d-dimensional normal distribution N(mu,Sigma)
    mu: d-dim vector
    Sigma: d*d-dim covariance matrix
    n: number of generated points
    """
    # return np.random.multivariate_normal(mu,sigma,size)
    return np.dot(np.random.randn(size,mu.size), np.linalg.cholesky(sigma)) + mu

def discriminant(x,mu,sigma,p_omega):
    """
    g(x) = ln p(x|omega) + ln P(omega)
    """
    d = mu.size
    return -1/2 * Manhalanobis(x,mu,sigma) - d/2 * np.log(np.pi) - 1/2 * np.log(np
        ↪ .abs(np.linalg.det(sigma))) + np.log(p_omega)

def L2(p1,p2):
    """
    Euclidean distance (L2 distance)
    """
    # return np.linalg.norm(p1-p2)
    return np.sqrt(np.sum(np.power(p1-p2,2)))

def Manhalanobis(x,mu,sigma):
    """
    Given covariance matrix Sigma, compute the Manhalanobis distance
    from point x to mean mu
    """
    return (x-mu).T.dot(np.linalg.inv(sigma)).dot((x-mu))
```

并且编写了一组测试样例进行检验

```
m = np.random.rand(4,10)
mu = np.mean(m,axis=1)
sig = np.cov(m)
v1 = m.T[1]
v2 = m.T[2]
print(normal_distribution(mu,sig))
print(discriminant(v1,mu,sig,1/2))
print(L2(v1,v2))
print(Manhalanobis(v1,mu,sig))
```

二、 实验结果与分析

结果如下图所示。



可以看到程序正常生成了10个4维的随机数，同时以v1和v2作为输入样例，计算了判别函数、欧式距离和Manhalanobis距离，结果均正常显示并且验证正确。

三、 实验总结

本次实验主要将贝叶斯决策论的一些基本函数给实现了，由于基于Python的矩阵运算库numpy，因此整个实现过程还是相对比较简单的。