模式识别 第一次作业 2021.9.17

姓名：樊宇

学号：2021E8018782022

Question 1

(1)

贝叶斯风险最小决策：

将具有x特征的样本决策为第i类的条件风险为：

$$R(\alpha_i|\boldsymbol{x}) = \sum_{j=1}^{c} \lambda(\alpha_i|w_j)P(w_j|\boldsymbol{x})$$

因此，贝叶斯最小风险决策为：

$$\arg\min_i R(\alpha_i|\boldsymbol{x})$$

即将具有特征$\boldsymbol{x}$的样本决策为条件风险最小的那一类。

最小错误率决策：

令：

$$\lambda(\alpha_i|w_j) = \begin{cases} 0, & i = j \\ 1, & i \neq j \end{cases} \qquad i, j = 1, 2, \ldots, c$$

则将$\boldsymbol{x}$为第j类的决策风险为：

$$\begin{aligned} R(\alpha_i|\boldsymbol{x}) &= \sum_{j=1}^{c} \lambda(\alpha_i|w_j)P(w_j|\boldsymbol{x}) \\ &= \sum_{j=1}^{c} P(w_j|\boldsymbol{x}) - P(w_i|\boldsymbol{x}) \\ &= 1 - P(w_i|\boldsymbol{x}) \end{aligned}$$

因此，最小错误率决策为：

$$\arg\max_i P(w_i|\boldsymbol{x})$$

又因为：

$$P(w_i|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|w_i)P(w_i)}{p(\boldsymbol{x})}$$

去掉无关项：

$$\arg\max_i P(\boldsymbol{x}|w_i)P(w_i)$$

(2)

依题中条件为错误决策的条件风险为λ_r，拒识的条件风险为λ_s

将$\boldsymbol{x}$决策为第i类的条件风险为：

$$
\begin{aligned}
R(\alpha_i|\boldsymbol{x}) &= \sum_{j=1}^{c}\lambda(\alpha_i|w_j)P(w_j|\boldsymbol{x}) \\
&= \sum_{i\neq j}\lambda_r P(w_j|\boldsymbol{x}) \\
&= \lambda_r(1-P(w_i|\boldsymbol{x}))
\end{aligned}
$$

因此，引入拒识的最小损失决策的决策规则为：

$$
\arg\min_i R(\alpha_i|\boldsymbol{x}) = \begin{cases} \arg\max_i P(w_i|\boldsymbol{x}), & \max_i P(w_i|\boldsymbol{x}) > 1-\frac{\lambda_s}{\lambda_r} \\ reject, & otherwise \end{cases}
$$

Question 2

(1)

假设两类条件概率密度中有$\mu_2 > \mu_1$：

$$
\begin{aligned}
P_e &= \int_{R_2} p(\boldsymbol{x}|w_1)p(w_1)d\boldsymbol{x} + \int_{R_1} p(\boldsymbol{x}|w_2)p(w_2)d\boldsymbol{x} \\
&= \int_{\frac{\mu_1+\mu_2}{2}}^{+\infty} p(\boldsymbol{x}|w_1)p(w_1)d\boldsymbol{x} + \int_{-\infty}^{\frac{\mu_1+\mu_2}{2}} p(\boldsymbol{x}|w_2)p(w_2)d\boldsymbol{x}
\end{aligned} \tag{*}
$$

对于*式中的第一项：

$$
\int_{\frac{\mu_1+\mu_2}{2}}^{+\infty} p(\boldsymbol{x}|w_1)p(w_1)d\boldsymbol{x} = \frac{1}{2}\int_{\frac{\mu_1+\mu_2}{2}}^{+\infty} \frac{1}{\sqrt{2\pi}\sigma} exp[-\frac{(x-\mu_1)^2}{2\sigma^2}]dx
$$

令$\frac{x-\mu_1}{\sigma}=\mu$，则可写为：

$$
\frac{1}{2\sqrt{2\pi}}\int_{\frac{\mu_2-\mu_1}{2\sigma}}^{+\infty} exp(-\frac{\mu^2}{2})d\mu
$$

对于*式中的第二项：

$$
\begin{aligned}
\int_{-\infty}^{\frac{\mu_1+\mu_2}{2}} p(\boldsymbol{x}|w_2)p(w_2)d\boldsymbol{x} &= \frac{1}{2}\int_{\frac{3\mu_2-\mu_1}{2}}^{+\infty} p(\boldsymbol{x}|w_2)p(w_2)d\boldsymbol{x} \\
&= \frac{1}{2}\int_{\frac{3\mu_2-\mu_1}{2}}^{+\infty} \frac{1}{\sqrt{2\pi}\sigma} exp[-\frac{(x-\mu_2)^2}{2\sigma^2}]dx
\end{aligned}
$$

令$\frac{x-\mu_2}{\sigma}=\mu$，则可写为：

$$
\frac{1}{2\sqrt{2\pi}}\int_{\frac{\mu_2-\mu_1}{2\sigma}}^{+\infty} exp(-\frac{\mu^2}{2})d\mu
$$

综合上述，当$\mu_2 > \mu_1$时：

$$
P_e = \frac{1}{\sqrt{2\pi}}\int_{\frac{\mu_2-\mu_1}{2\sigma}}^{+\infty} exp(-\frac{\mu^2}{2})d\mu
$$

同理，当$\mu_2 < \mu_1$时：

$$P_e = \frac{1}{\sqrt{2\pi}} \int_{\frac{\mu_1-\mu_2}{2\sigma}}^{+\infty} exp(-\frac{\mu^2}{2}) d\mu$$

综合上述两种情况有：

$$P_e = \frac{1}{\sqrt{2\pi}} \int_{\frac{|\mu_1-\mu_2|}{2\sigma}}^{+\infty} exp(-\frac{\mu^2}{2}) d\mu$$

得证

(2)

因为：

$$exp(-\frac{\mu^2}{2}) > 0$$

所以：

$$P_e = \frac{1}{\sqrt{2\pi}} \int_{\frac{|\mu_1-\mu_2|}{2\sigma}}^{+\infty} exp(-\frac{\mu^2}{2}) d\mu >= 0$$

又因为：

$$\lim_{a \to +\infty} \frac{1}{\sqrt{2\pi}a} e^{-\frac{a^2}{2}} = 0$$

由夹逼定理得：

$$\lim_{a \to +\infty} P_e = 0$$

得证

Question 3

(1)

$$p(x|w_i) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} exp(-\frac{1}{2}(x-\mu_i)^T \Sigma_i^{-1}(x-\mu_i))$$

(2)

令判别函数：

$$\begin{aligned} g_i(x) &= \ln p(x|w_i) + \ln P(w_i) \\ &= -1/2(x-\mu_i)^T \Sigma_i (x-\mu_i) - d/2 \ln(2\pi) - 1/2 \ln|\Sigma_i| + \ln P(w_i) \end{aligned}$$

(a) $\Sigma_i = arbitrary$

去掉无关项后：

$$g_i(x) = x^T W_i x + w_i^T x + w_{i0}$$

其中：

$$
\begin{aligned}
W_i &= -\frac{1}{2}\Sigma_i^{-1} \\
w_i &= \Sigma_i^{-1}\mu_i \\
w_{i0} &= -\frac{1}{2}\mu_i^T\Sigma_i^{-1}\mu_i - \frac{1}{2}\ln|\Sigma_i| + \ln P(w_i)
\end{aligned}
$$

(b) $\Sigma_i = \Sigma$

$$g_i(x) = w_i^T x + w_{i0}$$

其中：

$$
\begin{aligned}
w_i &= \Sigma^{-1}\mu_i \\
w_{i0} &= -\frac{1}{2}\mu_i^T\Sigma_i^{-1}\mu_i + \ln P(w_i)
\end{aligned}
$$

(3)

(1) 主成分分析（PCA）

将随即适量投影到低维子空间，使子空间投影的重建误差最小

选择特征值最大的m（m<d）个特征向量作为子空间的基

(2) 正则化判别分析

$$
\begin{aligned}
\Sigma_i(\alpha) &= \frac{(1-\alpha)n_i\Sigma_i + \alpha n\Sigma}{(1-\alpha)n + \alpha n} \\
\Sigma(\beta) &= (1-\beta)\Sigma + \beta I
\end{aligned}
$$

Question 4

ZCA白化的作用：降低输入数据的冗余性，使输入特征之间的相关性较低，特征具有相同的方差

推导过程：

设原始数据为x，其协方差矩阵为C

需要找到P，使得$y = Px$，且y的协方差矩阵为D为对角阵：

$$
\begin{aligned}
D &= \frac{1}{m}yy^T \\
&= \frac{1}{m}(Px)(Px)^T \\
&= P(\frac{1}{m}xx^T)P^T \\
&= PCP^T
\end{aligned}
$$

则P为矩阵C的特征向量组成的矩阵，将 C对角化

则标准化后：

$$y_{white} = \sqrt{D^{-1}}y = \sqrt{D^{-1}}Px$$

ZCA白化要使得其协方差矩阵A为单位矩阵：

$$
\begin{aligned}
A = \frac{1}{m} zz^T = E \\
&= P^T P \\
&= P^T \sqrt{D^{-1}} D \sqrt{D^{-1}} P \\
&= P^T \sqrt{D^{-1}} (\frac{1}{m} yy^T) \sqrt{D^{-1}} P \\
&= \frac{1}{m} (P^T \sqrt{D^{-1}} y)(P^T \sqrt{D^{-1}} y)^T
\end{aligned}
$$

由此可得：

$$z_{white} = P^T \sqrt{D^{-1}} y = P^T \sqrt{D^{-1}} P x$$

PCA白化与ZCA白化：

同：

PCA白化ZCA白化都降低了特征之间相关性较低，同时使得所有特征具有相同的方差。

异：

PCA白化需要保证数据各维度的方差为1，ZCA白化只需保证方差相等

PCA白化可进行降维也可以去相关性，而ZCA白化主要用于去相关性另外

ZCA白化相比于PCA白化使得处理后的数据更加的接近原始数据

Question 5

QDA正确率：0.9891253113746643

LDA正确率： 0.9995272159576416

python版本：3.8.8

文件类型： .ipynb

结果分析：

LDA和QDA都利用PCA降维使特征值大于1e-1的特征向量作为子空间的基，这样避免了因数据精度而导致的行列式为0和无法求逆的情况；

在特征值和特征向量的提取过程中，也有可能出现特征向量出现虚部的情况，经网上查找资料，得知是算法无法收敛而导致的比较小的误差，直接取实部即可；

LDA对数据的分类效果较QDA来说要更好一些，二者之间的区别就是一个是线性判别函数，一个是二次判别函数，且LDA的协方差矩阵是一致的，由此猜想QDA出现了过拟合，而LDA对数据的鲁棒性更好一些

```
import torch
import torchvision
from torch.utils import data
from torchvision import transforms
import math
import numpy as np
```

```
trans = transforms.ToTensor()
mnist_train =
torchvision.datasets.MNIST(root='./data',train=True,transform=trans,download=Tru
e)
mnist_test =
torchvision.datasets.MNIST(root='./data',train=False,transform=trans,download=Tr
ue)
train_len = len(mnist_train)
test_len = len(mnist_test)
train_iter = data.DataLoader(mnist_train, batch_size = train_len, shuffle =
True)
test_iter = data.DataLoader(mnist_test, batch_size = test_len, shuffle = True)
```

```
def train_data(x):
    for X, y in train_iter:
        mask = y==x
        result = X[mask, :, :, :]
        t = result.shape[0]
        return result.reshape((t, -1)), t

def test_data(x):
    for X, y in test_iter:
        mask = y==x
        result = X[mask, :, :, :]
        t = result.shape[0]
        return result.reshape((t, -1)), t
```

```
# 读取数据
zero_train_data, zero_train_len = train_data(0)
one_train_data, one_train_len = train_data(1)
zero_test_data, zero_test_len = test_data(0)
one_test_data, one_test_len = test_data(1)
```

```
# 计算均值
zero_mu = zero_train_data.mean(axis = 0, keepdim = True)
one_mu = one_train_data.mean(axis = 0, keepdim = True)
```

```
# 计算方差
zero_sigma = torch.mm((zero_train_data - zero_mu).T, zero_train_data - zero_mu)
/ zero_train_len
one_sigma = torch.mm((one_train_data - one_mu).T, one_train_data - one_mu) /
one_train_len
```

```python
# PCA降维
def PCA(sigma):
    covM = sigma.numpy()
    eigval, eigvec = np.linalg.eig(covM)
    mask = eigval > 0.1
    eigvec = eigvec[:, mask]
    eigvec = eigvec.real
    return torch.tensor(eigvec)
```

```python
# QDA
def QDA(x, mu, sigma, prior):
    convertMatrix = PCA(sigma)
    x_ = torch.mm(x, convertMatrix)
    mu_ = torch.mm(convertMatrix.T, mu.T)
    sigma_ = torch.mm(torch.mm(convertMatrix.T, sigma),convertMatrix)
    inv_sigma_ = torch.inverse(sigma_)
    W = -0.5 * inv_sigma_
    w = torch.mm(inv_sigma_, mu_)
    w0 = -0.5 *  torch.mm(torch.mm(mu_.T, inv_sigma_),mu_) - 0.5 *
math.log(torch.det(sigma_), math.e) + math.log(prior, math.e)
    return torch.diag(torch.mm(torch.mm(x_, W), x_.T), 0).reshape(-1, 1) +
torch.mm(x_, w) + w0
```

```python
zero_correct_QDA = QDA(zero_test_data, zero_mu, zero_sigma, 0.5) -
QDA(zero_test_data, one_mu, one_sigma, 0.5)
one_correct_QDA = QDA(one_test_data, one_mu, one_sigma, 0.5) -
QDA(one_test_data, zero_mu, zero_sigma, 0.5)
```

```python
zero_correct_QDA[zero_correct_QDA > 0] = 1
zero_correct_QDA[zero_correct_QDA < 0] = 0
one_correct_QDA[one_correct_QDA > 0] = 1
one_correct_QDA[one_correct_QDA < 0] = 0
correct_QDA =(one_correct_QDA.sum() + zero_correct_QDA.sum()) / (one_test_len +
zero_test_len)
```

```python
correct_QDA.item()
```

```
0.9891253113746643
```

```python
# LDA
def LDA(x, mu, sigma, prior):
    convertMatrix = PCA(sigma)
    x_ = torch.mm(x, convertMatrix)
    mu_ = torch.mm(convertMatrix.T, mu.T)
    sigma_ = torch.mm(torch.mm(convertMatrix.T, sigma),convertMatrix)
    inv_sigma_ = torch.inverse(sigma_)
    w = torch.mm(inv_sigma_, mu_)
    w0 = -0.5 * torch.mm(torch.mm(mu_.T, inv_sigma_), mu_) + math.log(prior,
math.e)
    return torch.mm(x_, w) + w0
```

```python
# 两类的先验概率各为0.5
sigma_share = (zero_sigma + one_sigma) / 2
```

```python
zero_correct_LDA = LDA(zero_test_data, zero_mu, sigma_share, 0.5) -
LDA(zero_test_data, one_mu, sigma_share, 0.5)
one_correct_LDA = LDA(one_test_data, one_mu, sigma_share, 0.5) -
LDA(one_test_data, zero_mu, sigma_share, 0.5)
```

```python
zero_correct_LDA[zero_correct_LDA > 0] = 1
zero_correct_LDA[zero_correct_LDA < 0] = 0
one_correct_LDA[one_correct_LDA > 0] = 1
one_correct_LDA[one_correct_LDA < 0] = 0
correct_LDA =(one_correct_LDA.sum() + zero_correct_LDA.sum()) / (one_test_len +
zero_test_len)
```

```python
correct_LDA.item()
```

```
0.9995272159576416
```