实验四 特征选择与特征变换

一、实验目的

掌握特征选择与特征变换的基本思想。通过特征选择与特征变换方法的学习，熟悉特征选择基本概念与方法，熟悉可分性判据，熟悉特征变换基本概念，掌握 K-L 变换方法和主要特点，掌握核函数变换方法和主要特点，掌握 Fisher 变换与判别方法。

二、实验内容

1. 数据集采用 Iris 鸢尾花数据，使用穷举法进行特征选择；
2. 对该数据进行特征变换，采用算法：PCA、KPCA、LDA。

三、实验原理

基本概念

1. 特征选取

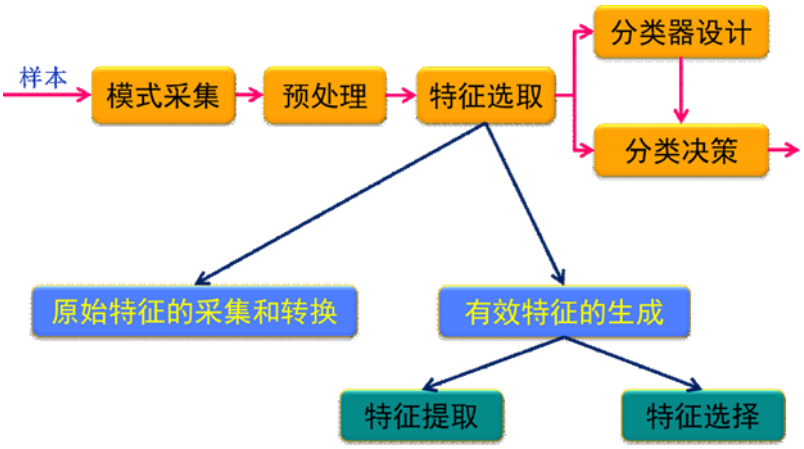


图 1 特征选取的内容

在模式识别系统中，确定分类和学习过程所使用的特征是非常重要的一个环节，获得对分类最有效的特征，同时尽最大可能减少特征维数，是特征选取的主要任务。特征选取可以分成原始特诊的采集和转换、有效特征的生成两个步骤。

（1）原始特征的采集和转换

对于一个模式识别任务，见过模式采集和预处理得到的模式信息不一定能直接用于模式分类，需要从中经过数据处理和转换得到对具体分类任务有效的特征。例如对于模式采集到的图像信息，其原始数据为像素点的颜色值矩阵，而对于不同的模式识别任务和模式识别算法，可以提取出不同类型的特征，如轮廓特征、颜色特征、颜色特征、数学特征。

（2）有效特征的生成

在获得了原始特征后，需要生成有效的特征，其主要目的是大幅度降低特征维度，减少模式识别算法的计算量。如果不经过这一降维过程，可能出现“维数灾难”，无法进行有效的模式识别分类。例如：在文本分类中，如果采用原始的词频统计数据作为分类特征，则有多少个不同的词就有多少维特征，一篇长文的特征维度会超过 1000 维，基本无法进行计算。

在降低特征维度的同时，还要提升所获得特征的有效性，因为尽管特征数量越多，用于分类的信息也越充足，但特征数量与分类有效性之间并不是线性关系。降维到同样数量时，不同的特征对分类的有效性是不同的。特征选取需要采用适当的算法，在降低特征维度的同

时，最大可能地保留对分类有效的信息。

2. 特征提取

特征提取是通过某种变换，将原始特征从高维空间映射到低维空间。

A：$X \rightarrow Y$；　　A 称为特征提取器，通常是某种正交变换。

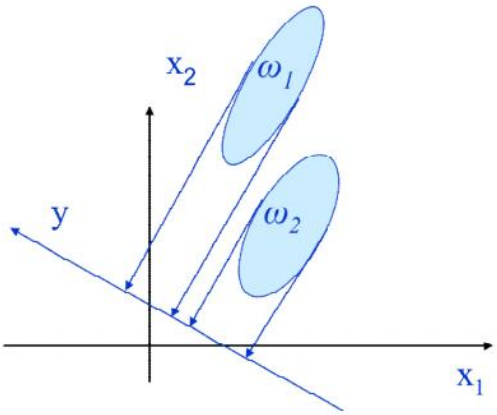


图 2　特征提取

对于各种可能的特征提取器，需要选择最优的一种，也就是降维后分类最有效的一种，通常设定一个准则函数 $J(A)$，使得取到最优特征提取时，准则函数值取到最大值，即 $J(A*)=\max J(A)$。

3. 特征选择

特征选择是从高维特征中挑选出一些最有效的特征，以达到降低特征空间维数的目的。

$$S:\{x_1,x_2,.....,x_D\} \rightarrow F:\{y_1,y_2,.....,y_d\}$$
$$y_i \in S, i=1,2,...,d; d<D$$

原始特征集合 S 中包含 D 个特征，目标特征集合 F 中包含 d 个特征。
同样，对于各种可能的特征选择方案，需要选择最优的一种，也就是降维后分类最有效的一种，通常设定一个准则函数 $J(F)$，使得取到最优特征选择时，准则函数值取到最大值，即 $J(F*)=\max J(F)$。

4. 准则函数的选取

（1）准则函数的选取原则

在设定了准则函数后，求取最优的特征提取或特征选择可以看作一个泛函求极值的问题，因此，准则函数的选取是特征提取或特征选择算法的关键。分类正确率是最佳的准则函数，如果经过某种方案的特征提取或特征选择后，得到的低维特征是所有可能方案中分类正确率最高的，就是最优的特征提取或特征选择。但是分类正确率难以直接计算，因此可以用特征选取方案对类别的可分性测度作为准则函数，通常两类之间的类别可分性测度要满足以下标准：

- 与分类正确率有单调递增关系
- 当特征独立时具有可加性，即 $J_{ij}(x_1,x_2,\ldots,x_d)=\sum_{k=1}^{d}J_{ij}(x_k)$
- 具有标量测度特性：$\begin{cases}J_{ij}>0, & \text{当}i\neq j\text{时}\\ J_{ij}=0, & \text{当}i=j\text{时}\\ J_{ij}=J_{ji}\end{cases}$
- 对特征数量具单调性，即：

$$J_{ij}(x_1,x_2,\ldots,x_d)<J_{ij}(x_1,x_2,\ldots,x_d,x_{d+1})$$

常用的类别可分析测度有基于类内类间距离和概率距离两种。

（2）类内类间距离

对于一个已知的样本集，类内类间距离的数学定义为：

设一个分类问题共有c类，令$x_k^{(i)}$，$x_l^{(j)}$分别为ω_i类及ω_j类中的D维特征向量，$\delta(x_k^{(i)},x_l^{(j)})$为这两个向量间的距离，则各类中各特征向量之间的距离的平均值，称为类内类间距离：

$$J_d(x)=\frac{1}{2}\sum_{i=1}^{c}P_i\sum_{j=1}^{c}P_j\frac{1}{n_in_j}\sum_{k=1}^{n_i}\sum_{l=1}^{n_j}\delta(x_k^{(i)},x_l^{(j)})$$

n_i为ω_i中的样本数，n_j为ω_j中的样本数，P_i，P_j是各类的先验概率。

对于随机性的统计分类，如果样本集是给定的，则无论其中各类样本如何划分，类内类间距离都是相等的，也就是说，类内类间距离本身和分类错误率不相关，不能直接用于类别可分性测度。虽然类内类间距离本身不能用作类别可分性测度，但对其进行分解处理后，可以得到与类别可分性相关的测度指标。

如采用均方欧氏距离来 度量两个特征向量之间 的距离，

则有 $\delta(x_k^{(i)},x_l^{(j)})=(x_k^{(i)}-x_l^{(j)})^{\mathrm{T}}(x_k^{(i)}-x_l^{(j)})$

用m_i表示第i类样本集的均值向量 ：$m_i=\frac{1}{n_i}\sum_{k=1}^{n_i}x_k^{(i)}$

用m表示所有各类样本集的 总均值向量： $m=\sum_{i=1}^{c}P_im_i$

则 $$J_d(x)=\sum_{i=1}^{c}P_i\left[\frac{1}{n_i}\sum_{k=1}^{n_i}(x_k^{(i)}-m_i)^T(x_k^{(i)}-m_i)+(m_i-m)^T(m_i-m)\right]$$

$$=\sum_{i=1}^{c}P_i\frac{1}{n_i}\sum_{k=1}^{n_i}(x_k^{(i)}-m_i)^T(x_k^{(i)}-m_i)+\sum_{i=1}^{c}P_i\frac{1}{n_i}(m_i-m)^T(m_i-m)$$

令类内离散度矩阵和类　间离散度矩阵分别为

$$S_w = \sum_{i=1}^{c} P_i \frac{1}{n_i} \sum_{k=1}^{n_i} \left(x_k^{(i)} - m_i\right)\left(x_k^{(i)} - m_i\right)^T$$

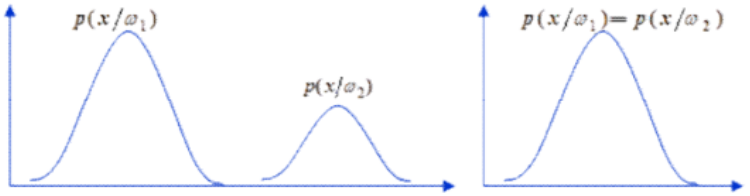
$$S_b = \sum_{i=1}^{c} P_i (m_i - m)(m_i - m)^T$$

则　$J_d(x) = tr(S_w + S_b) = tr(S_w) + tr(S_b) = J_w + J_b$

J_w 称为类内平均距离，J_b 称为是类间平均距离。从类别可分性的要求来看，希望 J_w 尽可能小， J_b 尽可能大。

（3）概率距离

类间的概率距离可用分布函数之间的距离来度量，例如对两类问题：



当两类完全可分时，若 $p(x|\omega_1) \neq 0$，则 $p(x|\omega_2)=0$；当两类完全不可分时：对任意 x，都有 $p(x|\omega_1) = p(x|\omega_2)$；一般情况下，两类会介于完全可分和完全不可分之间。

依据以上度量方式，可定义类别可分析的概率距离准则：

若任何函数 $J_p(\bullet) = \int g[p(x|\omega_1), p(x|\omega_2), P_1, P_2]dx$

满足以下条件：

a、$J_p \geq 0$;

b、当两类完全可分时　J_p取得最大值；

c、当两类完全不可分是　J_p为0；

则可作为两类之间可分　性的概率距离度量。

使用类内类间距离进行特征提取

1. 准则函数的构造

类内类间距离可表示为：$J_d=J_w+J_b= tr(S_w+S_b)$

其中 J_w是类内平均距离，J_b 是类间平均距离。

对于一个给定的样本集，J_d是固定不变的。而通过特征提取后，新获得的特征使得样本集可以划分为不同的类，最佳的特征提取应当是使得各类之间的可分性最好，也就是 J_b 最大，J_w最小。因此，可以直接采用 J_b 作为特征提取的准则函数，称为 J_1 准则。

但直接使用准则难以得到可行的特征提取算法，考虑到类内离散度矩阵 S_w和类间离散度矩阵 S_b 是对称矩阵，迹和行列式值在正交变换下具有不变性，常构造以下几种特征提取准则函数：

$$J_2 = tr(S_w^{-1} S_b), \quad J_3 = \ln\left[\frac{|S_b|}{|S_w|}\right], \quad J_4 = \frac{tr(S_b)}{tr(S_w)}, \quad J_5 = \frac{|S_w + S_b|}{|S_w|}$$

2. 基于 J_2准则的特征提取算法

假设有 D 个原始特征： $x=[x_1,x_2,\ldots,x_D]^T$

通过特征提取后压缩为 d 个特征： $y=[y_1,y_2,\ldots,y_d]^T$

其映射关系为： $y=W^Tx$

令 S_b、S_w 为原始特征空间中样本集的离散度矩阵， S_b^*、S_w^* 为特征提取后新特征空间中样本集的离散度矩阵，则有：

$$S_w^*=W^TS_wW, S_b^*=W^TS_bW$$

对于 J_2 准则，进行特征提取后，准则函数值为：

$$J_2=tr\left(S_w^{*-1}S_b^*\right)=tr[(W^TS_wW)^{-1}W^TS_bW]$$

求最优的特征提取，就是求最优的变换阵 W，使得准则函数值在此变换下能取得最大值。

将准则函数对 W 求偏导，并令其为 0，解出的 W 就是可使得准则函数 J_2 取得最大值的变换阵。结论为：

将矩阵 $S_w^{-1}S_b$ 的特征值按大小排序： $\lambda_1\geq\lambda_2\geq\ldots\geq\lambda_D$

则前 d 个特征值对应的特征向量 $\mu_1,\mu_2,\ldots,\mu_d$ 可构成变换阵 W，即

$$W=[\mu_1,\mu_2,\ldots,\mu_d]$$

此时的准则函数值为：

$$J_2(W)=\sum_{i=1}^{d}\lambda_i$$

基于 J_2 准则的特征提取算法事实上是保留了原特征空间中方差最大的特征维度成份。

特征选择算法

特征选择是从 D 个特征中挑选出最有效的 d 个特征的过程，常用的算法有以下几种。

1. 独立算法

独立算法是指分别计算 D 个特征单独使用时的准则函数值，选取最优的前 d个特征。除非各特征相互独立，准则函数满足可加性，否则独立算法所得到的特征组合均不能保证是最优的特征组合。因此除特殊情况外，独立算法并不实用。

2. 穷举算法

特征选择是一个组合过程，因此如从 D 个特征中考查所有可能的 d 个特征组合，计算其准则函数，找到最优的一个，从而得到最佳的特征选择结果，就是穷举法的算法。穷举法可以保证得到所有解中的全局最优解，但问题是计算量太大，例如当D=100，d=10，需要经过：

$$q=C_D^d=\frac{D!}{(D-d)!d!}=\frac{100!}{(100-10)!10!}=17310309456440$$

次计算才能得到特征选择结果，这在有限资源下基本是无解的。

其他算法包括分支限界算法、次优算法、遗传算法等。

特征变换算法

1. 主成分分析（PCA）

PCA 运算就是一种确定一个坐标系统的直交变换，在这个新的坐标系统下，变换数据点的方差沿新的坐标轴得到了最大化。这些坐标轴经常被称为是主成分。PCA 运算是一个利用了数据集的统计性质的特征空间变换，这种变换在无损或很少损失了数据集的信息的情况下降低了数据集的维数。

PCA 的主要步骤如下：

设训练样本集 $x=[x_1,x_2,...,x_n]^T$，均值 $\bar{x}=E(x)$。

协方差 $c_x=E[(x-\bar{x})(x-\bar{x})^T]$ 为是对称阵，可用来分解为 $c_x=UD_\lambda U^T$，其中 $U^T=U^{-1}$，$UU^T=I$，

$$D_\lambda=diag[\lambda_1,\lambda_2,...,\lambda_n]=\begin{bmatrix}\lambda_1 & 0 & \cdots. & 0\\ 0 & \lambda_2 & \cdots & 0\\ \vdots & \vdots & \ddots & \vdots\\ 0 & 0 & \cdots & \lambda_n\end{bmatrix}，且\ \lambda_1\geq\lambda_2\geq\cdots\geq\lambda_n\geq 0$$

矩阵 U 可以写成 $U=[\phi_1,\phi_2,\cdots,\phi_n]$，其中 $\phi_i=[\phi_{i1},\phi_{i2},\cdots,\phi_{in}]^T$ 是列向量，$c_xU=UD_\lambda U^TU=UD_\lambda$，得 $c_x[\phi_1,\phi_2,...,\phi_n]=[\phi_1,\phi_2,...,\phi_n]D_\lambda$

选取较大的 k 个特征值所对应的特征向量（主成分）组成变换矩阵 U^*，对测试数据进行变换，变换后的新模式向量为 $Y=X_{test}U^*$，降低了数据维数，最后再对向量 Y 进行分类。

各主成分的的累积贡献率定义为：$\sum_{i=1}^{k}\lambda_i\Big/\sum_{i=1}^{n}\lambda_i$

我们进行主成分分析的目的之一就是希望利用尽可能少的主成分来代替原来的 n 个指标，在实际中需要根据具体情况具体分析。

2. 核主成分分析（KPCA）

假设 $x_1, x_2, ..., x_M$ 为训练样本，用 $\{x_i\}$ 表示输入空间。KPCA 方法的基本思想是通过某种隐式方式将输入空间映射到某个高维空间(常称为特征空间)，并且在特征空间中实现 PCA。假设相应的映射为 Φ，其定义如下

$$\Phi : \mathbb{R}^d \to F$$
$$x \mapsto \xi = \Phi(x)$$

核函数通过映射 Φ 将隐式的实现点 x 到 F 的映射，并且由此映射而得的特征空间中数据满足中心化的条件，即

$$\sum_{\mu=1}^{M} \Phi(x_\mu) = 0 \tag{1}$$

则特征空间中的协方差矩阵为：

$$C = \frac{1}{M} \sum_{\mu=1}^{M} \Phi(x_\mu) \Phi(x_\mu)^T \tag{2}$$

现求 C 的特征值 $\lambda \geq 0$ 和特征向量

$$V \in F \setminus \{0\}, \quad C\nu = \lambda \nu \tag{3}$$

即有

$$(\Phi(x_\nu) \cdot C\nu) = \lambda (\Phi(x_\nu) \cdot \nu) \tag{4}$$

考虑到所有的特征向量可表示为 $\Phi(x_1), \Phi(x_2), ..., \Phi(x_M)$ 的线性张成，即

$$\nu = \sum_{i=1}^{M} \alpha_i \Phi(x_i) \tag{5}$$

则有

$$\frac{1}{M} \sum_{\mu=1}^{M} \alpha_\mu \left(\sum_{w=1}^{M} (\Phi(x_\nu) \cdot \Phi(x_w) \Phi(x_w) \Phi(x_\mu)) \right) = \lambda \sum_{\mu=1}^{M} (\Phi(x_\nu) \cdot \Phi(x_\mu)) \tag{6}$$

其中 $\nu = 1, 2, ..., M$。定义 $M \times M$ 维矩阵 K

$$K_{\mu\nu} \coloneqq (\Phi(x_\mu)\cdot\Phi(x_\nu)) \tag{7}$$

则式子(6)可以简化为

$$M\lambda K\alpha = K^2\alpha \tag{8}$$

显然满足

$$M\lambda\alpha = K\alpha \tag{9}$$

求解(9)就能得到特征值和特征向量，对于测试样本在特征向量空间V^k的投影为

$$(v^k\cdot\Phi(x)) = \sum_{i=1}^{M}(\alpha_i)^k(\Phi(x_i),\Phi(x)) \tag{10}$$

将内积用核函数替换则有

$$(v^k\cdot\Phi(x)) = \sum_{i=1}^{M}(\alpha_i)^k K(x_i,x) \tag{11}$$

当(1)不成立时，需进行调整，

$$\Phi(x_\mu)\to\Phi(x_\mu)-\frac{1}{M}\sum_{\nu=1}^{M}\Phi(x_\nu)\qquad \mu=1,...,M \tag{12}$$

则核矩阵可修正为

$$K_{\mu\nu}\to K_{\mu\nu}-\frac{1}{M}(\sum_{u=1}^{M}K_{\mu w}+\sum_{w=1}^{M}K_{w\nu})+\frac{1}{M^2}\sum_{w,t=1}^{M}K_{wt} \tag{13}$$

基于上述 KPCA 的基本原理，可得 KPCA 的处理过程如下：

1、将所获得的n个指标(每一指标有m个样品)的一批数据写成一个$(m\times n)$维数据矩阵

$$A=\begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}。$$

2、计算核矩阵，先选定高斯径向核函数中的参数，再由式(7)，计算核矩阵K。

3、通过(13)中心化核矩阵得到 KL。

4、计算 KL 的特征值$\lambda_1,...,\lambda_n$即对应的特征向量$v_1,...,v_n$。

5、特征值按降序排序(通过选择排序)得$\lambda_1^{'}>...>\lambda_n^{'}$并对特征向量进行相应调整得$v_1^{'},...,v_n^{'}$。

6 、单位正交化特征向量，得到$\alpha_1,...,\alpha_n$。

7、计算特征值的累积贡献率 $B_1,...,B_n$，根据给定的提取效率 p，如果 $B_t \geq p$，则提取 t 个主分量 $\alpha_1,...,\alpha_t$。

8、计算已修正的核矩阵 X 在提取出的特征向量上的投影 $Y = KL \cdot \alpha$，其中 $\alpha = (\alpha_1,...,\alpha_t)$。

所得的投影 Y 即为数据经 KPCA 降维后所得数据。

3. 线性判别分析

假设有一组属于两个类的 n 个 d 维样本 $x_1,...,x_n$，其中前 n_1 个样本属于类 ω_1，后面 n_2 个样本属于类 ω_2，均服从同协方差矩阵的高斯分布。各类样本均值向量 m_i（i=1，2）如式（1）：

$$m_i = \frac{1}{n_i} \sum_{x \in X_t} X \qquad i=1，2 \qquad (1)$$

样本类内离散度矩阵 S_i 和总的类内离散度矩阵 S_w 如式(2)、式（3）：

$$S_i = m_i \sum_{x \in X_t} (x - m_i)(x - m_i)^T \qquad i=1，2 \qquad (2)$$

$$S_w = S_1 + S_2 \qquad (3)$$

样本类间离散度矩阵 S_b 如式(4)：

$$S_b = (m_1 - m_2)(m_1 - m_2)^T \qquad (4)$$

现寻找一个最佳超平面将两类分开，则只需将所有样本投影到此超平面的法线方向上 $\omega \in R^d$，　||w||=1:

$$y_i = w^T x_i \qquad i=1,\cdots,n \qquad (5)$$

得到 n 个标量 $y_1,...,y_n \in R$，这 n 个标量相应的属于集合 Y_1 和 Y_2，并且 Y_1 和 Y_2 能很好的分开。

为了能找到这样的能达到最好分类效果的投影方向 w，Fisher 规定了一个准则函数:要求选择的投影方向 W 能使降维后 Y_1 和 Y_2 两类具有最大的类间距离与类内距离比:

$$J_F(w)=\frac{(\overline{m_1}-\overline{m_2})^2}{\overline{s_1}^2+\overline{s_2}^2} \quad (6)$$

其中类间距离用两类均值$\overline{m_1}$和$\overline{m_2}$之间的距离表示，类内距离用每类样本距其类均值距离的和表示，在式中为$\overline{s_1}^2+\overline{s_2}^2$。

其中$\overline{m_i}$(i=1，2)为降维后各类样本均值：

$$m_i=\frac{1}{n_i}\sum_{y=Y_i} y \qquad i=1，2 \quad (7)$$

$\overline{s_i}^2$（i=1，2)为降维后每类样本类内离散度，$\overline{s_1}^2+\overline{s_2}^2$为总的类内离散度$\overline{s_w}$：

$$\overline{s_i}^2=\sum(y-\overline{m_i})^2， \qquad i=1，2 \quad (8)$$

$$\overline{s_w}=\overline{s_1}^2+\overline{s_2}^2 \quad (9)$$

类间离散度表示为$(\overline{m_1}-\overline{m_2})^2$。但式(6)Fisher 准则函数并不是 w 的显示函数，无法根据此准则求解 W，因此需要对 Fisher 准则函数形式进行修改：

因$y_i=w^T x_i$ i=1,…,n ，则

$$\overline{m_i}=\frac{1}{n_i}\sum_{y=Y_i} y=\frac{1}{n_i}\sum_{x\in X_T} w^T X=w^T m_i \qquad i=1,2 \quad (10)$$

$$(\overline{m_1}-\overline{m_2})^2=(w^T m_1-w^T m_2)^2=w^T(m_1-m_2)(m_1-m_2)^T w \quad (11)$$

$$=w^T s_b w$$

同样s_i^2（i=1，2）也可推出与 w 的关系：

$$\overline{s_i}^2=\sum_{y\in Y_t}(y-\overline{m_i})^2=\sum_{x\in X_t}(w^T x-w^T m_i)^2$$

$$=w^T[\sum_{x\in X_t}(x-m_i)(x-m_i)^T]w=w^T S_i w \quad (12)$$

因此

$$\overline{s_1}^2 + \overline{s_2}^2 = w^T(s_1 + s_2)w = w^T s_w w \quad (13)$$

则最终可表示为：

$$J_F(w) = \frac{(\overline{m_1} - \overline{m_2})^2}{\overline{s_1}^2 + \overline{s_2}^2} = \frac{w^T s_b w}{w^T s_w w} \quad (14)$$

根据式(14)Fisher 准则函数，要寻找一投影向量 W，使 $J_F(w)$ 最大化，则需对 $J_F(w)$ 按变量 W 求导并使之为零：

$$\frac{\partial J_F(w)}{\partial w} = \frac{\partial(\frac{w^T s_b w}{w^T s_w w})}{\partial w} = \frac{s_b w(w^T s_w w) - s_w w(w^T s_b w)}{(w^T s_w w)^2} = 0 \quad (15)$$

则需

$$s_b w（w^T s_w w）- s_w w(w^T s_b w)=0$$

$$s_b w = J_F(w)\, s_w w \quad (16)$$

令 $J_F(w) \equiv \lambda$，则

$$WS_b = W\lambda S_w \quad (17)$$

这是一个广义特征值问题，若 s_w 非奇异，则

$$WS_w^{-1}S_b = \lambda W \quad (18)$$

因此可以通过对 $s_w^{-1} s_b$ 进行特征值分解，将最大特征值对应的特征向量作为最佳投影方向 W。

四、实验过程

1. 穷举法进行特征选择

Iris 数据集包含三类样本，分别是“setosa”、“versicolor“、”virginica“，每个样本有 4 个特征，分别是：'sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)'，因此使用穷举法时分别有 4 选 1、4 选 2、4 选 3、4 选 4，总共 $C_4^1 + C_4^2 + C_4^3 + C_4^4 = 15$ 种选择。用穷举法选择特征后，使用 sklearn 中的 LogisticRegression 模型进行训练（60%作为训练集、40%作为测试集，运行 500 次求平均）得到分类结果。可以看到，随着所选特征数的增多，分类准确率 Accuracy 也基本呈现上升

的趋势；选择 1 个特征时，'petal width (cm)'的分类效果最好；选择 2 个特征时，'sepal width (cm)'和 'petal width (cm)' 的分类效果最好；选择 3 个特征时，'sepal width (cm)', 'petal length (cm)', 'petal width (cm)'的分类效果最好；而选择全部 4 个特征时，分类效果在所有穷举法选出的特征组合中，效果是最好的。

选择特征	Accuracy
1	0.5980
2	0.4880
3	0.7497
4	0.8078
1，2	0.6912
1，3	0.8951
1，4	0.8553
2，3	0.8651
2，4	0.8978
3，4	0.8193
1，2，3	0.9076
1，2，4	0.8905
1，3，4	0.9264
2，3，4	0.9314
1，2，3，4	0.9400

2. 特征变换—PCA

用 PCA 方法进行特征变换后，使用 sklearn 中的 LogisticRegression 模型进行训练（60%作为训练集、40%作为测试集，运行 1000 次求平均）得到分类结果。在一次试验中分解得到的特征值为[4.19667516 0.24062861 0.07800042 0.02352514]，特征向量矩阵如下：

```
[[ 0.36158968 -0.65653988  0.58099728  0.31725455]
 [-0.08226889 -0.72971237 -0.59641809 -0.32409435]
 [ 0.85657211  0.1757674  -0.07252408 -0.47971899]
 [ 0.35884393  0.07470647 -0.54906091  0.75112056]]
```

可以看出，第一个特征值明显大于其他特征值，该特征值所对应的主成分对数据分类起了最主要的辨识作用。

当选择全部 4 个主成分时（评价指标分别为 Accuracy、Precision、Recall、F1），实验结果如下：

累计贡献率	Accuracy	Precision	Recall	F1
1	0.8999	0.9094	0.9032	0.8985

选择特征值大的前 3 个主成分时，实验结果如下：

累计贡献率	Accuracy	Precision	Recall	F1
0.9948	0.9009	0.9120	0.9044	0.8994

选择特征值大的前 2 个主成分时，实验结果如下：

累计贡献率	Accuracy	Precision	Recall	F1
0.9776	0.8686	0.8862	0.8735	0.8659

选择特征值大的前 1 个主成分时，实验结果如下：

累计贡献率	Accuracy	Precision	Recall	F1
0.9246	0.8581	0.8839	0.8645	0.8499

从上面的数据可以看出，数据进行降维之后的分类效果受到一定的影响，并且随着选取特征值的数目的增加，分类的效果逐渐上升。从之前的单次实验中可以看出，第一个特征值明显大于其他特征值，对应的主成分所含的数据信息量最大，对分类起到了最主要作用。因此，可以抛弃该主成分，使用其他三个主成分进行分类。

选择第 2 个主成分时，实验结果如下：

贡献率	Accuracy	Precision	Recall	F1
0.053	0.4213	0.3716	0.4416	0.3715

选择第 3 个主成分时，实验结果如下：

贡献率	Accuracy	Precision	Recall	F1
0.0172	0.4184	0.3848	0.4452	0.3728

选择第 4 个主成分时，实验结果如下：

贡献率	Accuracy	Precision	Recall	F1
0.0052	0.2943	0.2264	0.3387	0.209

可以看出，这 3 个主成分对对分类任务并未起到主要作用。

程序实现：

```
class PCA:
    def __init__(self, n_components=None):
        self.n_components = n_components

    def fit(self, X):
        self.mean_ = X.mean(axis=0)
        self.n_features = X.shape[1]
        _, sigmas, VT = svd(X - self.mean_, full_matrices=False)
        V = VT.T
        a = np.argsort(sigmas)
        # a = np.array([0,3,2,1])
        V = V[:, np.argsort(sigmas)[::-1]]
        # V = V[:, a[::-1]]
        # V = V[:, [3,0,1,2]]
        b = np.linalg.norm(V, axis=0)
        V /= np.linalg.norm(V, axis=0)
        self.eigenvalue = sigmas**2 / X.shape[0]
        self.scalings_ = V

    def transform(self, X):
        if not hasattr(self, 'scalings_'):
            raise Exception('Please run `fit` before transform')
        if not hasattr(self, 'mean_'):
            raise Exception('Please run `fit` before transform')
        assert X.shape[1] == self.n_features, 'X.shape[1] !=
self.n_features'
```

```
        if self.n_components is None:
            self.n_components = self.n_features
        return ((X - self.mean_) @
self.scalings_)[:, :self.n_components]

    def fit_transform(self, X):
        self.fit(X)
        return self.transform(X)
```

3. 特征变换—KPCA

用 KPCA 方法进行特征变换后，使用 sklearn 中的 LogisticRegression 模型进行训练（60%作为训练集、40%作为测试集，运行 500 次求平均）得到分类结果。本实验针对 IRIS 数据进行核主成分分析，采用线性核函数（linear）、多项式核函数（poly）、径向基核函数（rbf）。

- 线性核函数（linear）

centered kernel matrix 的特征值为：
[629.50127448 36.09429217 11.70006231 3.52877104]

当选择全部 4 个主成分时（评价指标分别为 Accuracy、Precision、Recall、F1），实验结果如下：

累计贡献率	Accuracy	Precision	Recall	F1
1.0	0.8997	0.9094	0.9032	0.8982

选择特征值大的前 3 个主成分时，实验结果如下：

累计贡献率	Accuracy	Precision	Recall	F1
0.9948	0.8975	0.9097	0.9014	0.8962

选择特征值大的前 2 个主成分时，实验结果如下：

累计贡献率	Accuracy	Precision	Recall	F1
0.9776	0.8691	0.8865	0.8735	0.8663

选择特征值大的前 1 个主成分时，实验结果如下：

累计贡献率	Accuracy	Precision	Recall	F1
0.9246	0.8578	0.8832	0.8637	0.8496

可以将采用线性核函数（linear）的 KPCA 与 PCA 得到的特征贡献度进行对比，发现其值是一样的，验证了采用线性核函数（linear）的 KPCA 就等价于 PCA。

- 多项式核函数（poly）

centered kernel matrix 的特征值为：
[251974.73068994 7339.55084925 3578.31449477 1071.06819495]

当选择全部 4 个主成分时（评价指标分别为 Accuracy、Precision、Recall、F1），实验结果如下：

累计贡献率	Accuracy	Precision	Recall	F1
1.0	0.9260	0.9278	0.9265	0.9252

选择特征值大的前 3 个主成分时，实验结果如下：

累计贡献率	Accuracy	Precision	Recall	F1

0.9959	0.9240	0.9282	0.9259	0.9215

选择特征值大的前 2 个主成分时，实验结果如下：

累计贡献率	Accuracy	Precision	Recall	F1
0.9824	0.9168	0.9244	0.9197	0.9137

选择特征值大的前 1 个主成分时，实验结果如下：

累计贡献率	Accuracy	Precision	Recall	F1
0.9546	0.8216	0.8425	0.8277	0.8075

- 径向基核函数（rbf）

centered kernel matrix 的特征值为：

[48.08181865 19.09195919 6.62368557 4.31294935]

当选择全部 4 个主成分时（评价指标分别为 Accuracy、Precision、Recall、F1），实验结果如下：

累计贡献率	Accuracy	Precision	Recall	F1
1.0	0.9124	0.9163	0.9154	0.9118

选择特征值大的前 3 个主成分时，实验结果如下：

累计贡献率	Accuracy	Precision	Recall	F1
0.9448	0.9060	0.9096	0.9083	0.9048

选择特征值大的前 2 个主成分时，实验结果如下：

累计贡献率	Accuracy	Precision	Recall	F1
0.8600	0.9046	0.9085	0.9073	0.9037

选择特征值大的前 1 个主成分时，实验结果如下：

累计贡献率	Accuracy	Precision	Recall	F1
0.6156	0.6378	0.5385	0.6667	0.5592

- PCA 与 KPCA

主成分分析属于代数特征分析方法，是模式识别领域中一种经典的特征抽取和降维方法。但是 PCA 的缺点是需要很大的存储空间和计算复杂度。如果原始空间的维数是n，PCA 需要分解一个n n 的非稀疏矩阵。因为PCA 是一种线性映射方法，降维后的表示是由线性映射生成的，它忽略了数据之间高于 2 阶的相互关系，所以抽取的特征并不是最优的，这在一定程度上影响了 PCA 方法的效果。核主成分分析是线性 PCA 的非线性扩展算法，它采用非线性的方法抽取主成分，即KPCA是在通过映射函数 把原始向量映射到高维空间 F，在 F上进行 PCA分析。

KPCA与PCA具有本质上的区别：PCA 是基于指标的，而KPCA是基于样本的。KPCA不仅适合于解决非线性特征提取问题，而且它还能比PCA提供更多的特征数目和更多的特征质量，因为前者可提供的特征数目与输入样本的数目是相等的，而后者的特征数目仅为输入样本的维数。KPCA的优势是可以最大限度地抽取指标的信息；但是KPCA抽取指标的实际意义不是很明确，计算也比PCA复杂。

程序实现：

```
class KernelPCA:
    def __init__(self, n_components=None, kernel='linear',
```

```
kernel_para=0.1):
        self.__kernels = {'linear': self.__kernel_linear,
                          'rbf': self.__kernel_rbf}
        assert kernel in self.__kernels, 'arg kernel =\'' + kernel + '\'
is not available'
        self.n_components = n_components
        self.kernel = kernel
        self.kernel_para = kernel_para

    def __kernel_linear(self, x, y):
        return x.T @ y

    def __kernel_rbf(self, x, y):
        result = np.zeros((x.shape[1], y.shape[1]))
        for i in range(result.shape[0]):
            for j in range(result.shape[1]):
                result[i, j] = np.exp(-self.kernel_para * (x[:, i] -
y[:, j]).T @ (x[:, i] - y[:, j]))
        return result

    def fit(self, X):
        self.n_samples, self.n_features = X.shape
        K = self.__kernels[self.kernel](X.T, X.T)
        one_M = np.ones((self.n_samples, self.n_samples)) /
self.n_samples
        K = K - one_M @ K - K @ one_M + one_M @ K @ one_M
        e_vals, e_vecs = eig(K)
        e_vals, e_vecs = np.real(e_vals), np.real(e_vecs)
        e_vecs /= np.linalg.norm(e_vecs, axis=0)
        e_vecs = e_vecs[:, np.argsort(e_vals)[::-1]] /
np.sqrt(np.sort(e_vals)[::-1])
        self.scalings_ = e_vecs
        self.e_vals_ = e_vals
        self.__X = X

    def transform(self, X):
        if not hasattr(self, 'scalings_'):
            raise Exception('Please run `fit` before transform')
        assert X.shape[1] == self.n_features, 'X.shape[1] !=
self.n_features'
        if self.n_components is None:
            self.n_components = X.shape[1]
        K = self.__kernels[self.kernel](X.T, self.__X.T)
        one_M = np.ones((self.n_samples, self.n_samples)) /
```

```
self.n_samples
        K = K - one_M @ K - K @ one_M + one_M @ K @ one_M
        return (K @ self.scalings_)[:, :self.n_components]

    def fit_transform(self, X):
        self.fit(X)
        return self.transform(X)
```

4. 特征变换—LDA

（1）对一二类进行分类：

第一类样本样本均值m_1：（5.006, 3.418, 1.464, 0.244）

第一类样本样本均值m_2：（5.936, 2.77 , 4.26 , 1.326）

类间离散度矩阵S_b:

[[0.8649 -0.60264 2.60028 1.00626],
[-0.60264 0.419904 -1.811808 -0.701136],
[2.60028 -1.811808 7.817616 3.025272],
[1.00626 -0.701136 3.025272 1.170724]]

类内散度矩阵S_w：

[[19.1434 9.0886 9.7528 3.25],
[9.0886 11.9388 4.6224 2.5794],
[9.7528 4.6224 12.2952 3.8612],
[3.25 2.5794 3.8612 2.4794]]

第一类与第二类最优W_{12}为：

W_{12} = (0.06600043 0.42695115 -0.49229259 -0.75564851)

（2）对一三类进行分类：

第一类样本样本均值m_1：（5.006, 3.418, 1.464, 0.244）

第一类样本样本均值m_3：（6.588, 2.974, 5.552, 2.026）

类间离散度矩阵S_b:

[[2.502724 -0.702408 6.467216 2.819124],
[-0.702408 0.197136 -1.815072 -0.791208],
[6.467216 -1.815072 16.711744 7.284816],
[2.819124 -0.791208 7.284816 3.175524]]

类内散度矩阵S_w：

[[25.901 9.509 15.652 2.9224],
[9.509 12.21 4.07 2.8942],
[15.652 4.07 16.4 2.6716],
[2.9224 2.8942 2.6716 4.2594]]

第一类与第三类最优W_{13}为：
W_{13} = (0.28105905 0.22080764 -0.65189412 -0.66879283)

（3）对二三类进行分类：
第一类样本样本均值m_2：（5.936, 2.77 , 4.26 , 1.326）
第一类样本样本均值m_3：（6.588, 2.974, 5.552, 2.026）

类间离散度矩阵S_b:
[[0.425104 0.133008 0.842384 0.4564],
[0.133008 0.041616 0.263568 0.1428],
[0.842384 0.263568 1.669264 0.9044],
[0.4564 0.1428 0.9044 0.49]]

类内散度矩阵S_w：
[[32.868 8.7684 23.8232 5.1388],
[8.7684 9.9212 7.5476 4.3528],
[23.8232 7.5476 25.7448 5.9744],
[5.1388 4.3528 5.9744 5.6124]]

第二类与第三类最优W_{23}为：
W_{23} = (-0.22684996 -0.35584988 0.44461153 0.79008262)

首先进行一二两类的分类，将第一二两类数据与W_{12}做点积，得到第一类样本均值 -1.52472315，第二类样本均值 0.88462257，两类样本中心为-0.32005029。比样本中心大的值判为第一类，比样本中心小的值判为第二类，第一二两类样本均分类正确。

同样对第一三类样本分类，将一三类数据与W_{13}做点积，得到第一类样本均值-2.46599151，第三类样本均值 1.04414367，两类样本中心为-0.71092392，同样所有样本分类正确。

对第二类和第三类样本分类，将第二三类数据与W_{23}做点积，得到第二类样本均值-1.51640554，第三类样本均值 0.60940916，两类样本中心为 1.06290735，第二类的第 21，34 个样本分类错误，第三类的第 34 个样本分类错误。

程序实现：

```
class LinearDiscriminantAnalysis:
    def __init__(self, n_components=None):
        self.n_components = n_components

    def fit(self, X, y):
        self.n_features = X.shape[1]
        u1 = X[y == 1, :].mean(axis=0).reshape((-1, 1))
        u0 = X[y == 0, :].mean(axis=0).reshape((-1, 1))
        Sb = (u1 - u0) @ (u1 - u0).T
        Sw = (X[y == 1, :].T - u1) @ (X[y == 1, :].T - u1).T + (X[y ==
0, :].T - u0) @ (X[y == 0, :].T - u0).T
```

```
        e_vals, e_vecs = eig(np.linalg.pinv(Sw) @ Sb)
        e_vals, e_vecs = np.real(e_vals), np.real(e_vecs)
        e_vecs = e_vecs[:, np.argsort(e_vals)[::-1]]
        e_vecs /= np.linalg.norm(e_vecs, axis=0)
        self.scalings_ = e_vecs

    def transform(self, X):
        if not hasattr(self, 'scalings_'):
            raise Exception('Please run `fit` before transform')
        assert X.shape[1] == self.n_features, 'X.shape[1] !=
self.n_features'
        if self.n_components is None:
            self.n_components = self.n_features
        return (X @ self.scalings_)[:, :self.n_components]

    def fit_transform(self, X, y):
        self.fit(X, y)
        return self.transform(X)
```

五、实验结论

通过本次实验，基本掌握了特征选择与特征变换的基本思想。通过特征选择与特征变换方法的学习，熟悉了特征选择基本概念与方法、可分性判据以及特征变换基本概念，掌握了核函数变换方法和主要特点及 Fisher 变换与判别方法。