第 5 章　特征降维与特征选择

特征降维是指减少特征的个数，最终的结果就是特征和特征之间不相关。本章首先重点介绍特征降维的线性判别分析和主成分分析两种方法。其后，介绍特征选择的三大方法：包装法、过滤法和嵌入法。其中，包装法有递归特征消除和交叉验证递归特征消除两种方法，过滤法有移除低方差特征和单变量特征选择两种方法。随后，重点介绍了皮尔森相关系数法。Sklearn 提供了 SelectFromModel 函数。

5.1　初识特征降维

由于特征矩阵过大，会导致计算量大、训练时间长，因此降低特征矩阵维度必不可少。特征降维是通过选取有代表性的特征，减少特征个数，得到一组不相关主变量的过程，具有如下作用：

(1) 降低时间的复杂度和空间复杂度。

(2) 使得较简单的模型具有更强的鲁棒性。

(3) 便于实现数据的可视化。

特征降维具有线性判别分析(Linear Discriminant Analysis，LDA)和主成分分析(Principal Component Analysis，PCA)等方法。LDA 和 PCA 有很多相似之处，其本质是将原始的样本映射到维度更低的样本空间中。但是 PCA 和 LDA 的映射目标不一样，具体如下：

(1) LDA 是为了让映射后的样本有最好的分类性能，而 PCA 是为了让映射后的样本具有最大的发散性。

(2) LDA 是有监督的降维方法，而 PCA 是无监督的降维方法。

(3) LDA 最多降到类别减 1 的维数，而 PCA 没有这个限制。

(4) LDA 除了可以用于降维，还可以用于分类。

5.2　线性判别分析

5.2.1　线性判别分析简介

线性判别分析是一种经典的降维方法。现假设有红、蓝两色的二维数据，投影到一维，要求同色数据的投影点尽可能接近，不同色数据的投影点尽可能远，使得红色数据中心和蓝色数据中心的距离尽可能大。例如，图 5.1 中右图的投影效果好于左图的投影效果。

线性判别分析就是寻找这样的一条线：$\boldsymbol{y}=\boldsymbol{w}^{\mathrm{T}}\boldsymbol{x}$，使得投影后类内方差最小，类间方

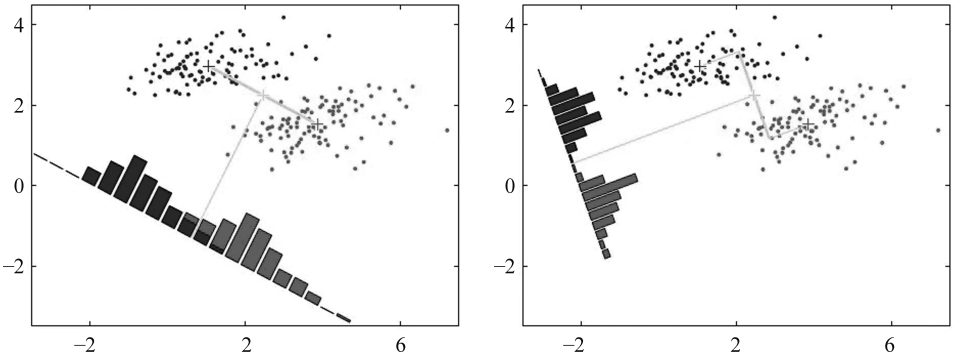

图 5.1　将二维数据投影到一维的两种效果

差最大，如图 5.2 所示。

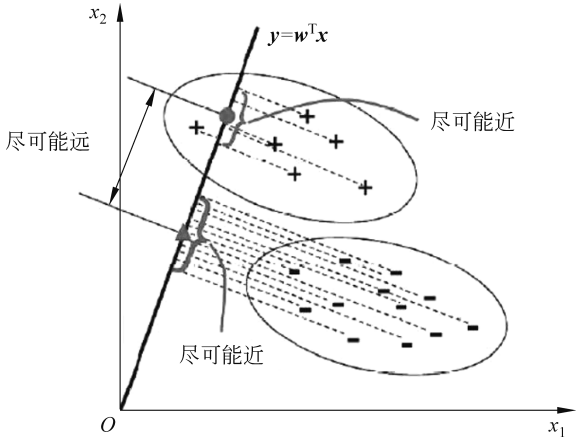

图 5.2　线性判别分析示例

5.2.2　线性判别分析示例

Sklearn 提供了 discriminant_analysis.LinearDiscriminantAnalysis 函数，用于实现线性判别分析，语法如下：

```
LinearDiscriminantAnalysis (n_components=n)
```

参数 n_components＝n 等号右侧的 n 代表减少的维数。

【例 5.1】 LDA 示例。

```
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
iris=datasets.load_iris()                    #鸢尾花数据集
X=iris.data
```

```
y=iris.target
target_names=iris.target_names
lda=LinearDiscriminantAnalysis(n_components=2)
X_r2=lda.fit(X, y).transform(X)
plt.figure()
for c, i, target_name in zip("rgb", [0, 1, 2], target_names):
    plt.scatter(X_r2[y==i, 0], X_r2[y==i, 1], c=c, label=target_name)
plt.legend()
plt.title('LDA of IRIS dataset')
plt.show()
```

【程序运行结果】

程序运行结果如图 5.3 所示。

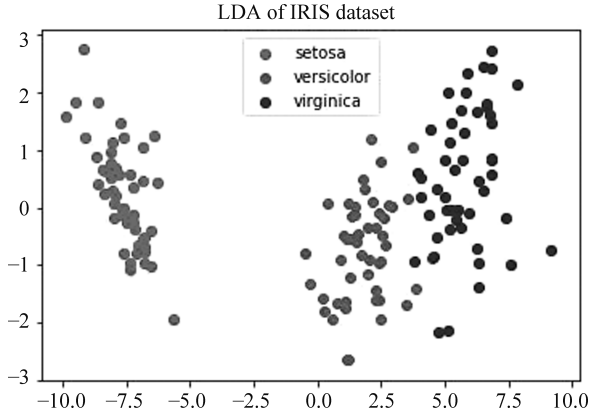

图 5.3　程序运行结果

5.3　主成分分析

5.3.1　主成分分析简介

在多变量的问题中，变量之间往往存在信息重叠。可以通过正交变换将一组可能存在相关性的变量转换为一组线性不相关的变量，转换后的变量称为主成分。主成分分析将关系紧密的变量删去，从而使得特征变得简单。其主要优点如下：

(1) 仅以方差衡量信息量。

(2) 各主成分两两正交，可消除原始数据成分间的相互影响因素。

(3) 计算方法简单，主要运算是特征值分解，易于实现。

主成分分析算法的主要缺点如下：

(1) 主成分各个特征维度的含义具有一定的模糊性。

(2) 非主成分也可能含有对样本差异的重要信息。

5.3.2 components 参数

Sklearn 提供了 decomposition.PCA 函数，用于实现主成分分析，具体语法如下：

```
PCA(n_components=n)
```

参数 n_components 取值有小数和整数之分。取值为小数时，表示保留的百分比；取值为整数时，则表示保留的特征数。

【例 5.2】 n_components 示例。

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.datasets.samples_generator import make_blobs
    #make_blobs:多类单标签数据集,为每个类分配一个或多个正态分布的点集
    #X 为样本特征,Y 为样本簇类别。共 1000 个样本,每个样本 3 个特征,共 4 个簇
    X,y=make_blobs(n_samples=10000, n_features=3, centers=[[3,3,3],[0,0,0],
        [1,1,1],[2,2,2]], cluster_std=[0.2, 0.1, 0.2, 0.2], random_state=9)
fig=plt.figure()
ax=Axes3D(fig, rect=[0, 0, 1, 1], elev=30, azim=20)
plt.scatter(X[:, 0], X[:, 1], X[:, 2],marker='o')
```

【程序运行结果】

```
scale=np.sqrt(self._sizes) * dpi / 72.0 * self._factor
```

程序运行结果如图 5.4 所示。

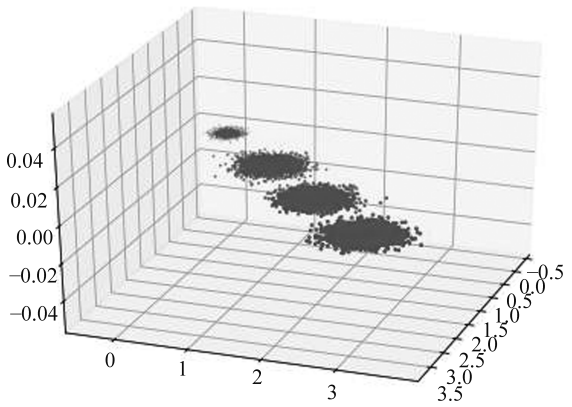

图 5.4 程序运行结果

```
from sklearn.decomposition import PCA
pca=PCA(n_components=3)
pca.fit(X)
print(pca.explained_variance_ratio_)
```

```
print(pca.explained_variance_)
```

【程序运行结果】

```
[0.98318212   0.00850037    0.00831751]
[3.78521638   0.03272613    0.03202212]
```

【程序运行结果分析】

投影后3个特征维度的方差分别为98.3%、0.85%和0.83%,第一个特征占了绝大多数。

下面采用主成分分析进行特征降维,对n_components取值分为如下两种情况:

情况一:n_components取整数。

```
#从三维降到二维,选择前两个特征,抛弃第三个特征
from sklearn.decomposition import PCA
pca=PCA(n_components=2)                                    #减少到两个特征
pca.fit(X)
print(pca.explained_variance_ratio_)
print(pca.explained_variance_)
X_new=pca.transform(X)
plt.scatter(X_new[:, 0], X_new[:, 1],marker='o')   #将转化后的数据分布可视化
plt.show()
```

【程序运行结果】

```
[0.98318212    0.00850037]
[3.78521638    0.03272613]
```

程序运行结果如图5.5所示。

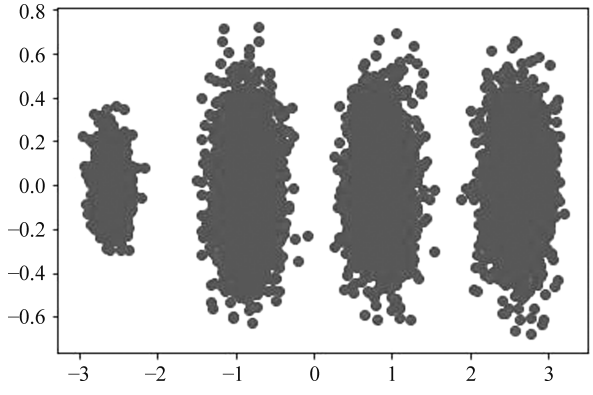

图5.5 程序运行结果

情况二:n_components取小数。

```
pca=PCA(n_components=0.95)                                 #保留 95%的信息
pca.fit(X)
```

```
print(pca.explained_variance_ratio_)
print(pca.explained_variance_)
print(pca.n_components_)
```

【程序运行结果】

```
[0.98318212]
[3.78521638]
1
```

【程序运行结果分析】

只有第一个投影特征被保留。这是由于第一个主成分占投影特征的方差比例高达98.3%，只选择这个特征维度便可以满足 95%的阈值要求。

5.4 特征选择

5.4.1 简介

特征选择，又称变量选择、属性选择或变量子集选择，是选择相关特征子集用于模型构造的过程。简要地说，通过检测相关特征，摒弃冗余特征，获得特征子集，从而以最小的性能损失更好地描述问题。

特征选择和降维都是防止数据过拟合的有效手段。但是两者又有本质上的区别。降维本质上是从一个维度空间映射到另一个维度空间，在映射的过程中特征值会相应地变化。特征选择就是单纯地从提取到的所有特征中选择部分特征作为训练集特征，特征在选择前和选择后不改变值其，但是选择后的特征维数肯定比选择前小。特征选择注重删除无用特征。

sklearn.feature_selection 模块中给出了特征选择的方法，如表 5.1 所示。

表 5.1 特征选择模块方法

方　法	说　明
VarianceThreshold	删除方差小的特征
SelectKBest	返回 K 个最佳特征，移除那些除了评分最高的 K 个特征之外的所有特征
SelectPercentile	按指定百分比返回表现最佳的特征

5.4.2 3 种方法

特征选择的方法有包装法、过滤法和嵌入法等，具体如下：

- 包装法(wrapper)。根据目标函数(通常是预测效果评分)，每次选择若干特征，或者排除若干特征。
- 过滤法(filter)。按照发散性或者相关性对各个特征进行评分，设定阈值或者待选

择阈值的个数，选择特征。

- 嵌入法(embedded)。使用某些机器学习的算法和模型进行训练，得到各个特征的权值系数，根据系数从大到小选择特征。嵌入法类似于过滤法，但是嵌入法通过训练来确定特征的优劣。

5.5 包装法

包装法具有递归特征消除和交叉验证递归特征消除两种方法，具体介绍如下。

5.5.1 递归特征消除

递归特征消除(Recursive Feature Elimination，RFE)是常见的特征选择方法。其工作原理是：递归删除特征，并在剩余的特征上构建模型，使用模型准确率来判断哪些特征(或特征组合)对预测结果贡献较大。

Sklearn 提供了 RFE 函数，以实现递归消除特征法，格式如下：

```
RFE(estimator=svc, n_features_to_select=no_features, step=1)
```

参数解释如下：

- estimator=svc：指定有监督型学习器，该学习器具有 fit 方法，通过 coef_ 属性或者 feature_importances_ 属性来提供 feature 重要性的信息。
- n_features_to_select=no_features：指定保留的特征数。
- step=1：控制每次迭代过程中删去的特征数。

【例 5.3】 递归特征消除示例。

```
from sklearn.feature_selection import RFE
from sklearn.svm import LinearSVC
from sklearn.datasets import load_iris
from  sklearn import model_selection
iris=load_iris()
X, y=iris.data, iris.target
#特征提取
estimator=LinearSVC()
selector=RFE(estimator=estimator, n_features_to_select=2)
X_t=selector.fit_transform(X, y)
#切分训练集和测试集
X_train, X_test, y_train, y_test=model_selection.train_test_split(X, y, test_
       size=0.25, random_state=0, stratify=y)
X_train_t, X_test_t, y_train_t, y_test_t=model_selection.train_test_split(X_
       t, y, test_size=0.25, random_state=0, stratify=y)
#训练和测试
clf=LinearSVC()
```

```
clf_t=LinearSVC()
clf.fit(X_train, y_train)
clf_t.fit(X_train_t, y_train_t)
print("Original DataSet: test score=%s" %(clf.score(X_test, y_test)))
print("Selected DataSet: test score=%s" %(clf_t.score(X_test_t, y_test_t)))
```

【程序运行结果】

```
Original DataSet: test score=0.9736842105263158
Selected DataSet: test score=0.9473684210526315
```

【程序运行结果分析】

原模型的性能在递归特征消除后确实下降了，这是因为递归特征消除自身的原因。虽然该方法可以较好地进行手动特征选择，但是原模型在去除特征后的数据集上的性能表现要差于其在原数据集上的表现，这是因为去除的特征中包含有效信息。

5.5.2 交叉验证递归特性消除

交叉验证递归特性消除（Recursive Feature Elimination with Cross Validation，RFECV）通过交叉验证来找到最优的特征数量，实现特征选择。该方法分如下两个阶段：

（1）RFE 阶段。

进行递归特征消除，对特征进行重要性评级。具体步骤如下：

① 以初始的特征集作为所有可用的特征。

② 使用当前特征集进行建模，然后计算每个特征的重要性。

③ 删除最不重要的一个（或多个）特征，更新特征集。

④ 跳转到步骤②，直到完成所有特征的重要性评级。

（2）CV 阶段

在完成特征评级后，通过交叉验证，选择最佳数量的特征。具体步骤如下：

① 根据 RFE 阶段确定的特征重要性，依次选择不同数量的特征。

② 对选定的特征集进行交叉验证。

③ 确定平均分最高的特征数量，完成特征选择。

RFECV 用于选取单模型特征的效果相当不错，但是它有如下两个缺陷：

（1）计算量大。

（2）随着预估器的改变，最佳特征组合也会改变。

Sklearn 提供了 RFECV 函数，以实现交叉验证递归消除，格式如下：

```
RFECV(estimator=svc, step=1, cv=StratifiedKFold( 2))
```

参数解释如下：

- estimator＝svc：指定用于递归构建模型的有监督型学习器。
- step＝1：控制每次迭代过程中删去的特征个数为 1。
- cv＝StratifiedKFold(2)：指定交叉验证次数。

【例 5.4】 交叉验证递归特征消除示例。

```
import matplotlib.pyplot as plt
from sklearn.model_selection import StratifiedKFold
from sklearn.feature_selection import RFECV
from sklearn.datasets import make_classification
#使用内置函数生成数据集(1000个样本,25个特征,3个有效特征,共8类)
X, y=make_classification(n_samples=1000, n_features=25, n_informative=3,n_
       redundant=2, n_repeated=0, n_classes=8, n_clusters_per_class=1,
       random_state=0)
#创建RFECV,用正确分类比例(accuracy)进行评分
svc=SVC(kernel="linear")
rfecv=RFECV(estimator=svc, step=1, cv=StratifiedKFold(2), scoring='accuracy')
rfecv.fit(X, y)
print("Optimal number of features: %d"%rfecv.n_features_)
#绘制特征数与交叉验证得分关系图
plt.figure
plt.xlabel("Number of features selected")
plt.ylabel("Cross validation score (nb of correct classifications)")
plt.plot(range(1, len(rfecv.grid_scores_)+1), rfecv.grid_scores_)
plt.show
```

【程序运行结果】

```
Optimal number of features: 3
```

程序运行结果如图 5.6 所示。

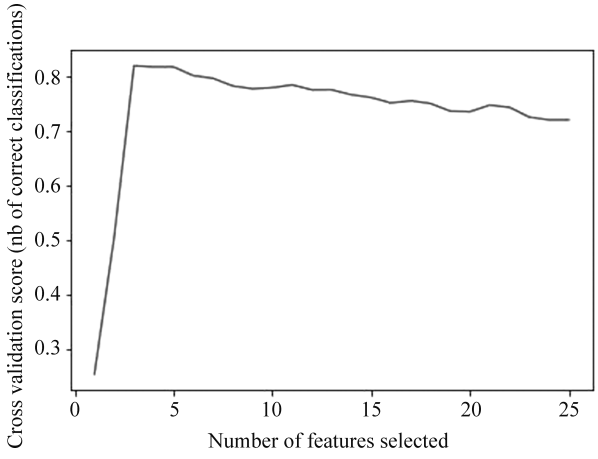

图 5.6 例 5.4 程序运行结果

5.6 过滤法

过滤法有移除低方差特征法和单变量特征选择两种方法。其中,单变量特征选择根据问题类型不同,其消除的指标不同。对于分类问题,采用卡方检验、f_classif 等指标。对于回归问题,采用皮尔森相关系数指标。

5.6.1 移除低方差特征

从方差的大小考虑,特征方差小是指某个特征的大多数样本的值比较相近,特征方差大是指某个特征很多样本的值有比较大的差别。移除低方差特征又称为方差选择法,用于删除低方差的一些特征。

Sklearn 提供 VarianceThreshold 函数实现此功能,其基本语法如下:

```
sklearn.feature_selection.VarianceThreshold(threshold)
```

参数 threshold 为移除方差的阈值。在默认情况下,它取值为 0,表示移除所有方差为 0 的特征,也就是移除所有取值相同的特征,保留所有非零方差特征。

【例 5.5】 移除低方差特征示例。

```
from sklearn.feature_selection import VarianceThreshold
import numpy as np
var=VarianceThreshold(threshold=1.0)              #将方差小于或等于 1.0 的特征删除
data=np.array([[0, 2, 0, 3], [0, 1, 4, 3], [0, 1, 1, 3]])
print("data\n",data)
print("data.shape\n",data.shape)
data_new=var.fit_transform(data)
print("data_new\n",data_new)
print("data_new.shape\n",data_new.shape)
```

【程序运行结果】

```
data
 [[0 2 0 3]
  [0 1 4 3]
  [0 1 1 3]]
data.shape
 (3, 4)
data_new
 [[0]
  [4]
  [1]]
data_new.shape
 (3, 1)
```