LDA

1. 实验目的

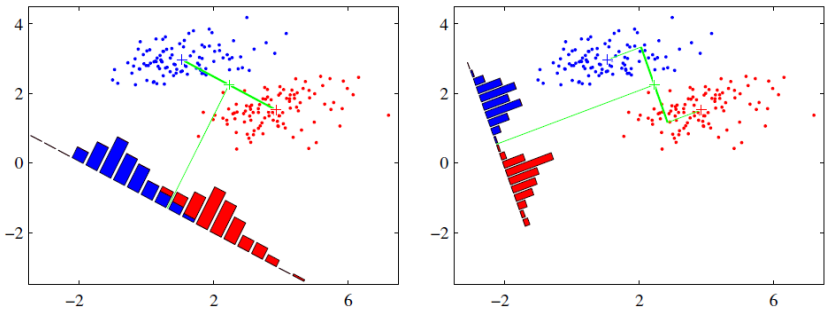
了解 LDA 算法的原理，并且可以简单应用

2. 算法原理

1)LDA 思想

LDA 是一种监督学习的降维技术，也就是说它的数据集的每个样本是有类别输出的。这点和 PCA 不同。PCA 是不考虑样本类别输出的无监督降维技术。LDA 的思想可以用一句话概括，就是“投影后类内方差最小，类间方差最大”。什么意思呢？ 我们要将数据在低维度上进行投影，投影后希望每一种类别数据的投影点尽可能的接近，而不同类别的数据的类别中心之间的距离尽可能的大。

可能还是有点抽象，我们先看看最简单的情况。假设我们有两类数据分别为红色和蓝色，如下图所示，这些数据特征是二维的，我们希望将这些数据投影到一维的一条直线，让每一种类别数据的投影点尽可能的接近，而红色和蓝色数据中心之间的距离尽可能的大。



上图中国提供了两种投影方式，哪一种能更好的满足我们的标准呢？从直观上可以看出，右图要比左图的投影效果好，因为右图的黑色数据和蓝色数据各个较为集中，且类别之间的距离明显。左图则在边界处数据混杂。以上就是 LDA 的主要思想了，当然在实际应用中，我们的数据是多个类别

的，我们的原始数据一般也是超过二维的，投影后的也一般不是直线，而是一个低维的超平面。

在我们将上面直观的内容转化为可以度量的问题之前，我们先了解些必要的数学基础知识，这些在后面讲解具体 LDA 原理时会用到。

2）LDA 算法流程

输入：数据集 $D=(x_1,y_1),(x_2,y_2),...,((x_m,y_m))$, 其中任意样本 x_i 为 n 维向量，$y_i \in C_1,C_2,...,C_k$，降维到的维度 d。

输出：降维后的样本集 D'

1) 计算类内散度矩阵 S_w

2) 计算类间散度矩阵 S_b

3) 计算矩阵 $S_{-1w}S_b$

4）计算 $S_{-1w}S_b$ 的最大的 d 个特征值和对应的 d 个特征向量$(w_1,w_2,...w_d)$, 得到投影矩阵 W

5) 对样本集中的每一个样本特征 x_i, 转化为新的样本 $z_i=W_Tx_i$

6) 得到输出样本集 $D'=(z_1,y_1),(z_2,y_2),...,((z_m,y_m))$

以上就是使用 LDA 进行降维的算法流程。实际上 LDA 除了可以用于降维以外，还可以用于分类。一个常见的 LDA 分类基本思想是假设各个类别的样本数据符合高斯分布，这样利用 LDA 进行投影后，可以利用极大似然估计计算各个类别投影数据的均值和方差，进而得到该类别高斯分布的概率密度

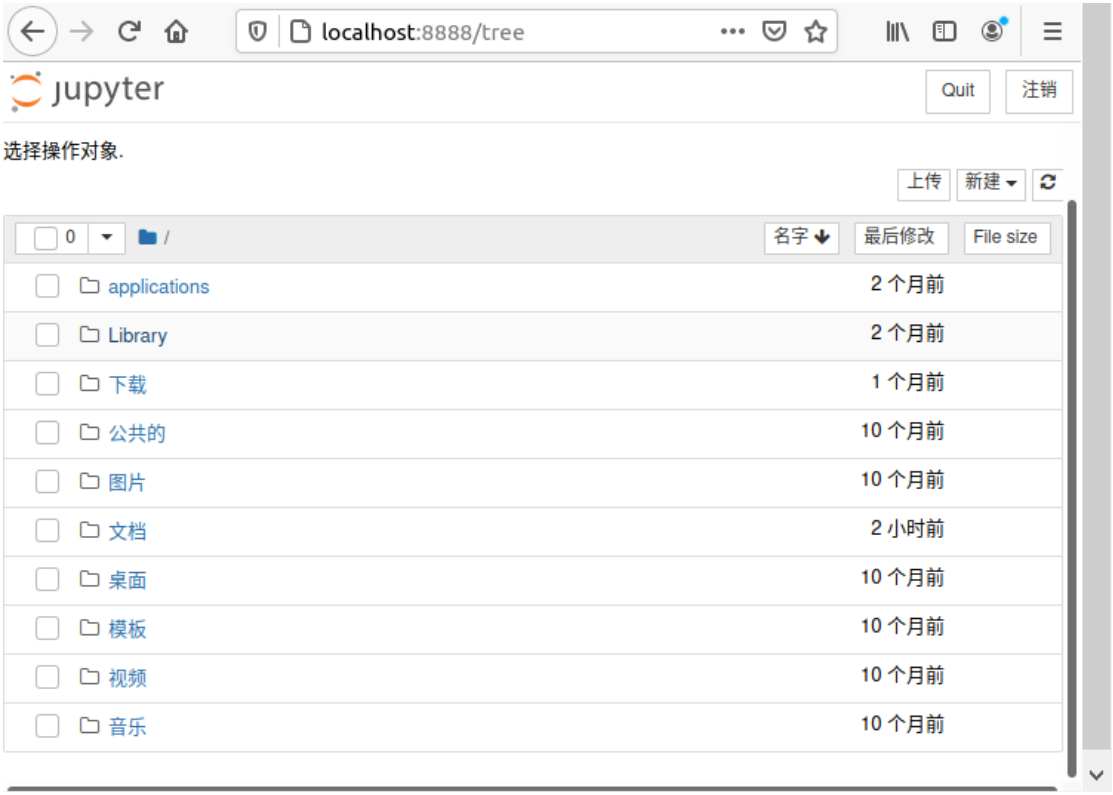
函数。当一个新的样本到来后，我们可以将它投影，然后将投影后的样本特征分别带入各个类别的高斯分布概率密度函数，计算它属于这个类别的概率，最大的概率对应的类别即为预测类别。

3. 实验环境

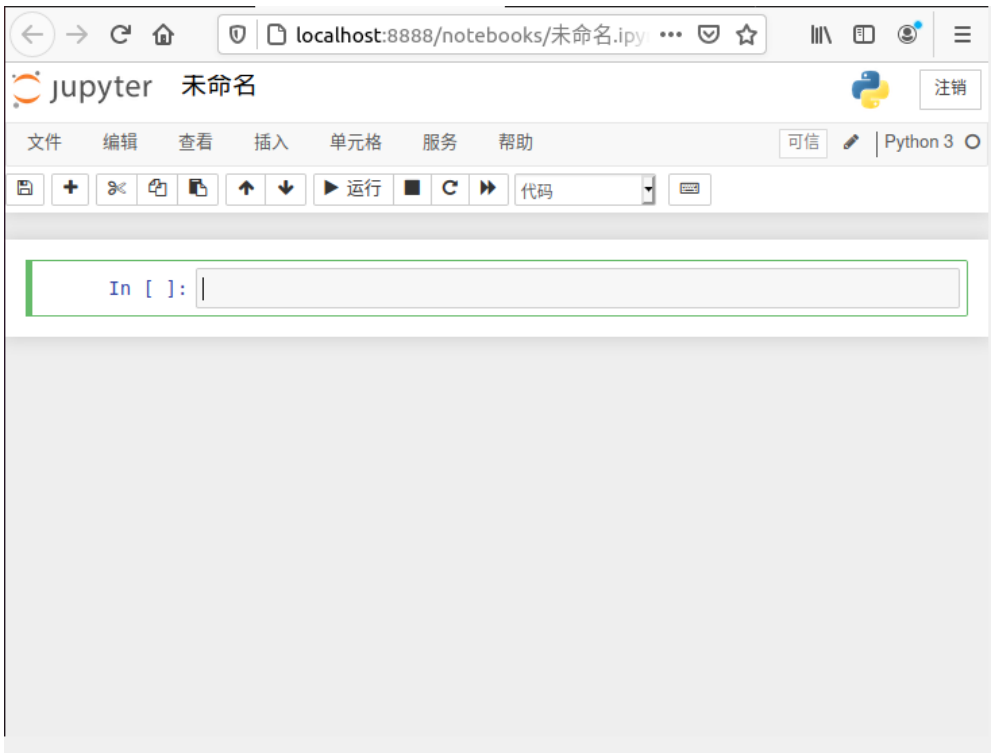
Ubuntu 20.04
Python 3.6
Jupyter notebook

4. 实验步骤

1)打开终端，然后输入 jupyter notebook，出现如下界面



2)选定特定文件夹，新建 ipynb 文件，在未命名出可重命名文件



5. 实操

Step 1:数据预处理

1. 导入库
2. 导入数据集
3. 设置列索引
4. 分割数据集
5. 数据集标准化

```
#导入库
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from matplotlib.colors import ListedColormap
from sklearn.linear_model import LogisticRegression

#导入数据集
df_wine = pd.read_csv('wine.data',header=None)


# 设置列索引
df_wine.columns =  ['Class label', 'Alcohol', 'Malic acid', 'Ash',
                    'Alcalinity of ash', 'Magnesium', 'Total phenols',
                    'Flavanoids', 'Nonflavanoid phenols', 'Proanthocyani
ns',
```

```
                   'Color intensity', 'Hue',
                   'OD280/OD315 of diluted wines', 'Proline']


# 数据集设置：X 为样本特征数据，y 为目标数据，即标注结果
X, y = df_wine.iloc[:, 1:].values, df_wine.iloc[:, 0].values

# 数据集划分： 将数据集划分为训练集和测试集数据（测试集数据为 30%，训练集为
70%）
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3
,
                                                    stratify=y,
                                                    random_state=0)


# 实例化
sc = StandardScaler()

# 对数据集进行标准化（一般情况下我们在训练集中进行均值和方差的计算，直接在测试
集中使用）
X_train_std = sc.fit_transform(X_train)
X_test_std = sc.transform(X_test)
```

Step 2:LDA 模型

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis as
 LDA
# 实例化
lda = LDA(n_components=2)
# 对训练数据进行 LDA 处理
X_train_lda = lda.fit_transform(X_train_std, y_train)
X_train_lda[0]
# 实例化逻辑回归
lr = LogisticRegression()
# 训练
lr = lr.fit(X_train_lda, y_train)
```

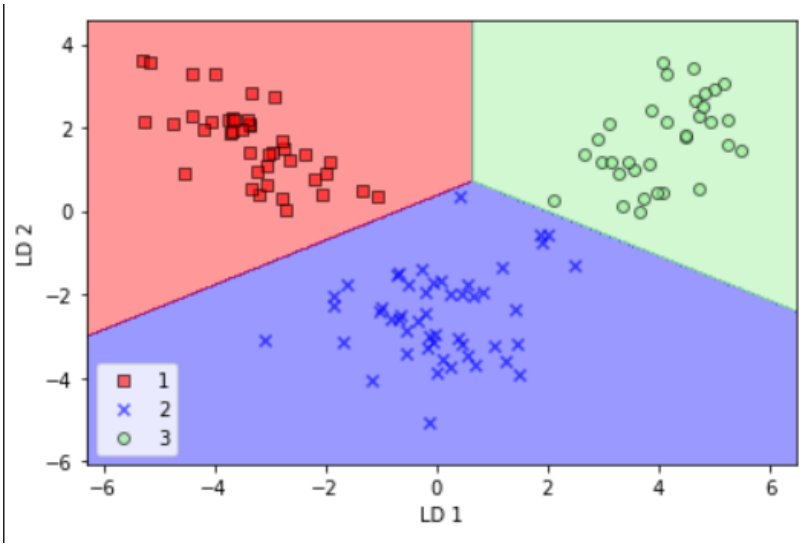
Step 3:样本绘制

```
# 绘制样本及其目标值
def plot_decision_regions(X, y, classifier, resolution=0.02):
    """
    X:样本特征值
    y:目标值
    classifier: 分类器
    """
```

```
    # 设置图像的标记及颜色
    markers = ('s', 'x', 'o', '^', 'v')
    colors = ('red', 'blue', 'lightgreen', 'gray', 'cyan')
    cmap = ListedColormap(colors[:len(np.unique(y))])

    # 利用样本点创建 meshgrid
    x1_min, x1_max = X[:, 0].min() - 1, X[:, 0].max() + 1
    x2_min, x2_max = X[:, 1].min() - 1, X[:, 1].max() + 1
    xx1, xx2 = np.meshgrid(np.arange(x1_min, x1_max, resolution),
                           np.arange(x2_min, x2_max, resolution))
    # 预测结果
    Z = classifier.predict(np.array([xx1.ravel(), xx2.ravel()]).T)
    Z = Z.reshape(xx1.shape)
    # 绘制预测结果的等高线
    plt.contourf(xx1, xx2, Z, alpha=0.4, cmap=cmap)
    plt.xlim(xx1.min(), xx1.max())
    plt.ylim(xx2.min(), xx2.max())

    # 绘制样本点,并根据真实值进行着色
    for idx, cl in enumerate(np.unique(y)):
        # 绘制散点图
        plt.scatter(x=X[y == cl, 0],
                    y=X[y == cl, 1],
                    alpha=0.6,
                    c=cmap(idx),
                    edgecolor='black',
                    marker=markers[idx],
                    label=cl)
# 训练数据结果
plot_decision_regions(X_train_lda, y_train, classifier=lr)
plt.xlabel('LD 1')
plt.ylabel('LD 2')
plt.legend(loc='lower left')
plt.tight_layout()
plt.show()
```



Step 4：测试结果

```
# 测试数据结果
X_test_lda = lda.transform(X_test_std)

plot_decision_regions(X_test_lda, y_test, classifier=lr)
plt.xlabel('LD 1')
plt.ylabel('LD 2')
plt.legend(loc='lower left')
plt.tight_layout()
plt.show()
```