PCA（是一种常见的数据分析方式，常用于高维数据的降维，可用于提取数据的主要特征分量。

PCA的算法步骤：

设有m条n维数据，

1.将原始数据按列组成n行m列矩阵X；

2.将X的每一行进行零均值化，即减去这一行的均值；

3.求出协方差矩阵；

4.求出协方差矩阵的特征值及对应的特征向量；

5.将特征向量按对应特征值大小从上到下按行排列成矩阵；取前k行组成矩阵P;

6.Y=PX即为降维到k维后的数据。

代码如下：

import numpy as np

from sklearn.decomposition import PCA

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import roc\_curve, auc

import os

from PIL import Image

import matplotlib.pyplot as plt

path = 'palmprint\_images' # 掌纹图库所在路径

X = []

y = []

for foldername in os.listdir(path):

class\_path = os.path.join(path, foldername)

for filename in os.listdir(class\_path):

image\_path = os.path.join(class\_path, filename)

image = Image.open(image\_path).convert('L') # converts to grayscale

image = image.resize((64, 64)) # resize the images

image\_data = np.asarray(image).flatten()

X.append(image\_data)

y.append(int(foldername))

X = np.array(X)

y = np.array(y)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

pca = PCA(n\_components=50) # 降为 50 维

X\_train\_pca = pca.fit\_transform(X\_train) # PCA 降维

X\_test\_pca = pca.transform(X\_test)

knn = KNeighborsClassifier(n\_neighbors=5) # k = 5

knn.fit(X\_train\_pca, y\_train) # 训练模型y\_score = knn.predict\_proba(X\_test\_pca)[:,1] # 获取每个测试样例的正例预测概率

fpr, tpr, thresholds = roc\_curve(y\_test, y\_score) # 计算ROC曲线的FPR和TPRroc\_

auc = auc(fpr, tpr) # 计算ROC曲线下面积

plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC Curve (area = %0.2f)' % roc\_auc)

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlabel('FPR')

plt.ylabel('TPR')

plt.title('PCA\_ROC曲线')

plt.legend(loc="lower right")

plt.show()

使用LBP算法来进行掌纹识别，具体步骤如下：

1.将图像划分成m×n个像素块，通常取m=8,n=8。

2.对于每个像素块，根据中心像素与周围8个像素的亮度大小关系，得到一个8位的二进制数，将这个二进制数转化为十进制数作为该像素块的LBP值。

3.将图像划分成不重叠的m×n个像素块，并得到每个像素块的LBP值。

4.对于每个像素块，计算其直方图(bin的数量通常取256)，并将所有直方图连接起来，作为整个图像的LBP特征向量。

5.使用LBP特征向量来训练分类器，并对测试集进行预测。

下面是使用LBP算法对掌纹图像进行特征提取和分类的代码。

import numpy as np

from sklearn.decomposition import PCA

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import roc\_curve, auc

from PIL import Image

import matplotlib.pyplot as plt

import glob

from skimage.feature import local\_binary\_pattern

folder = './palmprint\_images' # 掌纹图库所在文件夹名

image\_paths = glob.glob(folder + '/\*/\*.bmp') # 所有掌纹图像文件的文件路径

# 定义函数：从文件路径中加载LBP特征向量

def load\_LBP\_feature(image\_path, m=8, n=8, hist\_size=256, block\_size=16):

image = Image.open(image\_path).convert('L') # converts to grayscale

image = image.resize((m\*n, m\*n)) # resize the images

lbp = local\_binary\_pattern(image, 8, 1, method='uniform')

max\_bins = int(lbp.max() + 1)

hist, \_ = np.histogram(lbp, bins=max\_bins, range=(0, max\_bins))

block\_hist\_list = []

for i in range(m):

for j in range(n):

start\_x, end\_x = i\*block\_size, (i+1)\*block\_size

start\_y, end\_y = j\*block\_size, (j+1)\*block\_size

block\_hist, \_ = np.histogram(lbp[start\_x:end\_x, start\_y:end\_y], bins=hist\_size, range=(0, hist\_size))

block\_hist\_list.append(block\_hist)

feature = np.hstack(block\_hist\_list)

return feature

# 加载掌纹图像数据和标签

X = []

y = []

for image\_path in image\_paths:

class\_name = image\_path.split('/')[-2] # 类别名称

feature = load\_LBP\_feature(image\_path) # 加载LBP特征向量

X.append(feature)

y.append(int(class\_name))

X = np.array(X)

y = np.array(y)

# 将数据分为训练集和测试集

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# 使用PCA降维

pca = PCA(n\_components=50) # 降为 50 维

X\_train\_pca = pca.fit\_transform(X\_train) # PCA 降维

X\_test\_pca = pca.transform(X\_test)

# 构建kNN分类器，k=5

knn = KNeighborsClassifier(n\_neighbors=5)

knn.fit(X\_train\_pca, y\_train)

# 根据测试集样本的预测概率作为得分，计算模型的ROC曲线

y\_score = knn.predict\_proba(X\_test\_pca)[:, 1]

fpr, tpr, thresholds = roc\_curve(y\_test, y\_score)

roc\_auc = auc(fpr, tpr)

# 绘制ROC曲线

plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC Curve (area = %0.2f)' % roc\_auc)

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve')

plt.legend(loc="lower right")

plt.show()

使用Gabor算法对掌纹图像进行特征提取和分类

import numpy as np

from sklearn.decomposition import PCA

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import roc\_curve, auc

from PIL import Image

import matplotlib.pyplot as plt

import glob

from skimage.filters import gabor

from scipy import ndimage as ndi

folder = './palmprint\_images' # 掌纹图库所在文件夹名

image\_paths = glob.glob(folder + '/\*/\*.bmp') # 所有掌纹图像文件的文件路径

# 定义函数：从文件路径中加载Gabor特征向量

def load\_Gabor\_feature(image\_path, m=8, n=8, n\_directions=8, n\_levels=5):

image = Image.open(image\_path).convert('L') # converts to grayscale

image = image.resize((m\*n, m\*n)) # resize the images

feature\_list = []

for theta in np.linspace(0, np.pi, n\_directions+1)[:-1]:

for frequency in range(n\_levels):

filt\_real, filt\_imag = gabor(image, frequency=frequency+1, theta=theta, n\_stds=3)

feature = np.hstack((ndi.mean(filt\_real, np.ones((3, 3))), ndi.variance(filt\_real, np.ones((3, 3))))) # 提取均值和方差作为特征

feature\_list.append(feature)

feature = np.hstack(feature\_list)

return feature

# 加载掌纹图像数据和标签

X = []

y = []

for image\_path in image\_paths:

class\_name = image\_path.split('/')[-2] # 类别名称

feature = load\_Gabor\_feature(image\_path) # 加载Gabor特征向量

X.append(feature)

y.append(int(class\_name))

X = np.array(X)

y = np.array(y)

# 将数据分为训练集和测试集

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# 使用PCA降维

pca = PCA(n\_components=50) # 降为 50 维

X\_train\_pca = pca.fit\_transform(X\_train) # PCA 降维

X\_test\_pca = pca.transform(X\_test)

# 构建kNN分类器，k=5

knn = KNeighborsClassifier(n\_neighbors=5)

knn.fit(X\_train\_pca, y\_train)

# 根据测试集样本的预测概率作为得分，计算模型的ROC曲线

y\_score = knn.predict\_proba(X\_test\_pca)[:, 1]

fpr, tpr, thresholds = roc\_curve(y\_test, y\_score)

roc\_auc = auc(fpr, tpr)

# 绘制ROC曲线

plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC Curve (area = %0.2f)' % roc\_auc)

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve')

plt.legend(loc="lower right")

plt.show()

SIFT算法

SIFT即尺度不变特征变换，是用于图像处理领域的一种描述。这种描述具有尺度不变性，可在图像中检测出关键点，是一种局部特征描述子。

SIFT算法特点

1、具有较好的稳定性和不变性，能够适应旋转、尺度缩放、亮度的变化，能在一定程度上不受视角变化、仿射变换、噪声的干扰。

2、区分性好，能够在海量特征数据库中进行快速准确的区分信息进行匹配

3、多量性，就算只有单个物体，也能产生大量特征向量

4、高速性，能够快速的进行特征向量匹配

5、可扩展性，能够与其它形式的特征向量进行联合

使用SIFT算法对掌纹图像进行特征提取和分类

import numpy as np

from sklearn.decomposition import PCA

from sklearn.model\_selection import train\_test\_split

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import roc\_curve, auc

from PIL import Image

import matplotlib.pyplot as plt

import glob

import cv2

folder = './palmprint\_images' # 掌纹图库所在文件夹名

image\_paths = glob.glob(folder + '/\*/\*.bmp') # 所有掌纹图像文件的文件路径

# 定义函数：从文件路径中提取SIFT特征

def extract\_SIFT\_feature(image\_path):

image = cv2.imread(image\_path)

gray = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

sift = cv2.xfeatures2d.SIFT\_create()

keypoints, descriptors = sift.detectAndCompute(gray, None)

return descriptors.flatten()

# 加载掌纹图像数据和标签

X = []

y = []

for image\_path in image\_paths:

class\_name = image\_path.split('/')[-2] # 类别名称

feature = extract\_SIFT\_feature(image\_path) # 提取SIFT特征向量

X.append(feature)

y.append(int(class\_name))

X = np.array(X)

y = np.array(y)

# 将数据分为训练集和测试集

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# 使用PCA降维

pca = PCA(n\_components=50) # 降为 50 维

X\_train\_pca = pca.fit\_transform(X\_train) # PCA 降维

X\_test\_pca = pca.transform(X\_test)

# 构建kNN分类器，k=5

knn = KNeighborsClassifier(n\_neighbors=5)

knn.fit(X\_train\_pca, y\_train)

# 根据测试集样本的预测概率作为得分，计算模型的ROC曲线

y\_score = knn.predict\_proba(X\_test\_pca)[:, 1]

fpr, tpr, thresholds = roc\_curve(y\_test, y\_score)

roc\_auc = auc(fpr, tpr)

# 绘制ROC曲线

plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC Curve (area = %0.2f)' % roc\_auc)

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve')

plt.legend(loc="lower right")

plt.show()