Python 实现 PCA 降维和 KNN 人脸识别模型(PCA 和 KNeighborsClassifier 算法)项目实战

1.项目背景

人脸识别是基于人的脸部特征信息进行身份识别的一种生物识别技术。该技术蓬勃发展，应用广泛，如人脸识别门禁系统、刷脸支付软件等。

人脸识别在本质上是根据每张人脸图像中不同像素点的颜色进行数据建模与判断。人脸图像的每个像素点的颜色都有不同的值，这些值可以组成人脸的特征向量，不过因为人脸图像的像素点很多，所以特征变量也很多，需要利用 PCA 进行数据降维。

本项目先对人脸数据进行读取和处理，再通过 PCA 进行数据降维，最后用 K 近邻算法搭建模型进行人脸识别。

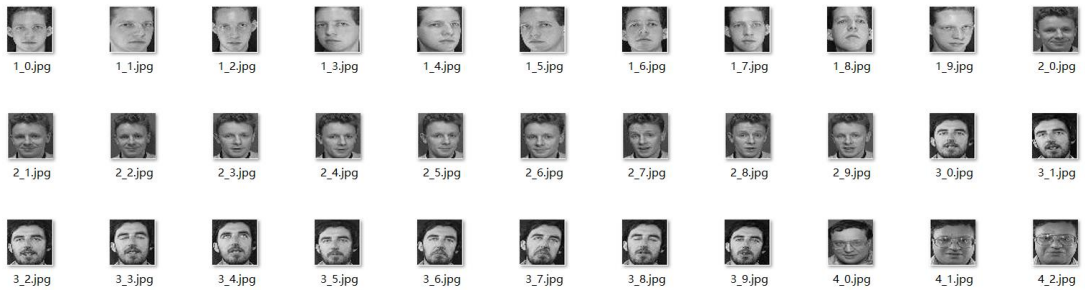
2.数据获取

本次建模数据来源于网络(本项目撰写人整理而成)，，数据项统计如下：
数据集中图片的文件名由 4 部分组成：

- 第 1 部分是该张图片对应的人脸编号；
- 第 2 部分是固定分隔符“_”；
- 第 3 部分是该张图片在该人脸 10 张图片中的顺序编号；
- 第 4 部分是文件扩展名“.jpg”。

以“10_0.jpg”为例，10 代表编号为 10 的人的图片，“_”是第 1 部分和第 3 部分的分隔符，0 代表这个人的 10 张图片中编号为 0 的那一张，“.jpg”为文件扩展名。

数据详情如下(部分展示)：



3.数据预处理

3.1 图片数据读取

使用 os 模块列出前 5 个图片的名称：

```
['10_0.jpg', '10_1.jpg', '10_2.jpg', '10_3.jpg', '10_4.jpg']
```

从上图可以看到，总共有 9 个字段。

关键代码：

```
names = os.listdir('data')  # 返回指定文件夹下的文件或文件夹的名称列表
print(names[0:5])  # 输出前5个文件的名称
```

3.2 特征变量提取

使用 Image. convert('L')方法进行特征变量提取：

```
**************************输出图片灰度值****************************
[[186  76  73 ... 100 103 106]
 [196  85  68 ...  85 106 103]
 [193  69  79 ...  82  99 100]
 ...
 [196  87 193 ... 103  66  52]
 [219 179 202 ... 150 127 109]
 [244 228 230 ... 198 202 206]]
```

关键代码：

```
# 对上面读取的图片img0进行灰度转换，参数'L'指转换成灰度格式的图像。在进行灰度处理后，图像的每个像素点的颜色就可以用0～255的数值表示，称为灰度值，
# 其中0代表黑色，255代表白色，（0，255）区间的数值则代表不同程度的灰色。这样便完成了将图像转换成数字的第一步，也是非常重要的一步。
img0 = img0.convert('L')
```

3.3 图片灰度值数据框显示

使用 Pandas 工具的 DataFrame()方法进行转换：

```
**********输出DataFrame数据框图片灰度值**********
[[186  76  73 ... 198 202 206]]
**********输出列表图片灰度值**********
[186, 76, 73, 87, 89, 88, 75, 81, 100, 102, 105, 92, 74, 65, 65, 53, 43, 55, 53, 42, 58, 77, 71, 75, 75, 73, 76, 85, 95,
 66, 52, 59, 70, 85, 62, 82, 89, 85, 106, 103, 193, 69, 79, 92, 105, 102, 112, 117, 106, 94, 91, 112, 101, 87, 75, 61, 5
 109, 98, 101, 86, 68, 74, 65, 58, 53, 51, 52, 42, 40, 42, 43, 40, 52, 41, 61, 69, 76, 76, 108, 179, 46, 41, 50, 53, 69,
 101, 173, 33, 43, 49, 48, 53, 64, 69, 72, 75, 82, 84, 84, 82, 72, 75, 69, 71, 67, 56, 58, 55, 38, 36, 33, 32, 39, 45, 6
 31, 32, 35, 32, 35, 49, 65, 64, 53, 87, 171, 24, 32, 36, 42, 55, 77, 101, 107, 102, 98, 83, 71, 64, 44, 48, 54, 64, 76,
  58, 65, 83, 90, 97, 108, 101, 97, 105, 105, 101, 89, 63, 45, 42, 41, 37, 61, 101, 172, 21, 22, 27, 28, 30, 33, 43, 46,
 171, 23, 30, 21, 30, 36, 44, 51, 46, 41, 45, 52, 61, 70, 84, 101, 112, 119, 126, 131, 134, 138, 136, 126, 120, 102, 70,
 143, 141, 142, 142, 133, 122, 114, 111, 77, 31, 36, 95, 104, 192, 42, 27, 37, 63, 87, 99, 111, 116, 116, 117, 117, 121,
  58, 102, 112, 110, 110, 107, 108, 107, 107, 115, 126, 126, 118, 98, 87, 69, 57, 55, 55, 70, 71, 86, 102, 107, 116, 84,
 62, 73, 73, 101, 116, 116, 99, 80, 88, 105, 190, 88, 102, 114, 99, 76, 55, 55, 50, 37, 60, 53, 49, 84, 161, 155, 109, 9
 109, 103, 96, 84, 84, 99, 123, 172, 176, 139, 130, 127, 120, 116, 115, 134, 137, 142, 146, 134, 120, 96, 115, 122, 93, 1
 164, 163, 158, 156, 142, 124, 110, 100, 114, 108, 86, 193, 83, 140, 130, 122, 141, 153, 160, 168, 177, 171, 154, 135, 13
 127, 107, 120, 134, 146, 163, 166, 155, 134, 138, 137, 153, 164, 141, 132, 132, 127, 148, 156, 157, 145, 132, 117, 105,
 75, 106, 142, 146, 142, 129, 116, 109, 105, 102, 86, 55, 60, 63, 194, 78, 87, 91, 92, 108, 113, 122, 127, 140, 148, 154
  88, 90, 104, 114, 120, 131, 141, 145, 150, 153, 142, 144, 124, 105, 111, 119, 121, 128, 128, 130, 129, 119, 108, 103, 1
 87, 86, 76, 88, 102, 113, 111, 107, 101, 85, 53, 51, 54, 55, 190, 86, 88, 87, 87, 87, 104, 115, 127, 115, 101, 88, 83, 8
  130, 103, 96, 110, 108, 108, 129, 126, 112, 119, 108, 96, 98, 105, 110, 117, 118, 111, 105, 103, 102, 83, 50, 49, 56, 5
  108, 102, 104, 99, 100, 111, 111, 57, 48, 52, 53, 192, 78, 83, 173, 211, 158, 114, 100, 87, 94, 108, 114, 115, 119, 12
  108, 101, 119, 86, 81, 94, 105, 115, 123, 127, 128, 126, 122, 119, 109, 97, 97, 98, 98, 93, 82, 80, 123, 145, 73, 43, 5
 90, 96, 77, 53, 160, 124, 103, 66, 52, 219, 179, 202, 196, 198, 146, 122, 118, 119, 94, 76, 73, 72, 74, 77, 73, 74, 79,
 193, 189, 181, 173, 171, 171, 171, 168, 168, 171, 173, 178, 182, 179, 179, 184, 177, 161, 202, 182, 207, 198, 202, 206]
```

3.4 批量处理图片

通过 for 循环批量处理图片：

**********输出DataFrame数据框所有图片灰度值**********

	0	1	2	3	4	5	...	1018	1019	1020	1021	1022	1023
0	186	76	73	87	89	88	...	202	182	207	198	202	206
1	196	90	97	98	98	87	...	203	209	205	198	190	190
2	193	89	97	99	75	74	...	195	201	206	201	189	190
3	192	84	93	89	97	89	...	203	200	196	186	182	184
4	194	72	49	45	56	37	...	174	224	200	218	176	168
..	...	...	...	...	...	...	...	...	...	...	...	...	...
395	114	115	115	119	115	120	...	142	144	143	141	143	215
396	115	118	117	117	116	118	...	189	193	148	144	142	212
397	113	116	113	117	114	121	...	203	192	144	143	137	212
398	110	109	109	110	110	112	...	175	136	142	141	137	213
399	105	107	111	112	113	113	...	140	139	142	141	138	213

[400 rows x 1024 columns]

**********输出数据集形状**********

(400, 1024)

10

3.5 目标变量提取

通过 Image 模块的 open()方法读取目标变量：

```
**********目标变量列表**********
[10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 11, 11, 11, 11, 11, 11, 11, 11, 11, 12, 12, 12, 12, 12, 12,
 15, 15, 15, 15, 16, 16, 16, 16, 16, 16, 16, 16, 16, 16, 17, 17, 17, 17, 17, 17, 17, 17, 17, 18, 18,
 20, 20, 20, 20, 20, 20, 21, 21, 21, 21, 21, 21, 21, 21, 21, 22, 22, 22, 22, 22, 22, 22, 22, 22, 22,
 26, 26, 26, 26, 26, 26, 26, 26, 26, 26, 27, 27, 27, 27, 27, 27, 27, 27, 27, 27, 28, 28, 28, 28, 28, 28,
 30, 30, 31, 31, 31, 31, 31, 31, 31, 31, 31, 32, 32, 32, 32, 32, 32, 32, 32, 32, 32, 33, 33, 33, 33,
 36, 36, 36, 36, 36, 36, 37, 37, 37, 37, 37, 37, 37, 37, 37, 37, 38, 38, 38, 38, 38, 38, 38, 38, 38, 38,
 4, 4, 4, 4, 4, 4, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 7, 7, 7,
```

关键代码：

```
img = Image.open('data\\' + i)  # 读取图片
y.append(int(i.split('_')[0]))  # 目标变量放入列表
```

4.探索性数据分析

4.1 显示第一张图片

用 Image 工具的 open()方法进行进行图片的显示：

5.特征工程

5.1 数据集拆分

数据集集拆分，分为训练集和测试集，80%训练集和 20%测试集。关键代码如下：

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)  # 进行数据集拆分
```

5.2 PCA 数据降维

使用 PCA 算法进行数据的降维，输出如下：

```
**********降维后的训练集形状**********
(320, 100)
**********降维后的测试集形状**********
(80, 100)
**********降维后的训练集前5行**********
            0           1           2   ...          97         98         99
0  1252.466959 -117.842516 -267.052983  ...   25.791113  49.816055  24.194365
1  -266.338613 -292.382022 -811.580229  ...  109.151090 -73.042498  -6.016991
2  -413.779989   96.315739  135.006098  ...   58.807184 -77.127941   5.245344
3  -194.824966  664.306884 -221.828129  ...   21.030591  11.792463  -0.868624
4  -846.141625  360.149630 -354.495056  ...  -10.299936  13.745622  23.194234

[5 rows x 100 columns]
**********降维后的测试集前5行**********
           0           1           2   ...         97         98         99
0 -649.817957  248.454477  164.494292  ... -11.517392  30.085427 -30.725577
1 -244.490266 -871.312328  310.576836  ... -12.492039  33.506193 -15.305980
2 -231.897730   45.380042 -579.914452  ...  11.808633  15.166789 -27.313308
3  787.103313   44.812164  286.764339  ...  10.903108 -47.256499 -23.731855
4 -802.880228 -131.568139  380.722274  ... -24.456686 -32.818183  20.334288

[5 rows x 100 columns]
```

关键代码如下：

```
print('**********降维后的训练集形状**********')
print(X_train_pca.shape)
print('**********降维后的测试集形状**********')
print(X_test_pca.shape)

print('**********降维后的训练集前5行**********')
print(pd.DataFrame(X_train_pca).head())
print('**********降维后的测试集前5行**********')
print(pd.DataFrame(X_test_pca).head())
```

6. 构建人脸识别模型

主要使用 KNeighborsClassifier 算法，用于目标分类。

6.1 模型构建

编号	模型名称	参数
1	KNN 人脸识别模型	n_neighbors=5(默认参数值)
2		weights=’ uniform’

7.模型评估

7.1 评估指标及结果

评估指标主要包括准确率、查准率、召回率、F1 分值等等。

模型名称	指标名称	指标值
测试集		
KNN 人脸识别模型	准确率	0.8875
	查准率	0.9012
	召回率	0.8875
	F1 分值	0.8753

从上表可以看出，人脸识别模型效果良好。

关键代码如下：

```
# 模型评估
from sklearn.metrics import precision_score, recall_score, f1_score, accuracy_score  # 导入模型评估工具
```

7.2 查看是否过拟合

查看训练集和测试集的分数：

```
训练集score: 0.9688
测试集score: 0.8875
```

通过结果可以看到，训练集分数和测试集分数基本相当，所以没有出现过拟合现象。

关键代码：

```
print('训练集score: {:.4f}'.format(knn.score(X_train_pca, y_train)))
print('测试集score: {:.4f}'.format(knn.score(X_test_pca, y_test)))
```

7.3 分类报告

人脸识别模型分类报告：

	precision	recall	f1-score	support
1	1.00	0.50	0.67	2
2	1.00	1.00	1.00	1
3	1.00	1.00	1.00	3
4	1.00	1.00	1.00	1
5	1.00	1.00	1.00	1
6	1.00	1.00	1.00	2
7	1.00	1.00	1.00	2
8	1.00	1.00	1.00	4
9	1.00	1.00	1.00	4
10	1.00	1.00	1.00	3
11	1.00	1.00	1.00	1
12	1.00	1.00	1.00	2
13	1.00	1.00	1.00	4
14	1.00	0.33	0.50	3
15	1.00	1.00	1.00	3
17	1.00	0.67	0.80	6
18	0.40	1.00	0.57	2
19	0.50	1.00	0.67	2
20	1.00	1.00	1.00	3
21	1.00	1.00	1.00	2
22	1.00	1.00	1.00	1
23	1.00	1.00	1.00	3
24	1.00	1.00	1.00	2
25	0.00	0.00	0.00	0
26	1.00	1.00	1.00	3
27	1.00	1.00	1.00	1
28	1.00	1.00	1.00	1

从上图可以看出，分类为 1 的 F1 分值为 0.67；分类为 2 的 F1 分值为 1.00，其它类型的以此类推。

8.结论与展望

综上所述，本项目采用了 PCA 数据降维和 KNN 分类模型，最终证明了我们提出的模型效果良好。