

---

# 天津大学

## 模式识别与深度学习课程

### 实验 2、PCA 降维 SVM 分类算法实验报告



学    院 智能与计算学部  
专    业 计算机科学与技术  
学    号 3019244266  
姓    名 李润泽

---

## 1. 实验目标

实验的总目标是帮助我们掌握 PCA 降维算法原理及代码实现，对已有算法灵活调用获取降维重建结果。根据实验数据的特性调节 PCA 降维数以及 SVM 线性函数或核函数。其中两个小实验的目标分别是：

第一部分，将 PCA 人脸降维重建算法理解，更改给出的重建代码，实现对本班级采集人脸数据进行 PCA 降维重建实验，并调试参数 `n_components` 的不同取值，查看不同取值对实验结果的影响，将结果记录分析。

第二部分，补全 `pca_svm.py` 代码，使用本班级人脸采集数据进行 PCA 降维，然后使用降维算法后的数据进行 SVM 分类实验。调试不同参数，查看不同参数对实验结果的影响，将不同参数对应的训练集和测试集准确率通过表格记录，并对结果进行分析。

## 2. 实验一

### 2.1 算法实现及参数调节说明

该实验应用 PCA 降维模型对灰度化处理后的图片进行降维，并利用 PCA 重建模型和降维后的主成分向量进行人脸的重建。

随着维度的增长，点间的距离增大。维度越大，过拟合的风险越高。除此以外，空间的复杂度也会增大；在每个训练实例涉及到成千上万的特征时，机器学习相关的训练会耗时巨大，并且不容易获得优解，导致维度灾难，采用诸如 PCA（映射）或 LLE（流形学习）这样的算法可以对高维数据维数约简、降维处理；PCA 定义一个超平面并将数据映射到上面。PCA 可能会通过找到保存最多训练变量的轴或是最小化卷方差距的轴，选取为需要引射到的轴。这些轴就是

---

Principal Component (PC); 通过 SVD (Singular Value Decomposition) 来将训练集矩阵  $X$  分解为三个矩阵乘积  $U \Sigma V^T$ 。其中  $V$  是主成分矩阵。

$$V = \begin{pmatrix} | & | & & | \\ \mathbf{c}_1 & \mathbf{c}_2 & \cdots & \mathbf{c}_n \\ | & | & & | \end{pmatrix}$$

PCA 用来压缩 (映射到  $d$  维度), 解压可以有如下的等式:

$$X_{d-proj} = X W_d$$

$$X_{recovered} = X_{d-proj} W_d^T = X W_d W_d^T$$

图像重建也是基于这样的思想。

算法的实现如下所示:

#### 1. 数据导入、预处理以及灰度化:

```
def dataload(path):  
    all_image = [] # list, 存储所有图像  
    re_path = [] # list, 存储图像路径  
  
    for face_path_name in os.listdir(path):  
        face_path = os.path.join(path, face_path_name)  
        for image_path_name in os.listdir(face_path):  
            image_path = os.path.join(face_path, image_path_name)  
            # re_path.append(os.path.join(path.split('_')[0]+'_recon', face_path_name, image_path_name))  
            re_path.append(os.path.join(path + '_recon', face_path_name, image_path_name))  
            '''  
            if not os.path.exists(os.path.join(path.split('_')[0]+'_recon', face_path_name)):  
                os.makedirs(os.path.join(path.split('_')[0]+'_recon', face_path_name))  
            '''  
            if not os.path.exists(os.path.join(path + '_recon', face_path_name)):  
                os.makedirs(os.path.join(path + '_recon', face_path_name))  
            img_gray = Image.open(image_path).convert('L')  
            # img_gray.save(os.path.join(os.path.join(path.split('_')[0]+'_recon', face_path_name, image_path_name)))  
            img_gray.save(os.path.join(os.path.join(path + '_recon', face_path_name, image_path_name)))  
            img_np = np.array(img_gray)  
            all_image.append(img_np)  
    all_image = np.array(all_image) # (98, 250, 250)  
    all_image_flatten = all_image.reshape((all_image.shape[0], -1)) # (98, 62500)  
    return all_image_flatten, re_path
```

#### 2. PCA 降维以及图片重建:

```
def PCA_recon(all_image_flatten, re_path, components_value):  
    model = PCA(n_components=components_value) # n_components不能超过 min(n_samples, n_features)  
    components = model.fit_transform(all_image_flatten)  
    face_recon = model.inverse_transform(components)
```

### 3. 数据灰度化图像保存:














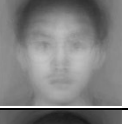

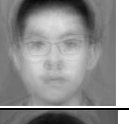
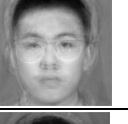
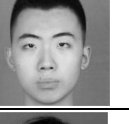






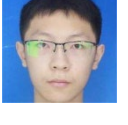




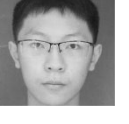
```
for i in range(face_recon.shape[0]):
    each_face_recon = face_recon[i]
    each_face_recon = Image.fromarray(each_face_recon.reshape((250,250)))
    each_face_recon = each_face_recon.convert('L')
    each_face_recon.save(re_path[i].split('.')[0]+'-recon-'+str(components_value)+'.jpg')
```

### 4. 使用多组 n\_component 作为传参比较实验。

```
components_value_list = [1, 5, 10, 30, 50]
for components_value in components_value_list:
    PCA_recon(all_image_flatten, re_path, components_value)
```

## 2.2 结果分析

实验结果如下表所示:

Name	Origin	n_component				
		1	5	10	30	50
chenjun						
lijiaxing						
lirunze						
liuchenghao						
liuhanxi						

由表格数据, 可知使用的 n\_components 越大, 在 PCA 映射的时候维度就越多, 相当于在压缩数据的时候, 丢失的信息就越少, 重建后表示的维度就越高, 恢复后可表示的信息也越多, 和原图相比就越相似。这里可以看出, 降维不是完全可逆的。其次, 对比不同的同学, 可以看出来背景色与肤色反差度大的同学面貌经 PCA 映射的时候, 信息重建时更完整。而本身图像不够清晰或者饱和度比较弱的图像自然很容易就丢失掉特有的信息, PCA 重建的时候会更困难。

## 3. 实验二

### 3.1 算法实现说明

这一部分在 sklearn 库的基础上实现较为简单。完善好 PCA 和 SVM 两个函数，把降维的样例用 SVM 分类。

数据集说明：

1、训练集样本数  $\text{train\_num} = 714$

2、数据集增强：

其中，每位同学的人脸数据进行了 20 倍数据增广，包括随机旋转，上下左右翻转，加噪声，随机剪裁；划分了训练集和测试集，每位同学对应的训练集包含 14 个样本，测试集包含 7 个样本。

3、数据集样例：



### 3.2 结果分析

训练和测试结果如下表所示

component_value	C		0.01	0.1	1	10	100
	kernel						
1	poly	trainAcc	0.048	0.055	0.067	0.066	0.066
		testAcc	0	0	0	0	0
	rbf	trainAcc	0.12	0.12	0.113	0.133	0.148
		testAcc	0	0	0	0	0
	sigmoid	trainAcc	0.052	0.055	0.036	0.08	0.083
		testAcc	0	0	0	0	0
5	poly	trainAcc	0.143	0.204	0.52	0.807	0.957
		testAcc	0	0.003	0.105	0.291	0.373
	rbf	trainAcc	0.391	0.391	0.52	0.864	0.986
		testAcc	0	0	0.052	0.333	0.428

	sigmoid	trainAcc	0.139	0.147	0.23	0.259	0.237
		testAcc	0	0	0.007	0.082	0.078
10	poly	trainAcc	0.183	0.216	0.63	0.931	0.997
		testAcc	0	0	0.098	0.304	0.435
	rbf	trainAcc	0.444	0.444	0.662	0.975	1
		testAcc	0	0	0.075	0.412	0.467
	sigmoid	trainAcc	0.189	0.188	0.339	0.364	0.375
		testAcc	0	0	0.013	0.114	0.108
30	linear	trainAcc	1	1	1	1	1
		testAcc	0.49	0.49	0.49	0.49	0.49
	poly	trainAcc	0.244	0.28	0.721	0.962	1
		testAcc	0	0	0.101	0.366	0.464
	rbf	trainAcc	0.552	0.552	0.775	0.997	1
		testAcc	0	0	0.098	0.5	0.503
50	sigmoid	trainAcc	0.256	0.256	0.476	0.608	0.664
		testAcc	0	0	0.016	0.199	0.199
	linear	trainAcc	1	1	1	1	1
		testAcc	0.477	0.477	0.477	0.477	0.477
	poly	trainAcc	0.261	0.296	0.741	0.972	1
		testAcc	0	0	0.105	0.363	0.467
	rbf	trainAcc	0.57	0.57	0.804	1	1
		testAcc	0	0	0.105	0.484	0.484
	sigmoid	trainAcc	0.282	0.283	0.5	0.678	0.749
		testAcc	0	0	0.016	0.265	0.261

根据上表可以得到如下总结：

1、人脸图像信息在被 PCA 降维后会丢失信息，但是实际上对后续 SVM 分类的结果影响有限，甚至在出现降维后反而使得分类结果提升的个别情况，比如  $\text{Prediction Acc (n\_components=40, kernel=poly)} > \text{Prediction Acc (n\_components=50, kernel=poly)}$ ，可能是因为又一些特征值冗余，影响了实验结果。

2、在比较内核后发现，即使 PCA 对数据降维，Linear 内核的 SVM 分类结果受到的影响较小，但是耗时巨大，可能是因为模型过于简单，或者实验数据线性不可分。今后可能会对这个现象有更深刻的认识。但是，可以从 Acc 的表现和 n\_components 两者之间的关系看出，sigmoid 受到降维影响最大。整体来看，在这次试验中，SVM 的结果 rbf 优于 poly，更优于 sigmoid。

3、对比了从主成分数量的角度和从保留变量比例的角度来看 SVM 的表现，变化趋势的区别不大，当然这也是和主成分数量和保留训练变量比例之间的关系有关的。

---

## 4. 总结

实验除了应用到 SVM 分类算法以外，加入了 PCA 主成分降维算法，以及对降维后的数据进行 SVM 的分类实际应用。

在这一实验中，深入学习了使用 LFW 数据集进行 PCA 降维重建的工程案例，并迁移学习，自己动手对班级内同学们的人脸图像进行 PCA 降维重建，以及运用之前所学的 SVM 支持向量机算法对降维后的 SVM 分类。在调节变量的过程中，也感受到了 PCA 降维算法对后续 SVM 分类加速的不错的效果。

通过这次实验，我对 SVM 和 PCA 模型有了更多的理解。在我对算法的代码进行理解分析之后，成功以此进行了第二部分代码并得以顺利运行，从而提升了我的代码能力。针对老师在实验指导书上的各种问题我也进行了分析，最终顺利完成了本次实验，对于我在学习模式识别和深度学习的过程中会有更大的帮助。