哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

2020 年春季学期

计算学部《机器学习》课程

Lab 4 实验报告

姓名	王科龙
学号	1180801203
班号	1803104
电子邮件	1264405807@qq.com
手机号码	19917620613

目录

1 问题描述

1.1 问题

目标：实现一个 PCA 模型，能够对给定数据进行降维（即找到其中的主成分）

测试：（1）首先人工生成一些数据（如三维数据），让它们主要分布在低维空间中，如首先让某个维度的方差远小于其它唯独，然后对这些数据旋转。生成这些数据后，用你的 PCA 方法进行主成分提取。
（2）找一个人脸数据（小点样本量），用你实现 PCA 方法对该数据降维，找出一些主成分，然后用这些主成分对每一副人脸图像进行重建，比较一些它们与原图像有多大差别（用信噪比衡量）。

2 数学原理

2.1 PCA 的数学原理

PCA 是一种降维手段，主要从 2 个方向取分析，但是得到的结果是相同的。
一个方向是最小重构性，选择降维的向量时要保证样本点离这个向量的距离最近。

另一个方向是最大可分性，选择降维的向量时要保证样本点在这个方向上的投影离的最远。要求投影之后的结果方差最大

从第二个方向入手分析：

设数据集为$\{\boldsymbol{x_1}, \boldsymbol{x_2}, \boldsymbol{x_3}, \dots, \boldsymbol{x_N}\}$,其中每一个数据均为$d$维向量，想要将其降至$d`$维。

$$\overline{\boldsymbol{x}} = \frac{1}{N}\sum_{n=1}^{N} x_n$$

$$\boldsymbol{S} = \frac{1}{N}\sum_{n=1}^{N} (\boldsymbol{x_n} - \overline{\boldsymbol{x}})(\boldsymbol{x_n} - \overline{\boldsymbol{x}})^T$$

其中$\overline{\boldsymbol{x}}$为数据集均值，$\boldsymbol{S}$为数据集协方差矩阵

如果$\boldsymbol{u}$为单位向量，则$\boldsymbol{x}$在$\boldsymbol{u}$上的投影就是$\boldsymbol{u^T x}$

投影后的协方差矩阵为：

$$\frac{1}{N}\sum_{n=1}^{N} (\boldsymbol{u^T x_n} - \boldsymbol{u^T}\overline{\boldsymbol{x}})(\boldsymbol{u^T x_n} - \boldsymbol{u^T}\overline{\boldsymbol{x}})^T$$

化简得到：

$$\boldsymbol{u^T S u}$$

因此就有

$$s.t\ \boldsymbol{u^T u} = \boldsymbol{1}$$
$$\boldsymbol{max(u)}\ \boldsymbol{u^T S u}$$

利用拉格朗日乘数法进行求解，得到：

$$\boldsymbol{Su} = \lambda \boldsymbol{u}$$

所以λ是矩阵$\boldsymbol{S}$的一个特征值，而$\boldsymbol{u}$是其对应的一个特征向量，这是第一主成分分析，上式左乘$\boldsymbol{u^T}$,可得目标函数就是要求λ最大。

因此要想获得最多的信息，就可以选取特征值最大的$d`$个特征向量，组成投影矩阵，将样本投影到低维空间中。

样本中心化的作用，使得样本均值为 0,这样对于协方差矩阵的运算更易求，可以直接表示成：

$$\frac{1}{N}\sum_{n=1}^{N} \boldsymbol{x_n x_n^T}$$

3 实验做法

3.1 生成数据

生成三维数据

```python
def generate_data(size, mu):
    # 协方差
    one = np.mat(np.array([100, 0, 0]))
    two = np.mat(np.array([0, 100, 0]))
    three = np.mat(np.array([0, 0, 1]))
    cov = np.vstack([one, two, three])
    one_kind = np.mat(np.random.multivariate_normal(mu, cov, size))
    return one_kind.T
```

PCA 主流程:

```python
def pca(self):
    c_data = self.__centralization()  # 中心化
    cov = c_data * c_data.T  # 数据的协方差矩阵(由于去中心化使得协方差不需要减去均值)
    eig_values, eig_vectors = np.linalg.eig(cov)  # 特征值分解
    eig_index = np.argsort(eig_values)  # 排序
    w_star = eig_vectors[:, eig_index[:-(self.small_d + 1):-1]]  # 寻找top-small_d大的维度，作为主成分
    self.__pca_data = w_star * w_star.T * c_data + self.mean  # 利用特征矩阵对数据降维
    return c_data, w_star  # 返回特征矩阵
```

中心化过程:

```python
def __centralization(self):  # 中心化
    mean = np.mean(self.data, 1)
    return self.data - mean
```

人脸数据读取(使用了 OpenCV 来处理图片):

```python
def read_face(file_path):
    file_list = os.listdir(file_path)  # 读取文件夹内文件
    data = []
    show_data = []
    size = (40, 40)  # 压缩原始图片大小以加快处理速度
    for file in file_list:
        path = os.path.join(file_path, file)
        img_gray = cv2.imread(path, cv2.IMREAD_GRAYSCALE)  # 读取灰度图
        img_gray = cv2.resize(img_gray, size)  # 图片压缩
        # 图片混合()
        if len(show_data) == 0:
            show_data = img_gray
        else:
            show_data = np.hstack([show_data, img_gray])
        h, w = img_gray.shape
        data_gray = img_gray.reshape(h * w)  # 图片拉伸
        data.append(data_gray)  # 加入处理数据集中
        a = np.array(show_data)
    cv2.imwrite('original pic.jpg', np.array(show_data))
    cv2.imshow('original picture', np.array(show_data))  # 显示图片
    cv2.waitKey(0)
    return np.mat(np.array(data)).T, size
```

对于人脸数据进行 PCA 处理

```
def pic_handle(img_data, sd, ori_size):
    """
    做PCA处理之后，在还原到原来的维度，然后显示，之后输出信噪比
    """
    Pca = PCA(sd, img_data)
    c_data, w_star = Pca.pca()  # 进行pca降维,获取投影矩阵
    w_star = np.real(w_star)
    print(w_star)
    new_data = w_star * w_star.T * c_data + Pca.mean  # 还原到原来的维度
    total_img = []
    # 图片混合
    for i in range(Pca.data_size):
        if len(total_img) == 0:
            total_img = new_data[:, i].T.reshape(ori_size)
        else:
            total_img = np.hstack([total_img, new_data[:, i].T.reshape(ori_size)])
    # 计算信噪比
    print('信噪比:')
    for i in range(Pca.data_size):
        a = psnr(np.array(data[:, i].T), np.array(new_data[:, i].T))
        print('图', i, '的信噪比为:', a, 'dB')
    # 处理图片
    total_img = np.array(total_img).astype(np.uint8)
    cv2.imwrite('pca image.jpg', total_img)  # 图片显示
    cv2.imshow('pca image', total_img)
    cv2.waitKey(0)
```

信噪比的计算公式从网上得知：

设原始图片为 A，为信息部分，带有噪声的图片为 B，那么噪声 C=B-A

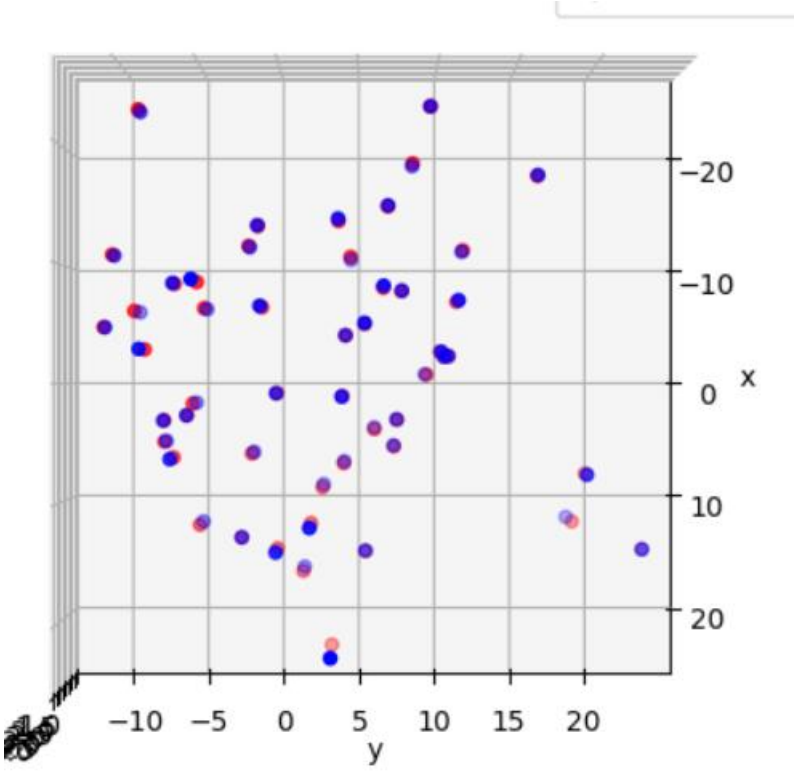
则 A/C 就是信噪比，然后再取$\log_{10}\frac{A}{C}$，以 dB 为单位

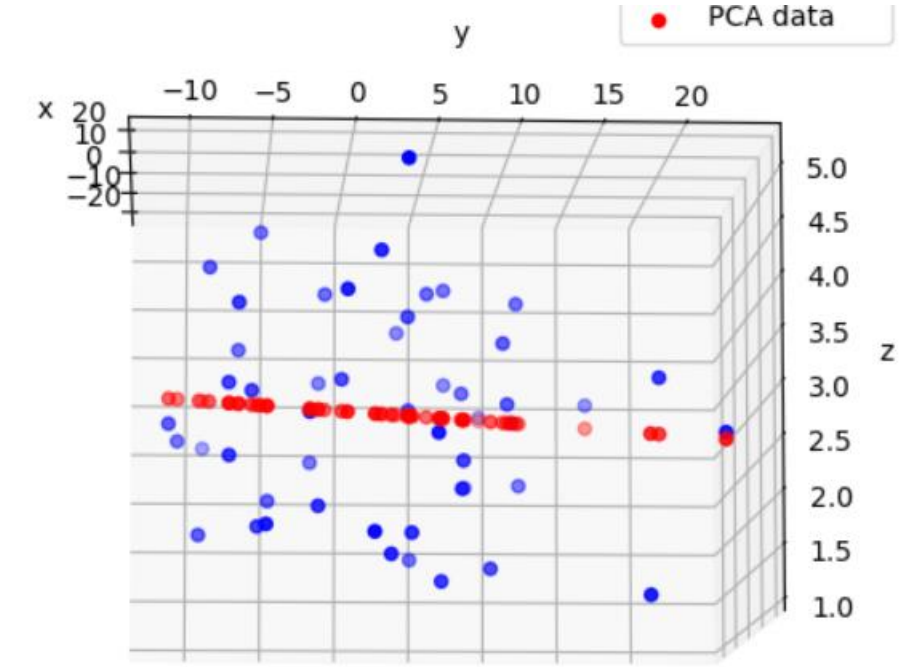
```
# 计算信噪比(越大越好)
def psnr(img1, img2):  # img1 原始图片 img2 处理图片(信息部分)
    img1 = np.real(img1)
    img2 = np.real(img2)
    noise = img1 - img2  # 噪声部分
    # 计算img2 和noise的方差
    var_info = np.mean((img2 - np.mean(img2)) ** 2)
    var_noise = np.mean((noise - np.mean(noise)) ** 2)
    snr = var_info / var_noise
    snr_dB = math.log10(snr)
    return snr_dB
```

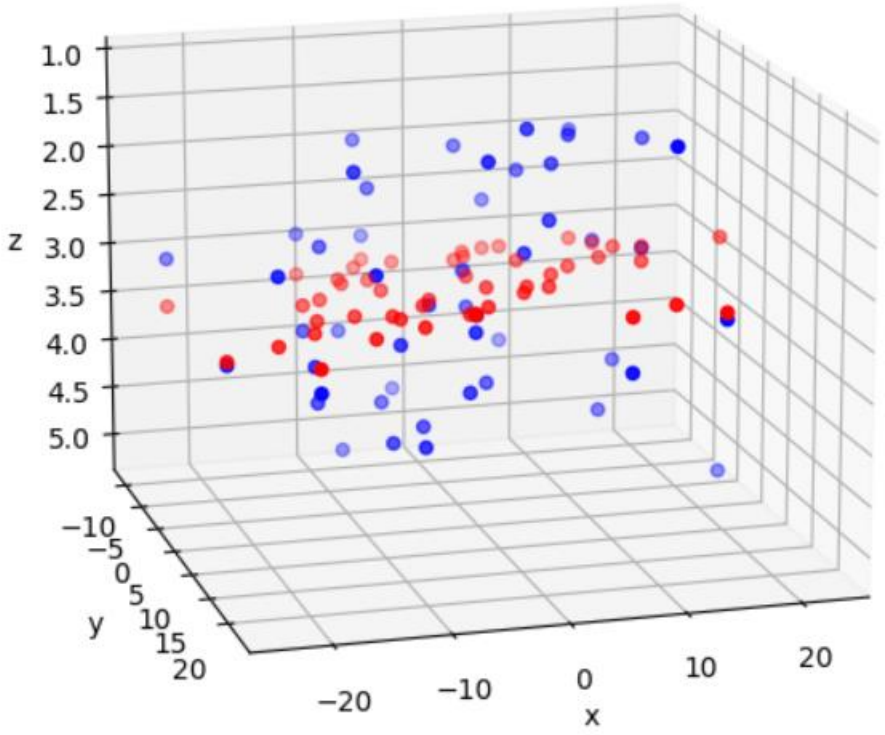
4 实验结果分析

4.1 人工数据结果

50 个点，将数据从 3 维降至 2 维
对降维数据做了处理($w\_star * w\_star.T * self.data$)







4.2 人脸处理结果

人脸处理结果：原图为 60*60 的图片
降至 20 维的处理效果：（上为原图，下为降维后图片）

信噪比为:

```
信噪比:
图 0 的信噪比为: 29.465050592737175 dB
图 1 的信噪比为: 29.557073728387817 dB
图 2 的信噪比为: 29.926581427710005 dB
图 3 的信噪比为: 29.5895736079409 dB
图 4 的信噪比为: 29.98879995545188 dB
```

降至 6 维的对比效果

信噪比为:

```
信噪比:
图 0 的信噪比为: 29.47570581449884 dB
图 1 的信噪比为: 29.55866999598459 dB
图 2 的信噪比为: 29.92494635224986 dB
图 3 的信噪比为: 29.599238379412586 dB
图 4 的信噪比为: 29.994727141453136 dB
```

降至 4 维的效果:

信噪比为:

```
信噪比:
图 0 的信噪比为: 29.47771220388963 dB
图 1 的信噪比为: 29.546119694028814 dB
图 2 的信噪比为: 29.927130422263282 dB
图 3 的信噪比为: 29.557525249403582 dB
图 4 的信噪比为: 29.987190536704126 dB
```

降至 3 维的对比效果:

信噪比为：

```
信噪比：
图 0 的信噪比为： 0.46933350378308064 dB
图 1 的信噪比为： 0.09463279505574224 dB
图 2 的信噪比为： 3.0222724321768974 dB
图 3 的信噪比为： 4.559905322376746 dB
图 4 的信噪比为： 1.4977153309298443 dB
```

降至 2 维的效果：

信噪比为：

```
信噪比：
图 0 的信噪比为： 0.21122263064771046 dB
图 1 的信噪比为： 0.09285105004319232 dB
图 2 的信噪比为： 1.3046375995149138 dB
图 3 的信噪比为： 2.2162780958630797 dB
图 4 的信噪比为： -0.2210330622652391 dB
```

从中可以看出 PCA 降维的效果随着维度的降低而变差，但是降值 20 维和降至 6 维的结果相差根本不大，降至 3 维和降至 6 维的结果出现了明显差异，并且有第一幅和第二幅中出现了相似特征，背景变黑又是由于第四幅图背景是黑色的缘故，主要特征已经混在一起无法明确的分辨图像了。而 2 维就导致均值的特征在各个图片中出现了。

信噪比在从 20 到 6 到 4 时，结果相差都不大，并且图片也很原图很像，降至 3 维和降至 2 维之后信噪比开始快速下降，图片也开始不清晰起来。

5 结论

(1) PCA 能根据主要的特征进行降维，在保留样本的多数特征的同时降低样本维度，简化计算。但是如果要降的维度过低，那么在怎么样也无法保留样本的主要特征了。

(2) PCA 能够进行图片压缩，在通过保留样本的均值，投影矩阵和降维后的中心化数据

就可以快速的近似还原图像，大大节省存储空间。但是当有一个新的样本加入时，需要重新计算均值，重新计算投影矩阵，重新计算降维后的数据仍旧不是很方便，所以这种存储方式适用于几乎没有新样本点加入的图片保存。