数据分析实验 3 —— PCA 图像压缩

16337053 信息安全 杜锦文

环境及工具

Windows10

python3.7.0

实验内容

输入一张灰度图片 Lena，使用 PCA 方法把原始图片分别按照 2:1、8:1、32:1 进行压缩，即压缩后的数据量为原始图片的 1/2、1/8、1/32。

分析压缩后的数据所含信息量大小，并比较压缩数据再经过重建后与原始图片的视觉差异。

数据处理思路

1、读入原始图像，将图像分割为 N 个 16*16 的块，作为 N 个 256 维度的样本；

2、使用 PCA 方法压缩到 128 维、32 维度、8 维，分别对应压缩比为 2：1，8：1，32：1；

3、计算压缩后数据的信息量大小；

4、对压缩的图像进行重建，还原为 N 个 256 维度样本；

5、将重建样本还原成 N 个 16*16 的块；

6、计算重建后的图像与原始图像的差值；

7、图像显示。

具体算法过程

1. **读取图像，将图像分割为 N 个 16*16 的块，作为 N 个 256 维度的样本**

使用 PIL 里的 Image 模块

```
from PIL import Image
# split img to N*256 blocks
# get Image object
# return ndarray object
```

```
def splitImage(im):
    im = im.convert('L')
    width,height = im.size
    im_list = []
    for i in range(height//16):
        for j in range(width//16):
            box = np.array(im.crop((j*16,i*16,j*16+16,i*16+16)))
            sample = box.reshape(1,256)
            im_list.extend((sample))
    return np.array(im_list)
```

2. **使用 PCA（主成分分析）将图像降到指定维度**

使用 sklearn 里面的 PCA 模块

```
from sklearn.decomposition import PCA
    # compression, to n dimensions
    pca = PCA(n_components=n, copy=True, whiten=False)
    compression = pca.fit_transform(img)
```

3. **计算压缩后数据的信息量大小**

利用 PCA 模块的 attribute 累计贡献率——explained_variance_ratio_，它是含有每个维度所含有图片信息量的一个 list，将其相加就可以得到降维后图片的信息量

```
 # info rate
    contri = 0
    for x in pca.explained_variance_ratio_:
        contri += x
```

4. **对压缩的图像进行重建，还原为 N 个 256 维度样本**

利用 PCA 模块自带的 inverse 逆向重建，还原图像到原来的维度，如果降维时进行了 whiten 操作，此处也会进行逆向。

```
# reconstruction
    recon_arr = pca.inverse_transform(compression)
    recon = combineImage(recon_arr)
    return compression,contri,recon
```

5. **将重建样本还原成 N 个 16*16 的块；**

```
# recover img to N*256 blocks
# get ndarray object
# return Image object
```

```
def combineImage(img):
    x = 0
    width = 512
    height = 512
    img_new = Image.new('L',(width,height))
    for i in range(width//16):
        for j in range(height//16):
            block =  img[x].reshape(16,16)
            sample = Image.fromarray(block).convert('L')
            img_new.paste(sample,(j*16,i*16,j*16+16,i*16+16))
            x += 1
    return img_new
```

6. **计算重建后的图像与原始图像的差值**

利用 PIL 里面的 ImageChops 模块，注意此处需要把图片转换为统一的格式，否则会报错，这里我将图片统一转化为二值图。

```
from PIL import ImageChops
# calculate different
def dif(im,recon):
    diff = ImageChops.difference(im,recon)
    return diff
```

7. **图像显示**

利用 matplotlib 的 pyplot，将上述的图像作为子图，综合画在一张图上

```
from matplotlib import pyplot as plt
def showim(im,cmpr,recon,diff,contri,n):
    plt.figure()
    ax = plt.subplot(232)
    plt.imshow(im)
    ax.set_title('origin')

    ax = plt.subplot(223)
    plt.imshow(recon)
    ax.set_title('cmpr:{}, info:{:.5f}'.format(256//n,contri))

    ax = plt.subplot(224)
    plt.imshow(diff)
    ax.set_title('diff')

    plt.show()
```

8. **图像处理函数**

```
# process of a image
def process(im,n):
    img = splitImage(im)
    cmpr,contri,recon = pca(img,n)
    diff = dif(im,recon)
    showim(im,cmpr,recon,diff,contri,n)
```

9. **主函数**

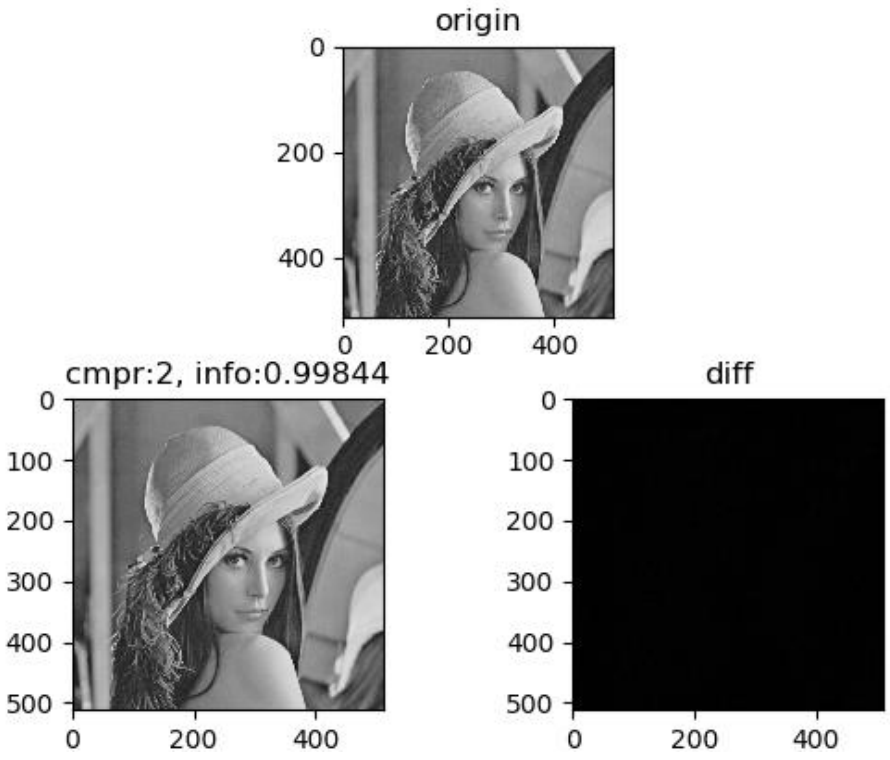
要得到压缩比为压缩比为 2：1，8：1，32：1 的图像，应该分别降维到 128、32、8 维。

```
def main():
    im = Image.open('img/origin.bmp')
    process(im,128)
    process(im,32)
    process(im,8)
```

数据处理结果

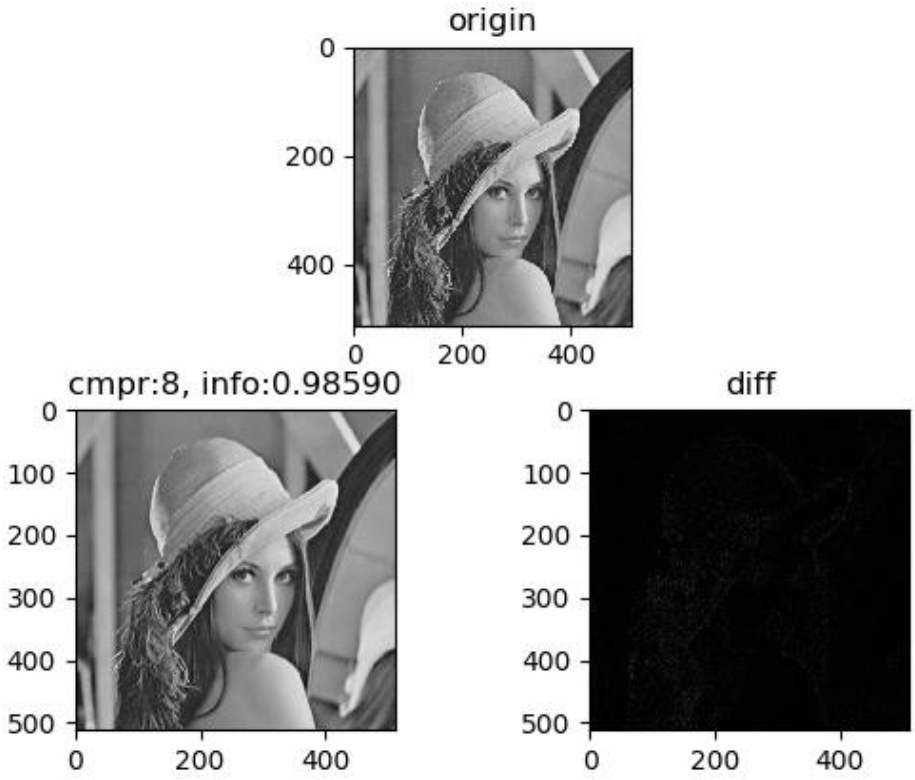
1. 2：1

基本上无差异，信息量接近 1



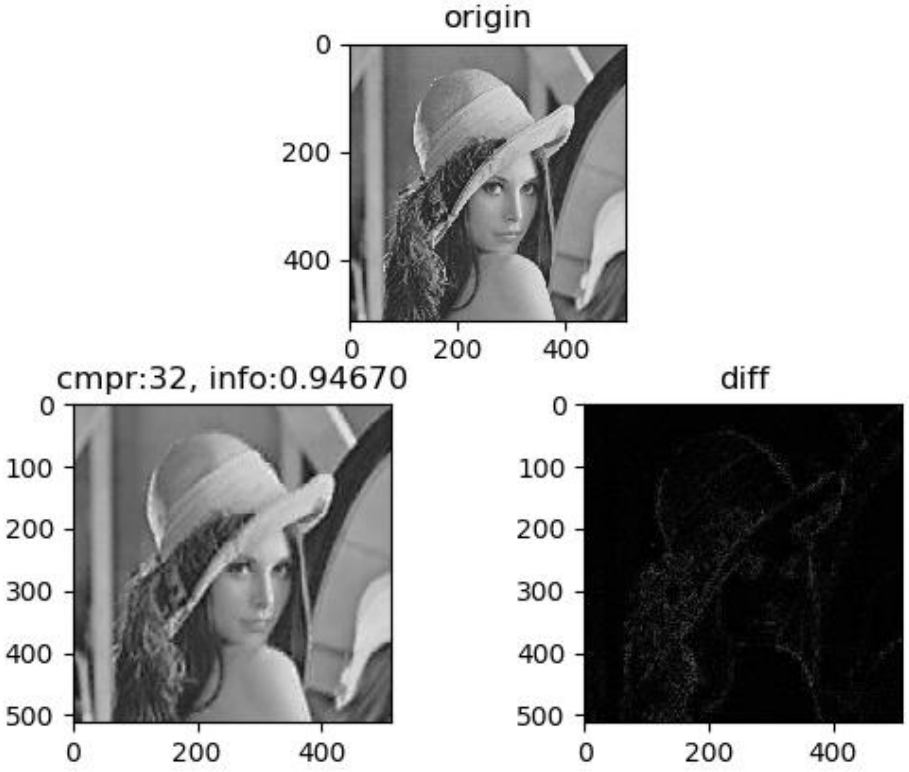
2. 8：1

视觉可见少许差异，图像信息量减小



3. 32：1

视觉可见差异，图像变模糊，信息量减小



【附】完整代码

File: pca.py

```
import numpy as np
from PIL import Image
from PIL import ImageChops
from sklearn.decomposition import PCA
from matplotlib import pyplot as plt

# split img to N*256 blocks
def splitImage(im):
    im = im.convert('L')
    width,height = im.size
    im_list = []
    for i in range(height//16):
        for j in range(width//16):
            box = np.array(im.crop((j*16,i*16,j*16+16,i*16+16)))
            sample = box.reshape(1,256)
            im_list.extend((sample))
    return np.array(im_list)

# recover img to N*256 blocks
def combineImage(img):
    x = 0
    width = 512
    height = 512
    img_new = Image.new('L',(width,height))
    for i in range(width//16):
        for j in range(height//16):
            block =  img[x].reshape(16,16)
            sample = Image.fromarray(block).convert('L')
            img_new.paste(sample,(j*16,i*16,j*16+16,i*16+16))
            x += 1
    return img_new

# PCA operations
def pca(img,n):
    # compression, to n dimensions
    pca = PCA(n_components=n, copy=True, whiten=False)
    compression = pca.fit_transform(img)
```

```
    # info rate
    contri = 0
    for x in pca.explained_variance_ratio_:
        contri += x
    # reconstruction
    recon_arr = pca.inverse_transform(compression)
    recon = combineImage(recon_arr)
    return compression,contri,recon

# calculate different
def dif(im,recon):
    diff = ImageChops.difference(im,recon)
    return diff

def showim(im,cmpr,recon,diff,contri,n):
    plt.figure()
    ax = plt.subplot(232)
    plt.imshow(im)
    ax.set_title('origin')
    ax = plt.subplot(223)
    plt.imshow(recon)
    ax.set_title('cmpr:{}, info:{:.5f}'.format(256//n,contri))
    ax = plt.subplot(224)
    plt.imshow(diff)
    ax.set_title('diff')
    plt.show()

def process(im,n):
    img = splitImage(im)
    cmpr,contri,recon = pca(img,n)
    diff = dif(im,recon)
    showim(im,cmpr,recon,diff,contri,n)

def main():
    im = Image.open('img/origin.bmp')
    process(im,128)
    process(im,32)
    process(im,8)

if __name__ == '__main__':
    main()
```