2.7 奇异值分解

- 奇异值分解的定义

  假设 A 是一个 m * n 的矩阵，则存在一个分解使得

  $$A_{m\times n} = U_{m\times m}\Lambda_{m\times n}V^T_{n\times n}$$

  其中 U 为左奇异值矩阵，Λ 为矩阵 A 奇异值，除了主对角线上的元素以外全为0，V 为右奇异值矩阵

- 奇异值分解的数学计算

  如果我们将A的转置和A做矩阵乘法，那么会得到n×n的一个方阵 A^TA 。既然 A^TA 是方阵，那么我们就可以进行特征分解，得到的特征值和特征向量满足下式：

  $$(A^TA)v_i = \lambda_i v_i$$

  这样我们就可以得到矩阵 A^TA 的n个特征值和对应的n个特征向量v了。将 A^TA 的所有特征向量张成一个n×n的矩阵V，就是我们SVD公式里面的V矩阵了。一般我们将V中的每个特征向量叫做A的右奇异向量。

  如果我们将A和A的转置做矩阵乘法，那么会得到m×m的一个方阵 AA^T 。既然 AA^T 是方阵，那么我们就可以进行特征分解，得到的特征值和特征向量满足下式：

  $$(AA^T)u_i = \lambda_i u_i$$

  这样我们就可以得到矩阵 AA^T 的m个特征值和对应的m个特征向量u了。将 AA^T 的所有特征向量张成一个m×m的矩阵U，就是我们SVD公式里面的U矩阵了。一般我们将U中的每个特征向量叫做A的左奇异向量。

  U和V我们都求出来了，现在就剩下奇异值矩阵Σ没有求出了.

  由于Σ除了对角线上是奇异值其他位置都是0，那我们只需要求出每个奇异值σ就可以了。

  我们注意到:

  $$A = U\Sigma V^T \Rightarrow AV = U\Sigma V^TV \Rightarrow AV = U\Sigma \Rightarrow Av_i = \sigma_i u_i \Rightarrow \sigma_i = Av_i/u_i$$

  这样我们可以求出我们的每个奇异值，进而求出奇异值矩阵Σ。

- **SVD的python实例**

  在 Python 中进行 SVD 分解非常简单，利用 Numpy 模块中的 np.linalg.svd() 函数，比如 u,sigma,v = np.linalg.svd(A)，其中 u，v 分别返回矩阵 A 的左右奇异向量，而 sigma 返回的是按从大到小的顺序排列的奇异值，利用 Python 进行 SVD 分解对图形进行压缩，也就是读取图片的像素矩阵，然后对矩阵进行 SVD 分解得到对应的奇异值和奇异向量，然后对奇异值和奇异向量进行筛选例如取前10%的数据，实现对图像的压缩

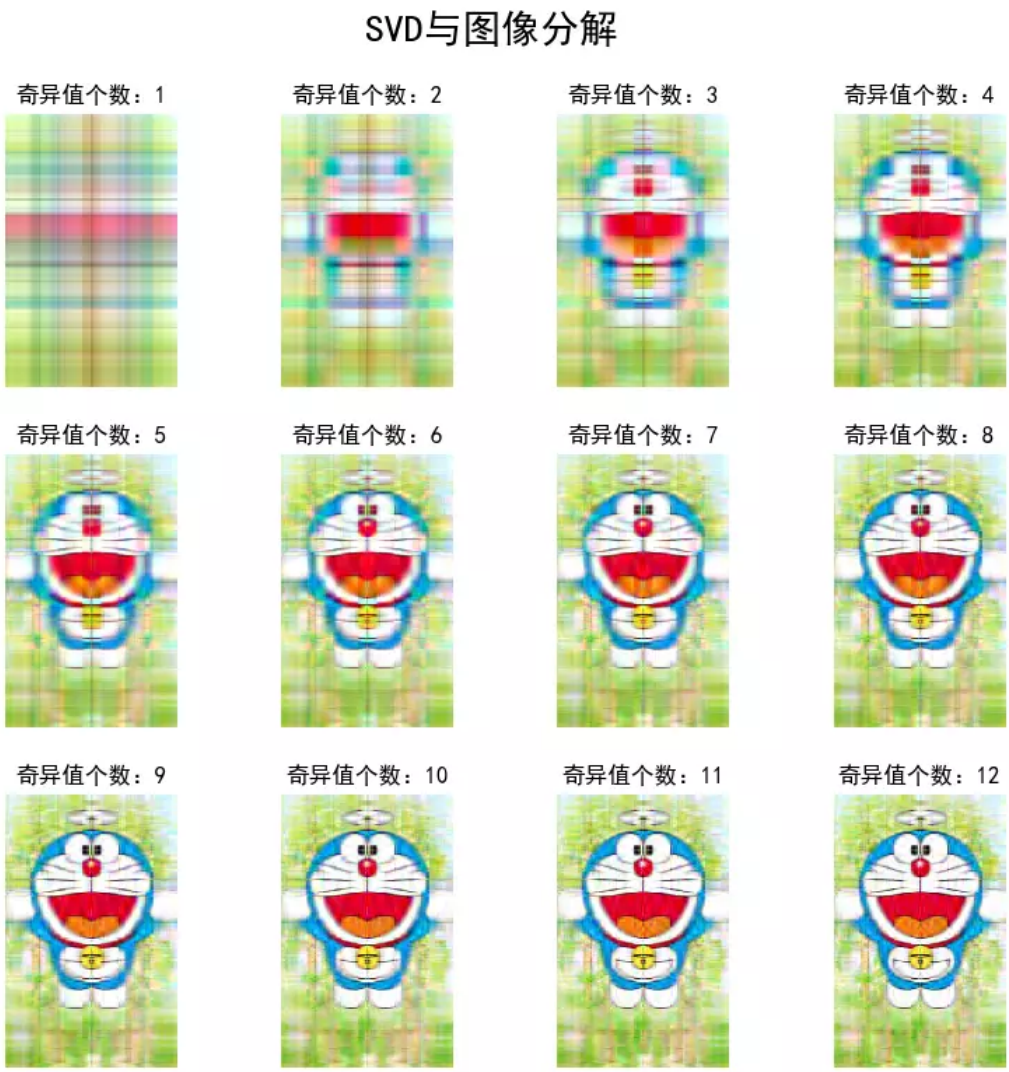
  原图：

- 进行 SVD 分解，选择前 50 个奇异值

```
import numpy as np
import os
from PIL import Image

def restore(sigma, u, v, K):  # 奇异值、左特征向量、右特征向量
    m = len(u)
    n = len(v[0])
    a = np.zeros((m, n))
    for k in range(K):
        uk = u[:, k].reshape(m, 1)
        vk = v[k].reshape(1, n)
        a += sigma[k] * np.dot(uk, vk)   # 前 k 个奇异值的加和
    a = a.clip(0, 255)
    return np.rint(a).astype('uint8')

if __name__ == "__main__":
    A = Image.open(".\\Pokonyan.jpg", 'r')
    output_path = r'.\SVD_Out'
    a = np.array(A)
    print('type(a) = ', type(a))
    print('原始图片大小：', a.shape)

   # 图片有RGB三原色组成，所以有三个矩阵
    u_r, sigma_r, v_r = np.linalg.svd(a[:, :, 0])    # 奇异值分解
    u_g, sigma_g, v_g = np.linalg.svd(a[:, :, 1])
```

```
    u_b, sigma_b, v_b = np.linalg.svd(a[:, :, 2])

    # 仅使用前1个，2个，...，50个奇异值的结果
    K = 50
    for k in range(1, K+1):
        R = restore(sigma_r, u_r, v_r, k)
        G = restore(sigma_g, u_g, v_g, k)
        B = restore(sigma_b, u_b, v_b, k)
        I = np.stack((R, G, B), axis=2)    # 将矩阵叠合在一起，生成图像
        Image.fromarray(I).save('%s\\svd_%d.jpg' % (output_path, k))
```

- 将图像进行奇异值分解后的图像



- 使用前50个奇异值的图像

可以看到，使用前50个奇异值就能大致还原原图像，也就是可以通过仅仅使用奇异值矩阵中前面一部分的值表示整体的情况，从而实现了特征的降维，这是因为在奇异值矩阵中奇异值减少的特别快，可以用最大的k个的奇异值和对应的左右奇异向量来近似描述矩阵