**SVD图像压缩实验**

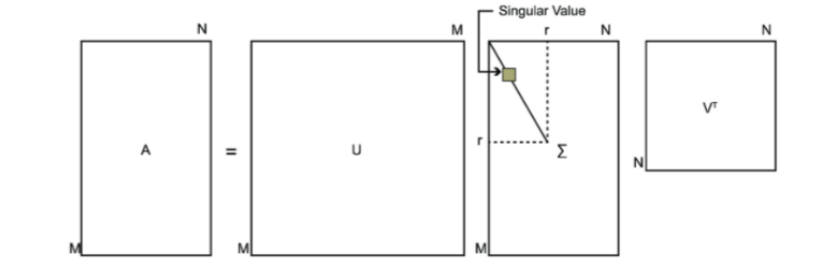
曹可想 2016100104008

1. **SVD原理**

SVD是对矩阵进行分解,但不同于特征值分解，SVD并不要求要分解矩阵为方阵。假设要分解的矩阵A是一个的举阵，那么定义矩阵A的SVD分解为：

其中U是的正交矩阵，是一个的矩阵，除主对角线上元素外全为0，主对角线上的元素为奇异值，V是一个的正交矩阵。

通过下图可以很直观的理解SVD的定义：



下面介绍U、、V的计算方法。

回忆方阵的相似对角化,如果B为对称矩阵，且将其特征向量标准化，那么由B特征向量构成的矩阵P即为正交矩阵，则相似对角化可以记为。

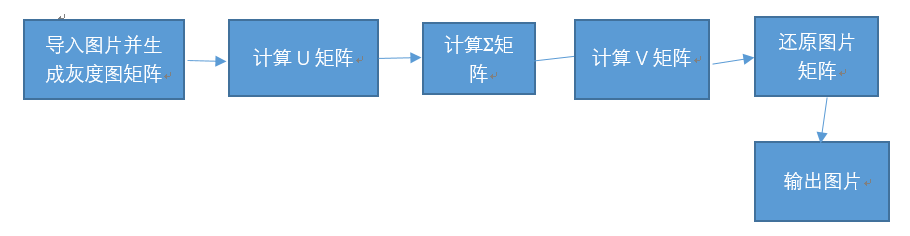
在中，刚好是一个对角矩阵，这样我们就可以得出结论矩阵U是矩阵的特征向量标准化之后构成的。同时可以得出矩阵主对线上的元素即为矩阵的特征值开根号后的结果。

同理可得：矩阵V是矩阵特征向量标准化之后构成的。

仔细观察和我们会发现，矩阵和矩阵的前个特征值相同，为奇异值的平方，其余特征值全为0.

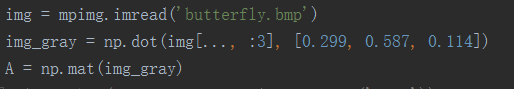
下面介绍奇异值的另一种求法：

1. **SVD图像压缩**
2. 编程思路



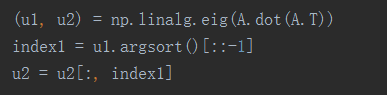
1. 代码
2. 读取图片生成灰度图矩阵

利用matplotlib.image的imread()函数读取图片为RGB形式存储的三维数组，然后利用公式转化为灰度图。



1. 计算U矩阵

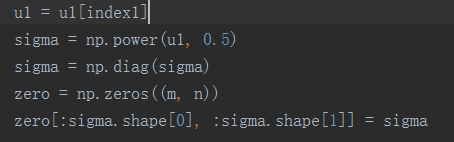
利用numpy.linalg.eig()函数计算的特征值、特征向量，特征向量会被自动标准化，然后把特征向量按特征值由大到小的顺序排列构成U矩阵。



1. 计算矩阵

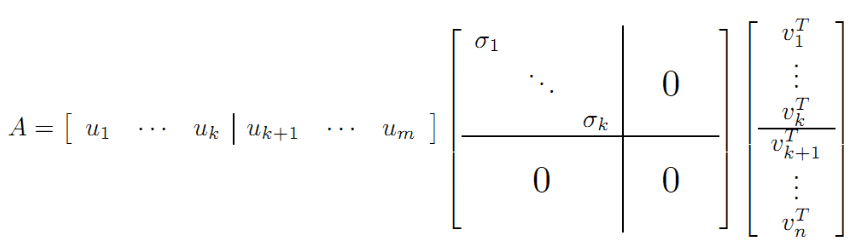
首先把的特征值按从大到小的顺序排列并开方，然后以其为对角元构建

矩阵。



1. 计算V矩阵

开始打算用跟求U矩阵相同的方法计算V矩阵，但是因为有0特征值，python计算的结果0特征值不会为0，而是一个非常小的数，所以最终计算结果并不准确，导致还原的图片无法识别。查找资料后发现如下规律：



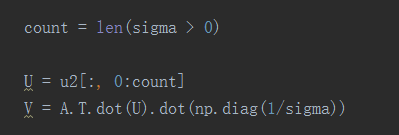
写成分块矩阵形式：

(k是奇异值非零的个数)

说明矩阵A的SVD分解可以将进一步化简。

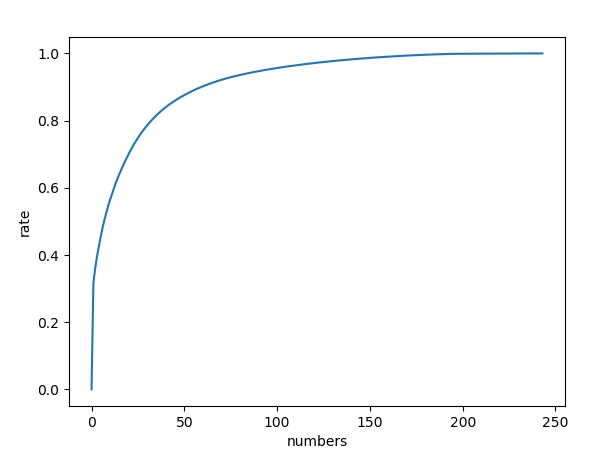
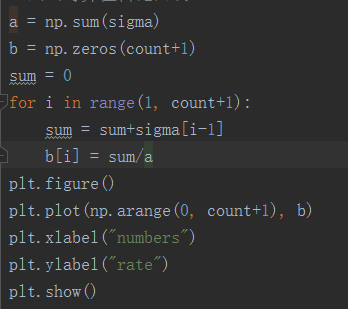
根据表达式，可以得到.

编程代码如下：



1. 选择奇异值个数

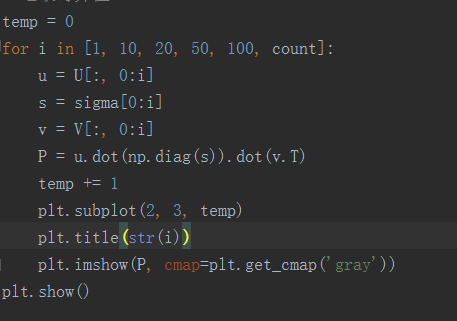
首先画出随奇异值个数增加，奇异值之和占总体奇异值的比重。

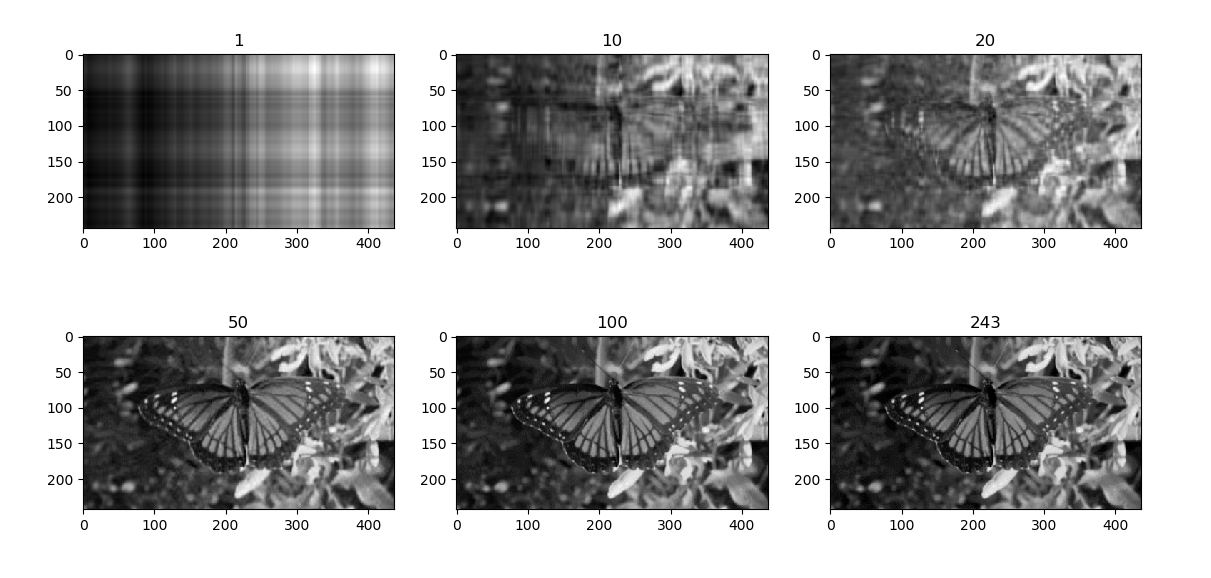


可以看出当奇异值为前100个时，比重已经达到90%。

在选择奇异值时，依此选择1，10，20，50，100，243个奇异值来观察图片变化。

1. 显示图片





1. **完整代码**

import numpy as np  
from SVD import \*  
import matplotlib.pyplot as plt  
import matplotlib.image as mpimg  
# 读取图片为灰度图  
img = mpimg.imread('butterfly.bmp')  
img\_gray = np.dot(img[..., :3], [0.299, 0.587, 0.114])  
# 对灰度图进行svd分解  
A = np.mat(img\_gray)  
U, sigma, V, count = get\_svd(A)  
# 画出奇异值占比曲线  
a = np.sum(sigma)  
b = np.zeros(count+1)  
sum = 0  
for i in range(1, count+1):  
 sum = sum+sigma[i-1]  
 b[i] = sum/a  
plt.figure()  
plt.plot(np.arange(0, count+1), b)  
plt.xlabel("numbers")  
plt.ylabel("rate")  
plt.show()  
# 选取奇异值  
temp = 0  
for i in [1, 10, 20, 50, 100, count]:  
 u = U[:, 0:i]  
 s = sigma[0:i]  
 v = V[:, 0:i]  
 P = u.dot(np.diag(s)).dot(v.T)  
 temp += 1  
 plt.subplot(2, 3, temp)  
 plt.title(str(i))  
 plt.imshow(P, cmap=plt.get\_cmap('gray'))  
plt.show()

import numpy as np  
# SVD函数  
def get\_svd(A):  
 (u1, u2) = np.linalg.eig(A.dot(A.T))  
 index1 = u1.argsort()[::-1]  
 u1 = u1[index1]  
 u2 = u2[:, index1]  
 sigma = np.power(u1, 0.5)  
 count = len(sigma > 0)  
 U = u2[:, 0:count]  
 V = A.T.dot(U).dot(np.diag(1/sigma))  
 return U, sigma, V, count