Singular Value Decomposition of an Image

Frank Cleary | www.frankcleary.com (http://www.frankcleary.com) | See also: Singular Value Decomposition Introduction (http://www.frankcleary.com/svd) | Notebook Gist (https://gist.github.com/frankcleary/4d2bd178708503b556b0)

Introduction

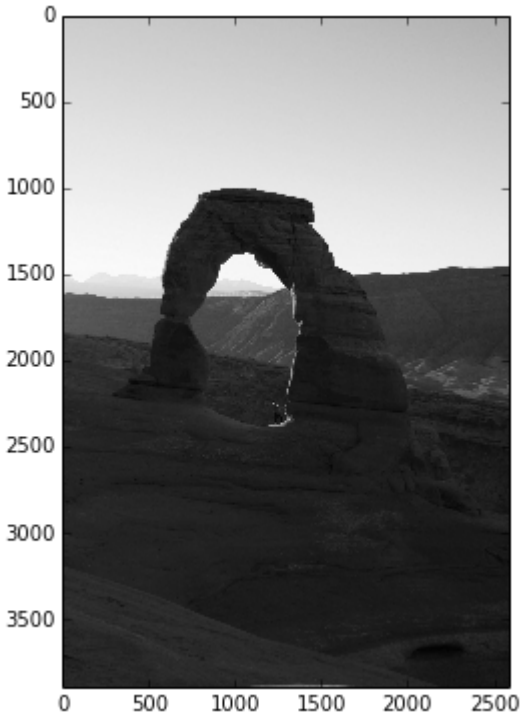
In my introduction to singular value decomposition (link to notebook (http://www.frankcleary.com/svd)), I mentioned that singular value decomposition has applications in image compression. Here I'll give a bit more explanation of how that works, and showcase some of the tools for manipulating images in python. The key here is that a black and white image is just a matrix where the numbers represent the intensity of a given pixel, which can be decomposed just like any other.

]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
import time

from PIL import Image
```

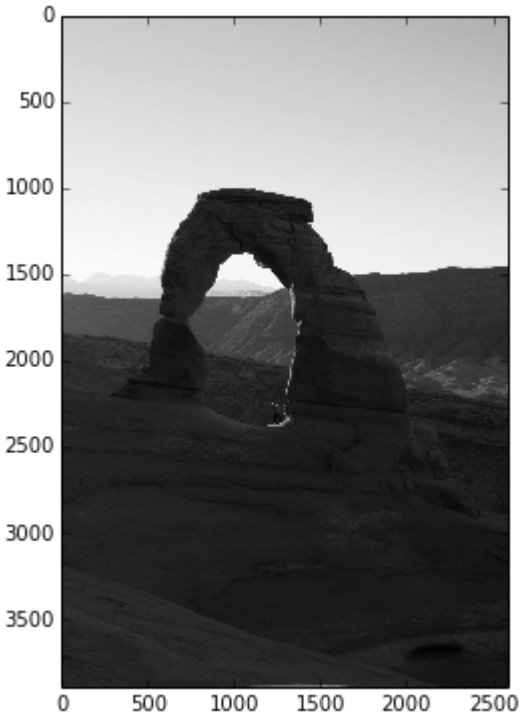
Here I'll load an image I took of Delicate Arch in Arches National Park, and convert it to black and white.

]:
```
img = Image.open('input/img_6847.jpg')
imggray = img.convert('LA')
plt.figure(figsize=(9, 6))
plt.imshow(imggray);
```



Now I'll convert the image data into a numpy matrix, plotting the result to show the data is unchanged.

]:
```
imgmat = np.array(list(imggray.getdata(band=0)), float)
imgmat.shape = (imggray.size[1], imggray.size[0])
imgmat = np.matrix(imgmat)
plt.figure(figsize=(9,6))
plt.imshow(imgmat, cmap='gray');
```

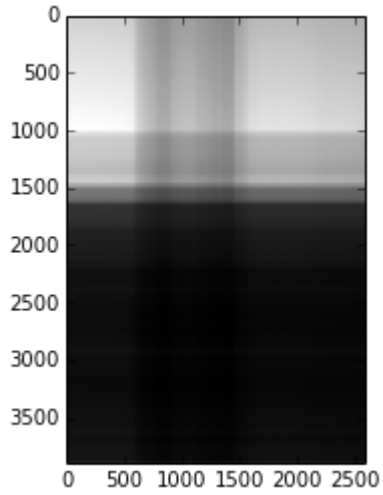


Now to compute the singular value decomposition:

]:
```
U, sigma, V = np.linalg.svd(imgmat)
```

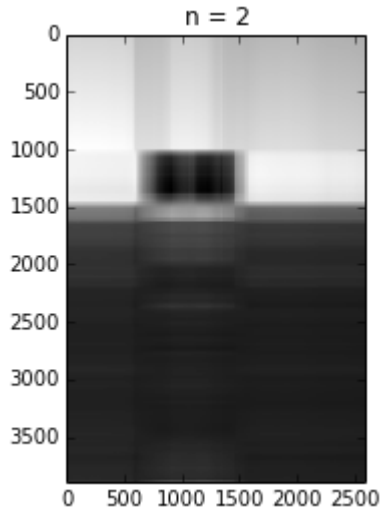
Computing an approximation of the image using the first column of U and first row of V reproduces the most prominent feature of the image, the light area on top and the dark area on the bottom. The darkness of the arch causes the extra darkness in the middle of the reconstruction. Each column of pixels in this image is a different weighting of the same values, $\vec{u}_1$:

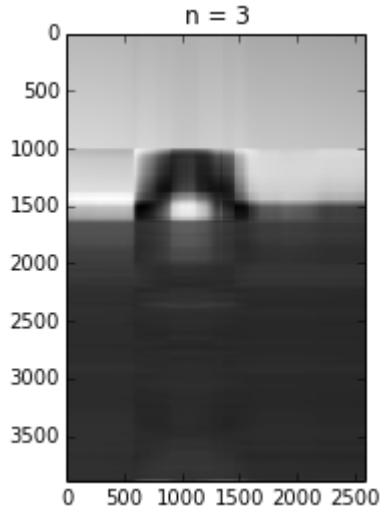
]:
```
reconstimg = np.matrix(U[:, :1]) * np.diag(sigma[:1]) * np.matrix(V[:1, :])
plt.imshow(reconstimg, cmap='gray');
```



> Even with just the second and third vectors, the shape of the arch begins to appear.

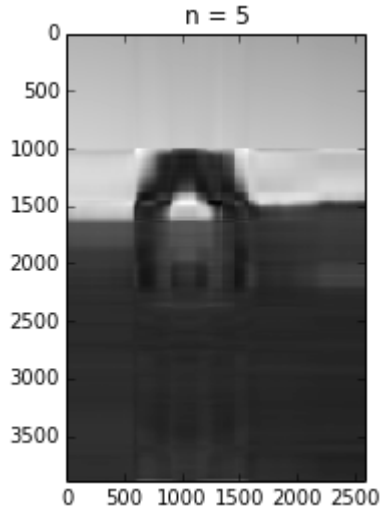
]:
```
for i in xrange(2, 4):
    reconstimg = np.matrix(U[:, :i]) * np.diag(sigma[:i]) * np.matrix(V[:i, :])
    plt.imshow(reconstimg, cmap='gray')
    title = "n = %s" % i
    plt.title(title)
    plt.show()
```

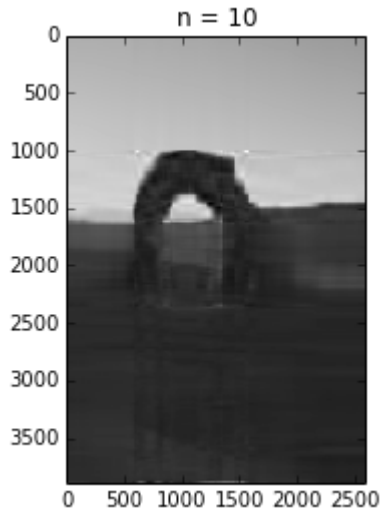


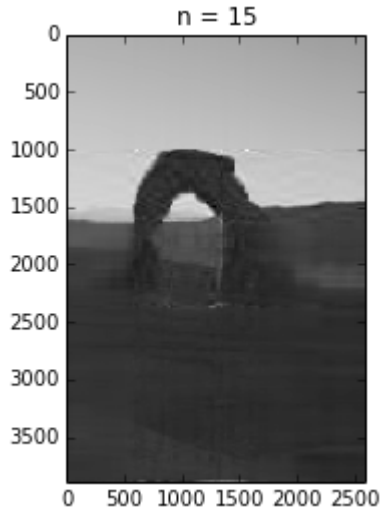


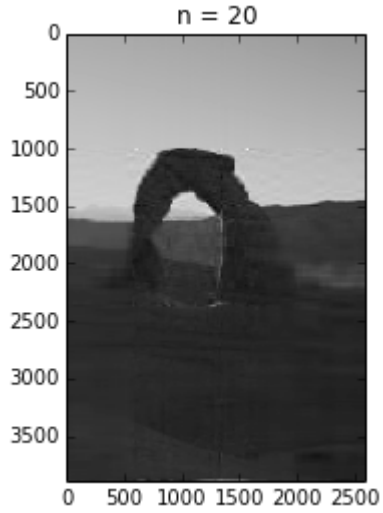
The loop below shows the reconstructed image using the first n vectors of the singular value decomposition (n is shown in the title of the plot). The first 50 vectors produce an image very close the original image, while taking up only $\frac{50*3900+50+50*2600}{3900*2600} \approx 3.2\%$ as much space as the original data.

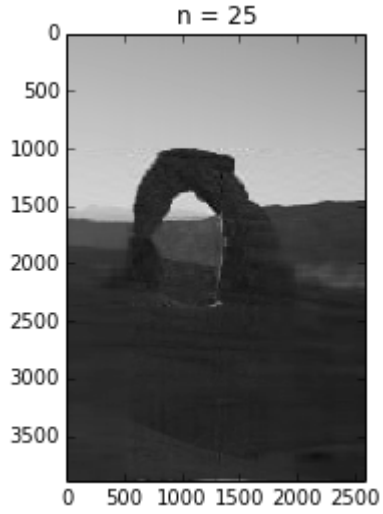
]:
```
for i in xrange(5, 51, 5):
    reconstimg = np.matrix(U[:, :i]) * np.diag(sigma[:i]) * np.matrix(V[:i, :])
    plt.imshow(reconstimg, cmap='gray')
    title = "n = %s" % i
    plt.title(title)
    plt.show()
```

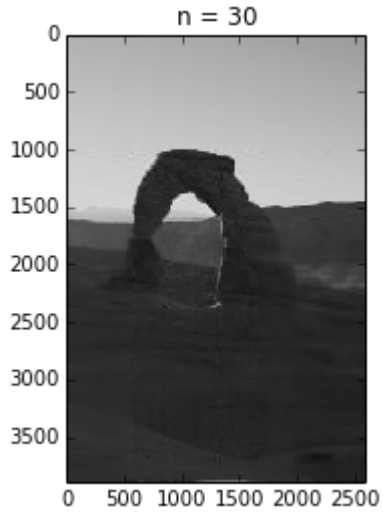

n = 5
0
500
1000
1500
2000
2500
3000
3500
0 500 1000 1500 2000 2500

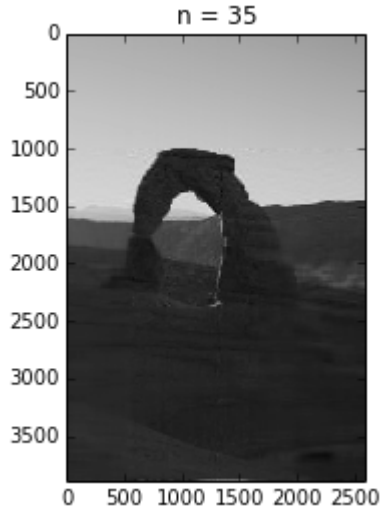

n = 10
0
500
1000
1500
2000
2500
3000
3500
0 500 1000 1500 2000 2500

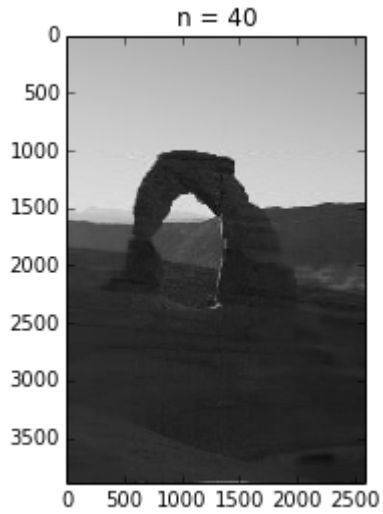

n = 15
0
500
1000
1500
2000
2500
3000
3500
0 500 1000 1500 2000 2500

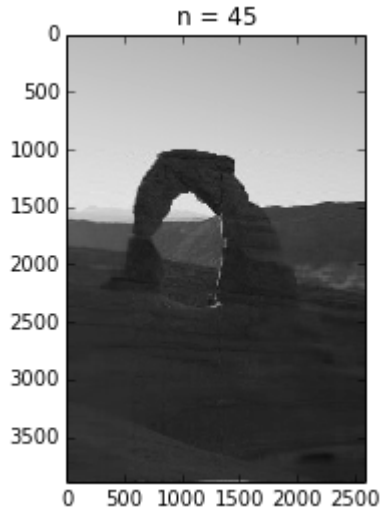

n = 20
0
500
1000
1500
2000
2500
3000
3500
0
500
1000
1500
2000
2500

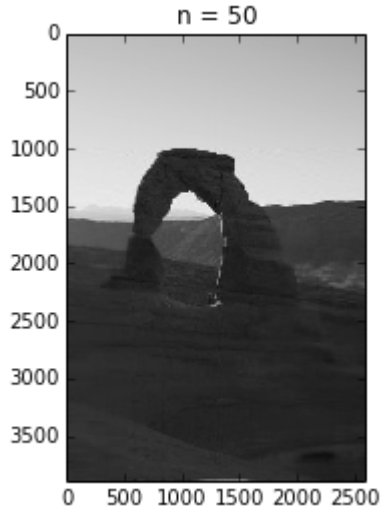

n = 25
0
500
1000
1500
2000
2500
3000
3500
0
500
1000
1500
2000
2500


n = 30
0
500
1000
1500
2000
2500
3000
3500
0
500
1000
1500
2000
2500


n = 35
0
500
1000
1500
2000
2500
3000
3500
0
500
1000
1500
2000
2500


n = 40
0
500
1000
1500
2000
2500
3000
3500
0
500
1000
1500
2000
2500


n = 45
0
500
1000
1500
2000
2500
3000
3500
0
500
1000
1500
2000
2500



See also: Singular Value Decomposition Introduction (http://www.frankcleary.com/svd)

Back to main site: www.frankcleary.com (http://www.frankcleary.com)