MAT4110/MAT3110 Compulsory assignment 2

Soran Hussein Mohmmed / soranhm

October 2018

Compress images

Representing an n x m matrix of a black and white image, consisting of normalized integer values between 0 and 1 with the function im2double() in matlab. Using SVD function in matlab to divide the matrix into u,σ and v. were u and v are the left and right matrices and σ is the matrix with singular values on the diagonal. We can get back back to matrix A we used SVD on be computing A = uσv'. Compressing each image with respect of a acceptable r. Compressed the σ matrix with singular values on the diagonal on a rxr marix and the other values equal to 0 (from r to n and r to m equal to 0). In the first image we have a Chessboard with 2 colors, black and white, a good acceptable choice of r is 2 then.

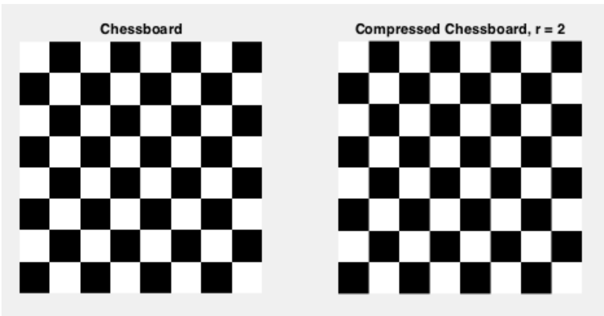

Figure 1: compressed and uncompressed Cheesboard, with r = 2

While for the Jellyfish picture we have more details and after picking some r values and zooming in on some point i get that picking r = 100 is a good approximation.

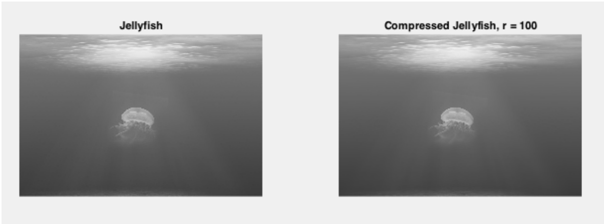

Figure 2: compressed and uncompressed Jellyfish, with r = 100

When it comes to the New York picture, it becomes a bit harder to find a r that is low and gives a acceptable view of the picture, but after trying out som r values and zooming in on the picture i found out that r = 600 is acceptable.

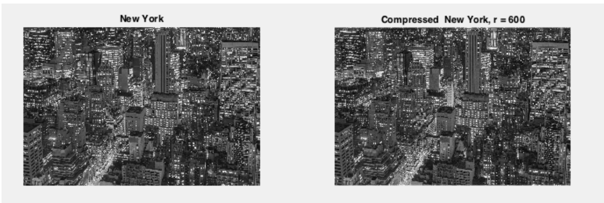

Figure 3: compressed and uncompressed New York, with r = 600

To get a better view of the singular values in each picture i plot the diagonal values of σ (singular values) with help of the semilogy() function in matlab, who gives a good view.

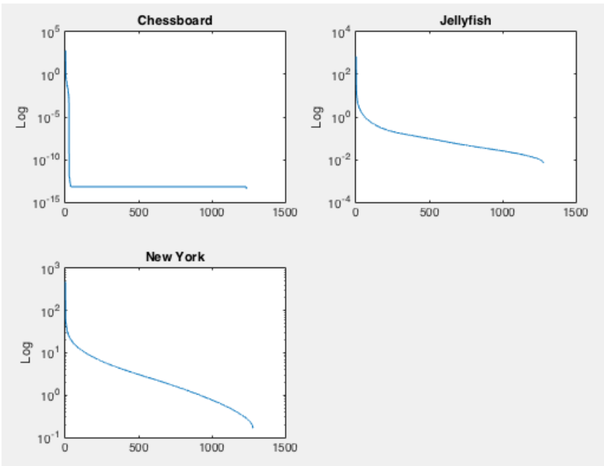

Figure 4: Log plot of the singular values

At last we have the compression ratio in each picture, who is described as:

$$compression\ ratio = \frac{uncomresssed\ size}{copmressed\ size}$$

For the uncompressed picture we have n x m , while for compressed picture with SVD we have: r(n + m + 1), this gives:

$$compression\ ratio = \frac{n * m}{r(n + m + 1)}$$

I caluclate those in each case with respect of the r values that gave best approximation:

```
Compression_ratio1 =

  314.2789


Compression_ratio2 =

    7.6752


Compression_ratio3 =

    1.2788
```

Figure 5: Compression rate in each case

```
% Convert the images to grey scale
% https://pixabay.com/en/board-chess-chessboard-black-white-157165/
im1 = rgb2gray(imread('chessboard.png'));
% https://pixabay.com/en/jellyfish-under-water-sea-ocean-698521/
im2 = rgb2gray(imread('jellyfish.jpg'));
% https://pixabay.com/en/new-york-city-skyline-nyc-690868/
im3 = rgb2gray(imread('new_york.jpg'));
% Convert to double between 0 and 1.
im1 = im2double(im1);
im2 = im2double(im2);
im3 = im2double(im3);

[u1,s_1,v1] = svd(im1);
[u2,s_2,v2] = svd(im2);
[u3,s_3,v3] = svd(im3);


%Plot of the log of the singular values
figure(4)
subplot(221)
semilogy(diag(s_1))
title('Chessboard')
ylabel('Log')
subplot(222)
semilogy(diag(s_2))
title('Jellyfish')
ylabel('Log')
subplot(223)
semilogy(diag(s_3))
title('New York')
ylabel('Log')


%Compressing
r1 = 2   % gives good approx for cheesboard
r2 = 100 % gives good approx for jellyfish
r3 = 600 % gives good approx for new york

s1 = s_1; s2 = s_2; s3 = s_3;
s1(r1+1:end, :) = 0; s1(:,r1+1:end) = 0; % make the rest zero
s2(r2+1:end, :) = 0; s2(:,r2+1:end) = 0; % make the rest zero
s3(r3+1:end, :) = 0; s3(:,r3+1:end) = 0; % make the rest zero

D1 = u1*s1*v1';
D2 = u2*s2*v2';
D3 = u3*s3*v3';

imwrite(D1,'compressed_chessboard.png')
imwrite(D2,'compressed_jellyfish.jpg')
imwrite(D3,'compressed_new_york.jpg')


% Plot the images
figure(1)
```

```
subplot(121)
imshow(im1,'InitialMagnification',50)
title('Chessboard')
subplot(122)
imshow(D1,'InitialMagnification',50)
title('Compressed Chessboard, r = 2')

figure(2)
subplot(121)
imshow(im2,'InitialMagnification',50)
title('Jellyfish')
subplot(122)
imshow(D2,'InitialMagnification',50)
title('Compressed Jellyfish, r = 100')

figure(3)
subplot(121)
imshow(im3,'InitialMagnification',50)
title('New York')
subplot(122)
imshow(D3,'InitialMagnification',50)
title('Compressed  New York, r = 600')


% Compression ratio
[cn1,cm1] = size(D1); [ucn1,ucm1] = size(im1);
[cn2,cm2] = size(D2); [ucn2,ucm2] = size(im2);
[cn3,cm3] = size(D3); [ucn3,ucm3] = size(im3);
Compression_ratio1 = (ucn1 * ucm1)/(r1 * (cn1 + cm1 + 1))
Compression_ratio2 = (ucn2 * ucm2)/(r2 * (cn2 + cm2 + 2))
Compression_ratio3 = (ucn3 * ucm3)/(r3 * (cn3 + cm3 + 3))
```