Aula prática 6
Imagens e SVD

Will Sena*

Contents

*wllsena@protonmail.com

1 Visualização

Faça a leitura da imagem usando o comando Scilab A = imread('imagem.png') (a imagem também pode estar no formato .jpg). A matriz A será então uma matriz m x n com elementos inteiros de 0 a 255. Escreva uma função Scilab chamada visualization que recebe uma matriz A representativa de uma imagem em tons de cinza e mostra essa imagem na tela. Use os comandos subplot e imshow. (Veja os "helps" desses comandos no Scilab).

```
function visualization(positions, varargin)
  [m, n] = size(positions);
  k      = 0;
  for i = 1:m
    for j = 1:n
      k = k + 1;
      subplot(m, n, k);
      imshow(varargin(positions(i, j)));
    end
  end
endfunction
```

Teste:

```
--> img00 = imread('img00.png');

--> img01 = imread('img01.png');

--> img02 = imread('img02.png');

--> img03 = imread('img03.png');

--> visualization([1 2; 3 4], img00, img01, img02, img03);
```




2 Compressão e SVD

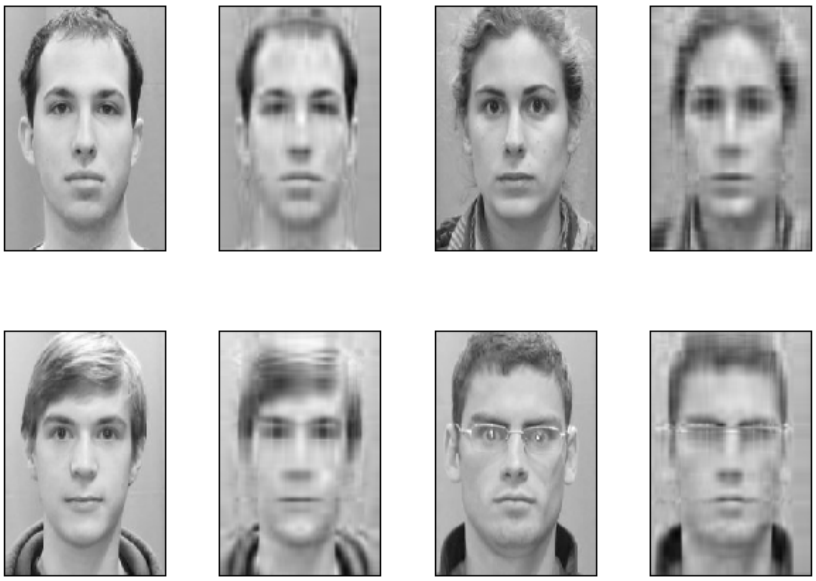
A decomposição em valores singulares pode comprimir uma imagem. Usando a função do Scilab svd que fornece a decomposição em valores singulares de uma matriz A (Veja o "help" desse comando no Scilab) e sendo r o número de valores singulares positivos, podemos comprimir a imagem representada pela matriz A usando apenas os s (s $<$ r) maiores valores singulares de A, conforme visto em aula.
Escreva uma função Scilab com variáveis de entrada A e p, onde 0 $<$ p $<$ 1 representa um percentual do número de valores singulares positivos de A a serem utilizados, para fazer uma compressão da imagem representada por A.
Faça s = max(1; o maior inteiro menor ou igual a pr) e use [U, S, V] = svd(A). Para isso, você precisará primeiro converter os elementos de A para reais. Use o comando double do Scilab. (Veja o "help" desse comando no Scilab).
Testar esta função de compressão de dados com vários valores de p: 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, ... Em cada caso visualize lado a lado a imagem original e a comprimida: C = U(:, 1:s) *

S(1:s, 1:s) * V'(1:s, :); //s é um inteiro entre 1 e r.
Você precisará reconverter os elementos de C para inteiros de 0 a 255. Para isso, use a função Scilab C = iconvert(C, 11); //converte para o format inteiro com 1 byte (0 a 255). **(Veja o "help" desse comando no Scilab).**
A partir de que valor de p a imagem comprimida é suficientemente "boa" comparada com a original?

```
function [C] = svd_compress(A, p)
  [U, S, V, r] = svd(double(A));
  s = max(1, floor(p * r));
  C = U(:, 1:s) * S(1:s, 1:s) * V'(1:s, :); //s é um inteiro entre 1 e r
  C = iconvert(C,11); //converte para o format inteiro com 1 byte (0 a 255)
endfunction
```

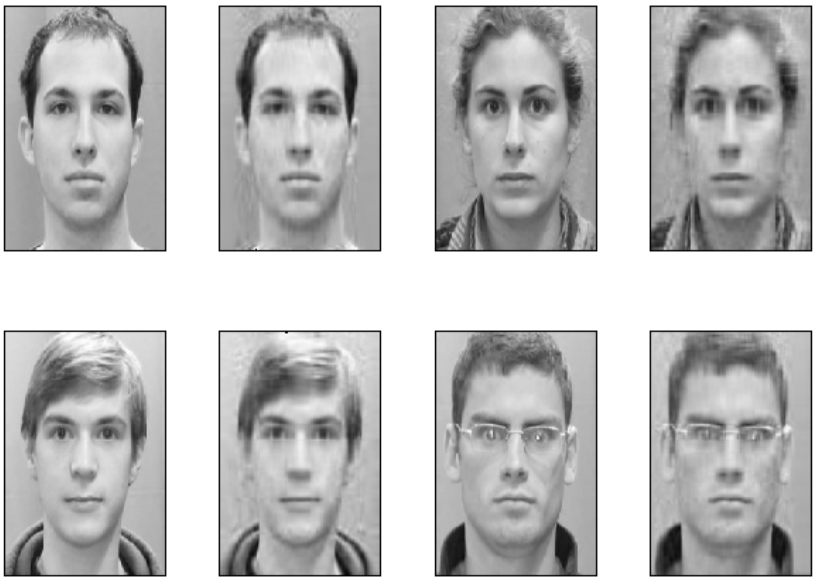
- Teste (p = 0.05)

```
--> p = 0.05; visualization([1 5 2 6; 3 7 4 8],
img00, img01, img02, img03,
svd_compress(img00, p),
svd_compress(img01, p),
svd_compress(img02, p),
svd_compress(img03, p));
```



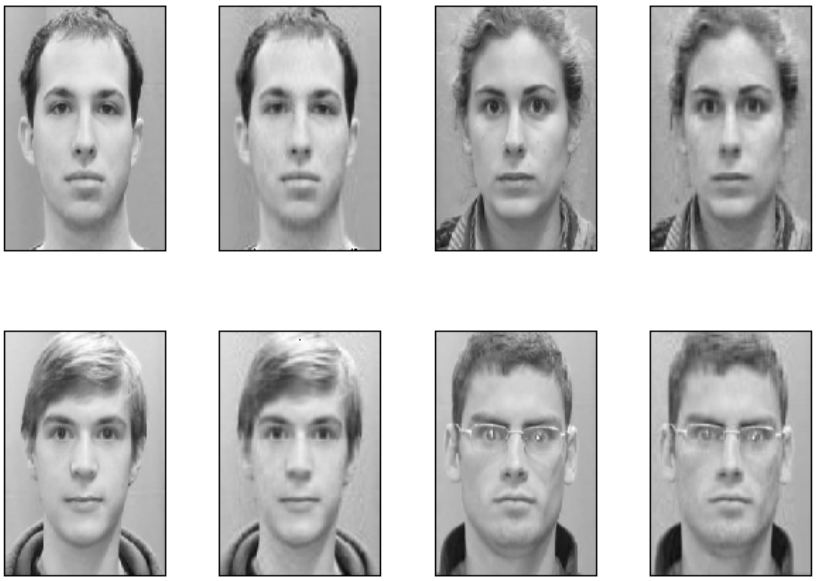
- Teste (p = 0.1)

```
--> p = 0.1; visualization([1 5 2 6; 3 7 4 8],
img00, img01, img02, img03,
svd_compress(img00, p),
svd_compress(img01, p),
svd_compress(img02, p),
svd_compress(img03, p));
```



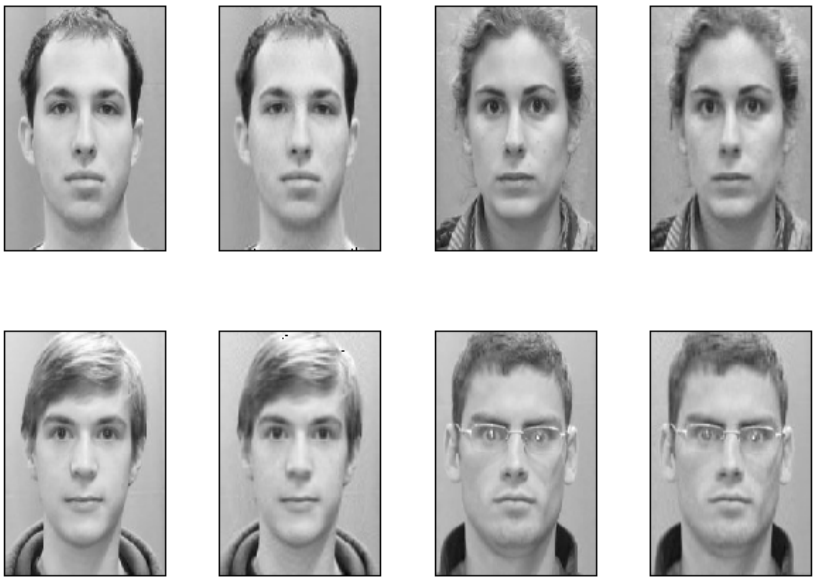
- Teste (p = 0.15)

```
--> p = 0.15; visualization([1 5 2 6; 3 7 4 8],
img00, img01, img02, img03,
svd_compress(img00, p),
svd_compress(img01, p),
svd_compress(img02, p),
svd_compress(img03, p));
```



- Teste (p = 0.2)

```
--> p = 0.2; visualization([1 5 2 6; 3 7 4 8],
img00, img01, img02, img03,
svd_compress(img00, p),
svd_compress(img01, p),
svd_compress(img02, p),
svd_compress(img03, p));
```



Para p = 0.2 é quase imperceptível a diferença entre as fotos originais e as comprimidas

2.1 Lenargb

```
function [C] = svd_compress_RGB(A, p)
  C          = A;
  C(:, :, 1) = svd_compress(A(:, :, 1), p);
  C(:, :, 2) = svd_compress(A(:, :, 2), p);
  C(:, :, 3) = svd_compress(A(:, :, 3), p);
endfunction
```

Respectivamente a imagem original, p=0.00, p=0.05, p=0.10, p=0.15, p=0.20, p=0.25, p=0.30 e p=0.35:

```
--> lenargb = imread('lenargb.png');

--> visualization([1 2 3; 4 5 6; 7 8 9],
lenargb,
svd_compress_RGB(lenargb, 0.00),
```

```
svd_compress_RGB(lenargb, 0.05),
svd_compress_RGB(lenargb, 0.10),
svd_compress_RGB(lenargb, 0.15),
svd_compress_RGB(lenargb, 0.20),
svd_compress_RGB(lenargb, 0.25),
svd_compress_RGB(lenargb, 0.30),
svd_compress_RGB(lenargb, 0.35))
```

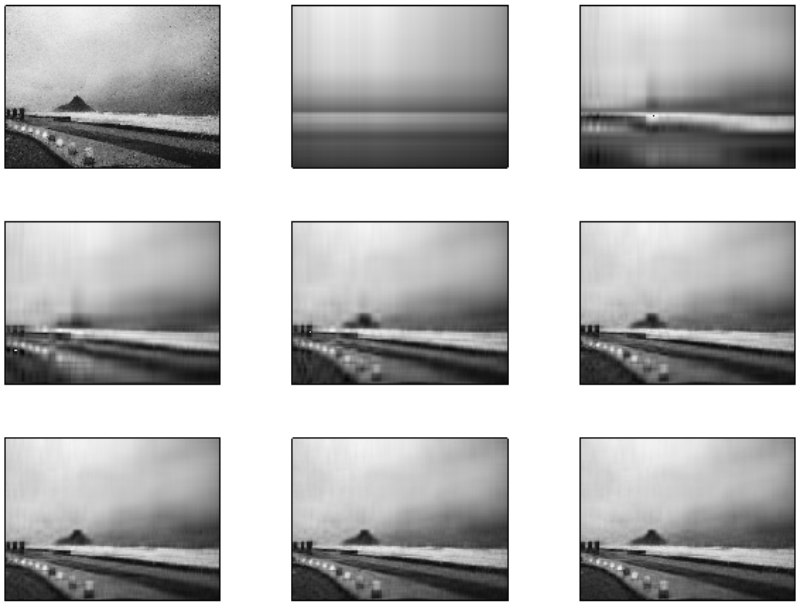
2.2 Marinha

Respectivamente a imagem original, p=0.00, p=0.01, p=0.02, p=0.03, p=0.04, p=0.05, p=0.06 e p=0.07:

```
--> marinha = imread('marinha.png');

--> visualization([1 2 3; 4 5 6; 7 8 9],
marinha,
svd_compress(marinha, 0.00))
svd_compress(marinha, 0.01),
svd_compress(marinha, 0.02),
```

```
svd_compress(marinha, 0.03),
svd_compress(marinha, 0.04),
svd_compress(marinha, 0.05)
svd_compress(marinha, 0.06),
svd_compress(marinha, 0.07);
```



```
--> [m n] = size(marinha)
m  =

3006.
n  =

5344.

--> marinha_size = m * n / 1000000
marinha_size  =

16.064064

--> s = floor(0.07 * rank(double(marinha)))
s  =
```

```
210.

--> marinha_comp_size = (m * s + s + s * n) / 1000000
marinha_comp_size  =

1.75371

--> marinha_comp_size / marinha_size
ans  =

0.1091698
```

Salvando as matrizes U, S e V com apenas 7% do valores singulares, mantendo uma boa qualidade, ao invés da imagem original reduz seu peso de 16MB para 1.75MB (quase 90% menos).