Usage of matrix decomposition

Matrix decomposition is a branch of linear algebra that has taken a more and more important role each day. This is because it takes a vital place in many modern data science disciplines: large data manipulation, digital image compression and processing, noise reduction, machine learning, recommender systems, interest rates,...

MatDeck has implemented several types of matrix decomposition techniques:

- Cholesky decomposition
- Cholesky LDP decomposition
- Hessenberg decomposition
- LU decomposition
- LU with permutation matrices
- QR decomposition
- QR decomposition with permutation matrix
- SVD decomposition
- Diagonalization

We will illustrate the usage of SVD decomposition (Singular Value Decomposition) on a image compression problem. Let's try to reduce the size of the image cow.jpg 200 x 200 as much as we can while we will trying to keep picture quality at acceptable level. The goal is to find the best ratio between size reduction and image quality.

Original image

pic := image read("cow.jpg")

Read image in variable **pic**

a := image2matrix(pic)

Convert image to matrix and save it to variable **a**

rank(a) = 199

Rank of matrix

The image we have imported has a rank of 199, so it will have 199 singular values. We will take a reduced number of singular values to form a lower rank approximation matrix of the original matrix. The amount of data of the original image we are dealing with is as follows

Image resolution = 200 x 200

$$\text{Original matrix} = U_{mat} \cdot S_{mat} \cdot V_{mat}^{T}$$

Matrix before SVD algorithm

U_{mat} = 200 x 200 = 40000 elements

S_{mat} = 200 x 200 = 40000 elements

V_{mat} = 200 x 200 = 40000 elements

It means that the total number of matrix entries needed to reconstruct the image using the SVD is 120.000. If we apply the low rank approximation we will get the following matrix sizes:

r = New number of singular values

U_{mat} = 200 x r elements
S_{mat} = r x r elements
V_{mat} = r x 200 elements

Lets reduce the number of singular values by changing the preferred number of diagonal values in matrix S_{mat} to zero. For gray-scale pictures, the matrices of blue, red and green values are the same while the transparency parameter alpha is 255. We will extract the red component, calculate it with red values' matrix and join blue, green and red values of the final matrix based on the extracted red component.

```
red := image2matrix(image red(pic))
svd := decsvd(red)

s_orig := svd[1]      v_orig := svd[2]
u_orig := svd[0]      s_new := svd[1]
```

Extract the red values of picture

Implement the SVD algorithm

Separate U_{mat}, S_{mat} and V_{mat} matrices

Now, we will leave the first 50 singular values on matrix S_{mat} and rest of the values from the main diagonal will be set to zero.

```
for(i := 50; i < rows(a); i += 1)
  s_new[i + i * rows(a)] = 0
```

All that is left to be done is to recreate the matrix based on the SVD multiplication algorithm with the changed diagonal matrix S_{mat}, and to join the red, blue, green and transparency values to form a new picture.

```
red_new := u_orig * s_new * mat transpose(v_orig)
red_new = round(abs(red_new))
red_new1 := matrix2image(red_new)
trans := matrix create(rows(a), cols(a), 255)
trans1 := matrix2image(trans)
img := image bgra(red_new1,red_new1,red_new1,trans1)
```

The newly created image is:

$$img1 := \text{image widget}(0, img)$$

Image after low rank approximation

After we have applied the low rank approximation, the size of the image matrix is:

50 = New number of singular values

U_{mat} = 200 x 50 = 10000 elements
S_{mat} = 50 x 50 = 2500 elements
V_{mat} = 50 x 200 = 10000 elements

Using the low rank approximation and taking only the 50 largest singular values reduces the total number of matrix elements needed to a lower amount of 22500. We have just reduced the total number of elements needed by 81.25% all the meanwhile retaining a high level of picture quality.