Singular Value Decomposition (SVD) Project

Alex Nisbet and Langyu Qie

11/30/20

Contents

Singular Value Decomposition

Preliminary Facts, Terms, and Aims

Let X be a (real) finite-dimensional inner product space, and denote by $\mathcal{L}(X)$ the space of bounded linear operators on X. The **adjoint** T^* of $T \in \mathcal{L}(X)$ is the unique (by Riesz) $T^* \in \mathcal{L}(X)$ characterized by $\langle Tx, y\rangle = \langle x, T^*y\rangle$ for all $x, y \in X$. If $T = T^*$, then T is called **self-adjoint**, and if T is in addition positive semi-definite, then T is said to be **positive**. T^*T and TT^* are always positive for $T \in \mathcal{L}(X)$, and the important property of positive operators is that they have square roots, i.e. if T is positive, then there exists a unique (positive) operator $\sqrt{T} \in \mathcal{L}(X)$ such that $T = \sqrt{T}\sqrt{T}$. Now, recall the following essential results:

- Polar Decomposition: Every $T \in \mathcal{L}(X)$ can be expressed $T = S\sqrt{T^*T}$, where $S \in \mathcal{L}(X)$ is an isometry.
- Spectral Theorem: Each self-adjoint $T \in \mathcal{L}(X)$ is diagonalizable with respect to some orthonormal basis of X.
- Matrix Representation: Putting $\dim X = k$ and given a basis $\mathcal{B}$ for X, each $T \in \mathcal{L}(X)$ has a unique representation as a matrix in $\mathbb{M}^{k\times k}$. Conversely, each matrix in $\mathbb{M}^{k\times k}$ is the representation with respect to $\mathcal{B}$ of a unique linear operator in $\mathcal{L}(X)$. Of importance is the fact that the adjoint of a matrix is given by its conjugate transpose.

We wish to examine the so-called **singular value decomposition (SVD)** for a matrix $A \in \mathbb{R}^{m\times n}$. This says that any $A \in \mathbb{R}^{m\times n}$ can be expressed in the form

$$A = U\Sigma V^*,$$

where $U \in \mathbb{R}^{m\times m}$ and $V \in \mathbb{R}^{n\times n}$ are unitary (i.e. orthogonal), and $\Sigma \in \mathbb{R}^{m\times n}$ is diagonal with non-negative entries.

Derivation in Embedded Form

Let $A \in \mathbb{R}^{m\times n}$, so that A is a bounded linear mapping from $\mathbb{R}^n$ into $\mathbb{R}^m$. Putting $r = \max(m, n)$, we can embed A as a mapping from $\mathbb{R}^{r\times r}$ into itself by appending rows or columns of zeros. Namely, if $m < n$, then we can add $n - m$ rows of zeros to A and the image vector, and if $n < m$, then we can add $n - m$ columns of zeros to A and the input vector. Denote by A_r the embedding of A into $\mathbb{R}^{r\times r}$. Our strategy will be to first form the singular value decomposition of A_r from its polar decomposition.

$\sqrt{A_r^*A_r} = (\sqrt{A^*A})_r$ is self-adjoint, and so by the spectral theorem there is some orthonormal basis $\{v_i\}_{i=1}^r$ of $\mathbb{R}^r$ which diagonalizes it, i.e.

$$\sqrt{A_r^*A_r}\;\vec{x} = \sum_{i=1}^{r} \sigma_i \langle \vec{x}, v_i\rangle v_i$$

for all $\vec{x} \in \mathbb{R}^r$. We choose to order the basis $\{v_i\}_{i=1}^r$ so that the σ_i are in descending order, and in particular these are all non-negative, since $A_r^* A_r$ is positive.

The polar decomposition allows us to express $A_r = S\sqrt{A_r^* A_r}$ for some isometry $S \in \mathbb{R}^{r \times r}$. Thus

$$A_r \vec{x} = \sum_{i=1}^{r} \sigma_i \langle \vec{x}, v_i \rangle S v_i = \sum_{i=1}^{r} \sigma_i \langle \vec{x}, v_i \rangle u_i,$$

for all $x \in \mathbb{R}^r$, where the $u_i := S v_i$ are orthonormal. If we interpret this sum in terms of matrices, then we have

$$A_r = U \Sigma V^*,$$

where U and V have the u_i and v_i for columns, respectively, and Σ is diagonal with $\Sigma_{ii} = \sigma_i$. In particular, U and V are orthogonal matrices, and we have the desired decomposition of A_r.

Singular Values

Note that the SVD expression for A_r implies the following relations:

$$A_r A_r^* = U(\Sigma \Sigma^*) U^* \qquad \text{and} \qquad A_r^* A_r = V(\Sigma^* \Sigma) V^*,$$

so that U and V are orthogonal matrices which diagonalize $A_r A_r^*$ and $A_r^* A_r$, respectively, and moreover $A_r A_r^*$ and $A_r^* A_r$ have identical eigenvalues, the squares σ_i^2. The σ_i found in this way are called the **singular values** of A_r.

Clearly, the columns of U and V are (unit) eigenvectors for $A_r^* A_r$ and $A_r^* A_r$, respectively, with columns u_i and v_i corresponding to the eigenvalue σ_i^2. In particular, we have

$$A_r(v_i) = U \Sigma V^*(v_i) = U \Sigma \left(\vec{\delta}_i \right) = U \left(\sigma_i \vec{\delta}_i \right) = \sigma_i u_i,$$

and similarly

$$A_r^*(u_i) = V \Sigma U^*(u_i) = \sigma_i v_i.$$

Generally speaking, if x and y are vectors and A a matrix with the relations

$$A^* x = \sigma y \quad \text{and} \quad Ay = \sigma x$$

for some non-negative real number σ, then x and y are called **left-singular** and **right-singular vectors** for σ. By multiplying on the left by A and A^* we get

$$AA^* x = \sigma A y = \sigma^2 x \quad \text{and} \quad A^* A y = \sigma A^* x = \sigma^2 y,$$

so that singular values are equivalently characterized by these relations. Note then that we can always choose pairs of left-singular and right-singular vectors to have norm 1. Thus the diagonal of Σ consists of the singular values of A_r, and the orthonormal columns of U and V are corresponding left-singular and right-singular vectors. In this way, we can express the left/right singular relations

$$A_r^* U = V\Sigma \quad \text{and} \quad A_r V = U\Sigma,$$

which amounts to rearrangements of the SVD expression $A_r = U\Sigma V^*$.

In particular, if we have a U or V, then we can determine the singular values by conjugating $A_r A_r^*$ or $A_r^* A_r$, and we can then form the other via the left-singular/right-singular relationship. For example, if we have orthonormal U, then $\Sigma = \sqrt{U^* A_r A_r^* U}$, and $V' = A_r^* U \Sigma^{-1}$, where Σ^{-1} is Σ with the non-zero diagonal entries replaced by their reciprocals. There is of course a difficulty here in that if A_r only has $d < r$ nonzero singular values, then V' will have $n - d$ columns of zeros on the right. If this happens then we can make always make V' orthogonal by choosing vectors that orthonormally extend the column space of V' to all of $\mathbb{R}^r$. We explore this in more detail next.

Calculating U_r and V_r

Above, A was an $m \times n$ real matrix, and we embedded A into $\mathbb{R}^{r\times r}$ as A_r, where $r = \max(m, n)$. Without loss of generality, suppose that $m \leq n$—for otherwise we could instead consider A^* and the SVD $A^* = V\Sigma U^*$. In this case A_r was gotten by appending $n - m$ rows of zeros to A. Since $A_r^* A_r$ and $A_r A_r^*$ share eigenvalues, it follows that A_r has at most m nonzero singular values. Consequently, in the product $A_r = U_r \Sigma V_r^*$, the $n - m$ rightmost columns of U_r and V_r are inconsequential in that they are zeroed out in the product. Thus, if we are to calculate using the singular relations, these last $n - m$ columns aren't calculated—so what should we do?

First consider U_r. This is supposed to diagonalize $A_r A_r^*$, and $A_r A_r^*$ is zero outside of the top left $m \times m$ block, and so if U_r is calculated from $A_r A_r^*$, then its top left $m \times m$ block does the diagonalizing work. That is, U_r is an $m \times m$ orthonormal matrix U diagonalizing AA^* which is embedded and extended orthonormally into $\mathbb{R}^{r\times r}$ by, say, making the remaining $n-m$ diagonals 1. If V_r were then to be constructed via the singular relation $V_r = A_r^* U_r \Sigma^{-1}$, V_r would have orthonormal n-vectors in the first m columns and zero vectors for the remaining $n - m$ columns. V_r as calculated is not then an orthogonal matrix, but is expandable to one by replacing the zero columns. This would however require some work.

Now, if we were instead to calculate V_r as an orthonormal matrix diagonalizing $A^*A = A_r^* A_r$, then we do not have this issue. In this case, calculating U_r via the singular relation $U_r = A_r V \Sigma^{-1}$ gives an $m \times m$ block with zeros elsewhere, which is easily orthonormally extended by changing the last $n - m$ diagonals to 1.

Thus, if $m < n$, we should first calculate V_r and Σ by diagonalizing A^*A, and then calculate U_r using the singular relation $U_r = A_r V_r \Sigma^{-1}$. And if $m > n$, then we calculate U_r by diagonalizing and V_r via the singular relations.

In the end we want $U \in \mathbb{R}^{m\times m}$, $\Sigma \in \mathbb{R}^{m\times n}$, and $V \in \mathbb{R}^{n\times n}$. If $m \leq n$, then U_r can be made $m \times m$

by simply discarding the last $n-m$ rows and columns, Σ can be made $m \times n$ by similarly discarding the last $n-m$ rows, and V_r is already $n \times n$. If $m > n$, then we follow the same process with the roles of U_r and V_r interchanged.

Implementation

Description of Program

For an input matrix $A \in \mathbb{R}^{m \times n}$ we consider A if $m \leq n$ and A^* if $m > n$ (call this A). First we use orthogonal similarity transformations to put A^*A into upper Hessenberg form. This is done with a composition Q of Householder reflections, and since A^*A is symmetric, the output H is symmetric tridiagonal. Next we iteratively apply Givens rotations to H until it is diagonal, and record this composition of rotations as G. The diagonal matrix is Σ^2, so we take the square root and discard the bottom $n-m$ rows to find Σ, and we calculate V as $V = Q^*G^*$:

$$\Sigma^2 = G(Q(A^*A)Q^*)S^* \quad \Longrightarrow \quad A^*A = (Q^*G^*)\Sigma^2(GQ) = V\Sigma^2V^*.$$

Next we store Σ^{-1} by replacing the nonzero diagonals of Σ by their reciprocals and transposing, and we calculate $U = AV\Sigma^{-1}$. If $m \leq n$ for our original input A, then we output U, Σ, and V, and if originally $m > n$, then we relabel U as V, relabel V as U, and transpose Σ.

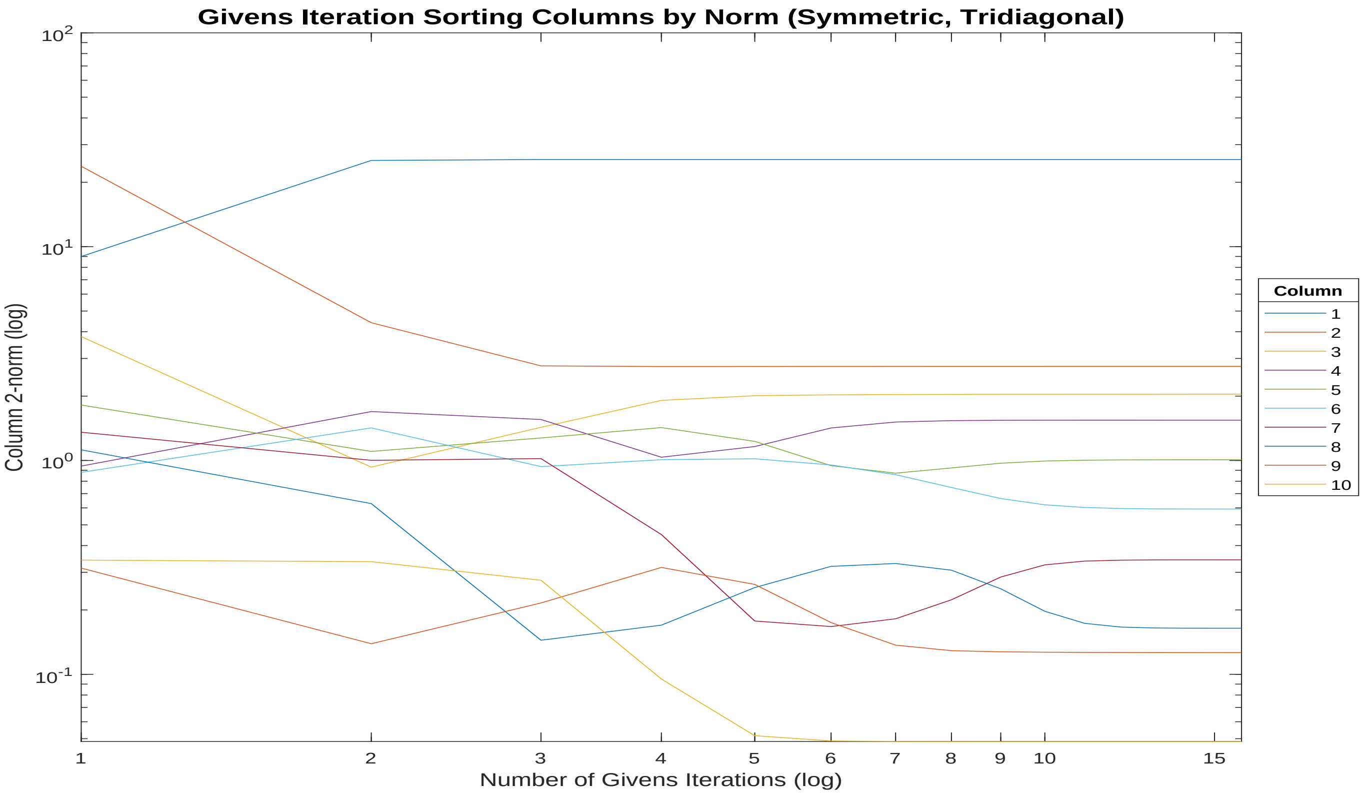
SVD Program

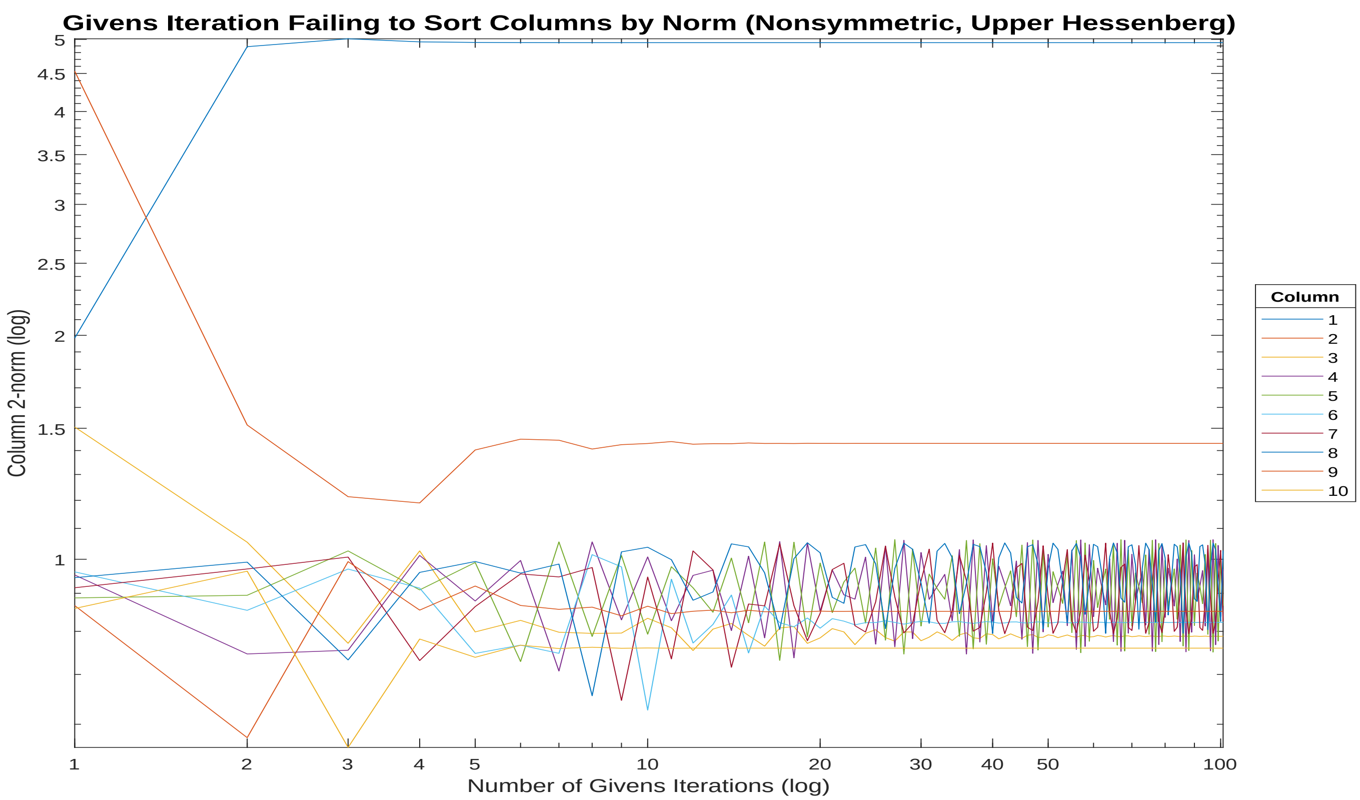
```
%This program takes an m-by-n matrix A and computes the SVD
%decomposition. U and V are m-by-m and n-by-n orthonormal matrices
%such that A=U*SIGMA*V' for m-by-n zero matrix SIGMA with diagonal
%consisting of the singular values of A.

function [U,SIGMA,V] = SVD(A)
    [m,n] = size(A);
    dual = 0;   %indicate whether working with A or A'
    if m > n    %standardizes m>n to m<n
        A = A';
        [m,n] = size(A);
        dual = 1;
    end
    [H,Q] = Hessenberg(A'*A);
    [SIGMA2,G] = IterateGivens(H,1000); %iterations determine accuracy
    SIGMA = sqrt(abs(SIGMA2));
```

```
    SIGMAINV = zeros(n);
    for i = 1:n %construct pseudoinverse of SIGMA
        if SIGMA(i,i) ~= 0
            SIGMAINV(i,i) = 1/SIGMA(i,i);
        end
    end
    V = Q'*G';
    U = A*V*SIGMAINV;
    U = U(1:m,1:m); %trim U to be m-by-m
    SIGMA = SIGMA(1:m,1:n); %trim SIGMA to be m-by-n
    if dual == 1 %convert answer from A' to A if swapped
        [U,V] = deal(V,U);
    end
end
```

Remark on Givens Rotations

One point that has not been addressed is that we required the singular values of A to be ordered from largest to smallest in Σ. Interestingly, since H is symmetric tridiagonal, iterating the Givens rotations does this automatically. The following graph comes from iteratively applying the Givens rotations to a 10×10 symmetric tridiagonal matrix. Each line corresponds to a column vector, the x-axis represents the number of iterations, and the y-axis the column 2-norm. Note that the sorting happens quite quickly, and in particular these norms converge to the singular values. This does not occur for a general upper Hessenberg matrix.


Givens Iteration Sorting Columns by Norm (Symmetric, Tridiagonal)
Column 2-norm (log)
Number of Givens Iterations (log)
Column
1
2
3
4
5
6
7
8
9
10


Givens Iteration Failing to Sort Columns by Norm (Nonsymmetric, Upper Hessenberg)
Column 2-norm (log)
Number of Givens Iterations (log)
Column
1
2
3
4
5
6
7
8
9
10

Uses of SVD

Pseudoinverse

Nearest Orthogonal Matrix

Let $U\Sigma V^*$ be the singular value decomposition of A. Then $Q = UV^*$ is the nearest orthogonal matrix to A , i.e. $\|A - Q\|_2 \geq \|A - UV^*\|_2$ for all orthogonal matrices Q given the norm of the matrix is defined as $\|A\|_2 = \sigma_{max}(A)$, where $\sigma_{max}(A)$ represents the largest singular value of matrix A.

Now we prove that UV^* be the singular value decomposition of A.

$$\|A - Q\| = \|U\Sigma V^* - Q\| = \|U^*(U\Sigma V^* - Q)V\| = \|\Sigma - U^*QV\| = \|\Sigma - Q'\|\,, \tag{1}$$

where

$$\begin{aligned} \|\Sigma - Q'\| = \max_{\|x\|=1} \|\Sigma x - Q'x\| \geq \max_{\|x\|=1} |\, \|\Sigma x\| - \|Q'x\|\,| = \max_{\|x\|=1} |\, \|\Sigma x\| - 1| \\ = \max(|\sigma_1 - 1|, |\sigma_n - 1|) \\ = \|\Sigma - I\|\,. \end{aligned} \tag{2}$$

Since

$$\|\Sigma - I\| = \|U^*AV - I\| = \|U^*(A - UV^*)V\| = \|A - UV^*\| \tag{3}$$

From equation (1),(2),(3), we have

$$\|A - Q\|_2 \geq \|A - UV^*\|_2\,,$$

By this point, we've proved UV^* is the nearest orthogonal matrix to A.

Lower Rank Approximation of a Matrix

Least Squares Problem

The singular value decomposition (SVD) of a matrix A is very useful in the context of least squares problems. Consider the linear least square

$$\min_x \|Ax - b\|_2^2$$

Let$A = U\Sigma V^*$ be the SVD of $A \in \mathbb{R}^{m\times n}$. Using the orthogonality of U and V we have

$$\|Ax - b\|_2^2 = \|U^*(AVV^*x - b)\|_2^2 = \left\|\Sigma \underbrace{V^*x}_{=\text{z}} - U^*b\right\|_2^2 = \sum_{i=1}^{r} (\sigma_i z_i - u_i^* b)^2 + \sum_{i=r+1}^{m} (u_i^* b)^2$$

The solution is given

$$z_i = \frac{u_i^* b}{\sigma_i}, i = 1, 2, ...r,$$

$$z_i = arbitrary, i = r+1, ..., n.$$

As a result

$$\min_x \|Ax - b\|_2^2 = \sum_{i=r+1}^{m} (u_i^* b)^2.$$

Recall that $z = V^* x$.Since V is orthogonal, we find that

$$\|x\|_2 = \|VV^* x\|_2 = \|V^* x\|_2 = \|z\|_2 .$$

All solutions of the linear least squares problem are given by $z = V^* x$ with

$$z_i = \frac{u_i^* b}{\sigma_i}, i = 1, 2, ...r,$$

$$z_i = arbitrary, i = r+1, ..., n.$$

The minimum norm solution of the linear least squares problem is given by

$$x_\dagger = V z_\dagger,$$

where $z_\dagger \in \mathbb{R}^n$ is the vector with entries

$$z_i^\dagger = \frac{u_i^* b}{\sigma_i}, i = 1, ..., r,$$

$$z_i^\dagger = 0, i = r+1, ..., n.$$

The minimum norm solution is

$$x_\dagger = \sum_{i=1}^{r} \frac{u_i^* b}{\sigma_i} v_i.$$

Solving LLS with SVD Decomposition code

```
%This program computes the least square solution for linear equation
   Ax=b and the value of the norm of Ax-b.

function [x]=LLS_SVD(A,b)
[~,n]=size(A);
% compute the SVD:
[U,S,V] = SVD(A);
s = diag(S);
```

```
% determine the effective rank r of A using singular values
r = 1;
while( r < size(A,2) && s(r+1) >= max(size(A))*eps*s(1) )
r = r+1;
end
d = U'*b;
x = V* ( [d(1:r)./s(1:r); zeros(n-r,1) ] );
disp('error=');
disp(norm(A*x-b));
end
```

Image Compression

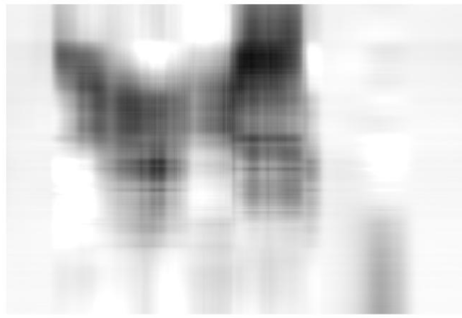
Image compression with singular value decomposition is a frequently occurring application of the method. The image is treated as a matrix of pixels with corresponding color values and is decomposed into smaller ranks that retain only the essential information that comprises the image. In this example, we are interested in compressing the below 433×650 image of a cat into a real-valued representation of the picture which will result in a smaller image file size.

The method of image compression with SVD is based on the idea that if SVD is known, some of the singular values σ are significant while others are small and not significant. Hence, if the significant values are kept and the small values are discarded(set to 0) then only the columns of U and V corresponding to the singular values are used. We will see in the following that as more and more singular values are kept, the quality and representation compared to the original image improves.

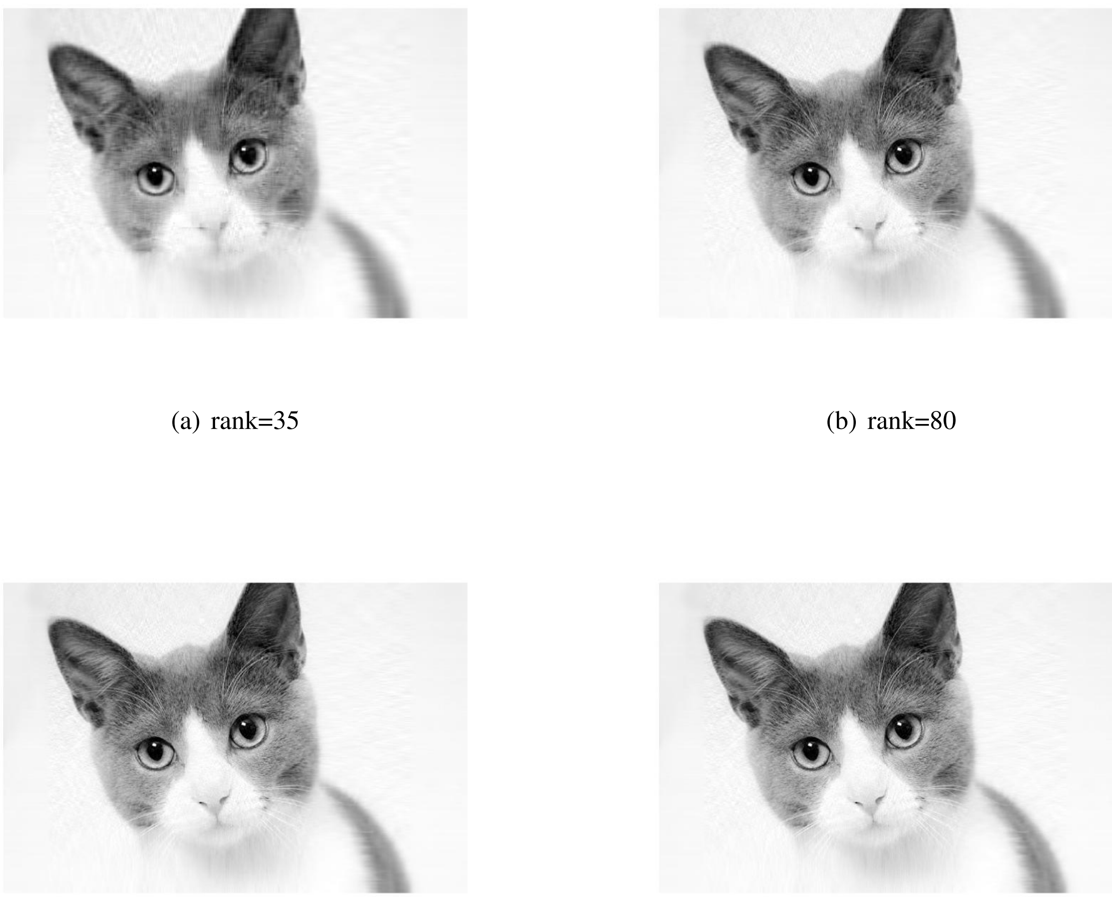
```
cat=imread('cat.jpg');
imshow(cat);
I=rgb2gray(cat);
figure;
imshow(I);
```

First, we turned the image into a matrix I of its size, namely, the output I is of size 433×650. Then we performed SVD on I and took the resulting matrix factorization and reconstructed the original matrix:

```
I2=im2double(I);
[u,s,v]=SVD(I2);
s2=s;
s2(5:end,:)=0;
s2(:, 5:end)=0;
D=u*s2*v';
imshow(D);
s2=s;
s2(35:end,:)=0;
s2(:, 35:end)=0;
D=u*s2*v';
imshow(D);
s2=s;
s2(80:end,:)=0;
s2(:, 80:end)=0;
D=u*s2*v';
imshow(D);
s2=s;
s2(150:end,:)=0;
s2(:, 150:end)=0;
D=u*s2*v';
imshow(D);
s2=s;
s2(200:end,:)=0;
s2(:, 200:end)=0;
D=u*s2*v';
imshow(D);
```



With just 5 singular values remaining the result image retains very few of the original image's characteristics. Thus, we want to keep more singular values so that the image of reconstructed matrix could be more alike to the original one.



(a) rank=35

(b) rank=80

(c) rank=150

(d) rank=200

Figure 1: Visual comparison between the reconstructed image with matrix rank=35,80,150,200.

From four graphs above, we found that at rank 35, the resulting image is much more representative of the original. As we keep more singular values, the picture becomes more detailed and less blurred. At rank 200, the resulting compressed image is rather unrecognizable from the original one. The rank 200 image has a file size of 37KB compared to the original image size of 49KB, which results in a 24% smaller file size. We can see the difference in the file sizes quickly converge to around -24%, likely indicating further ranks would not result in a more efficient compression ratio.

Appendix: Programs Used

Householder Reflection

```
%This function takes a vector x as an input and outputs the
%Householder reflection R such that Rx has the 2-norm of x in the
%first component and zeros in the others.

function R = Householder(x)
    n = length(x);
    alpha = sign(x(1))*norm(x);
    x = -x;
    x(1) = x(1) - alpha;
    R = eye(n) - (2/(x.'*x))*x*x.';
end
```

Upper Hessenberg Form

```
%This function takes an n-by-n matrix A and converts it to upper Hessenberg
%form H using Householder reflections. Q is the composition of Householder
%reflections such that H=QAQ'.

function [H,Q] = Hessenberg(A)
        [n,~] = size(A);
    for j = 1:n-2
        R = Householder(A(j+1:n,j));
        P(:,:,j) = blkdiag(eye(j),R);
        A = P(:,:,j)*A*P(:,:,j)';
```

```
    end
    Q = P(:,:,1);
    for j = 2:n-2
        Q= P(:,:,j)*Q;
    end
    H = A;
end
```

Givens Rotation

```
%This function takes an n-by-n Upper Hessenberg matrix A and uses
%Givens rotations to transform it into another Hessenberg matrix
%via orthonormal similarity transformations. In particular, the
%function iteratively multiplies by a Givens rotation on the left
%for each column (except the last), and then at the end multiplies
%on the right by the transpose of the product of Givens matrices.
%The end product of Givens matrices is called Q, and the output
%upper Hessenberg Matrix is called H.

function [H,Q] = Givens(A)
    [n,~] = size(A);
    for j = 1:n-1
        alpha = sqrt(A(j,j)^2 + A(j+1,j)^2);
        c = A(j,j)/alpha; s = A(j+1,j)/alpha;
        G(:,:,j) = eye(n);
        G(j,j,j) = c; G(j,j+1,j) = s;
        G(j+1,j,j) = -s; G(j+1,j+1,j) = c;
        A = G(:,:,j)*A;
    end
    Q = G(:,:,1);
    for j = 2:n-1
        Q = G(:,:,j)*Q;
    end
    H=A*Q';
end
```

Givens Rotation Iteration

```
%This function is for iterating the Givens function. H is the
%resulting Upper Hessenberg matrix, and Q is the composition of
%Givens rotations.

function [H,Q] = IterateGivens(A,k)
    [H,Q] = Givens(A);
    for i = 1:k-1
        [H,G] = Givens(H);
        Q = G*Q;
    end
end
```