Image Compression with SVD

Joseph Johnston

March 16, 2018

1 Abstract

This article offers a derivation of the singular value decomposition (SVD) and its application in image compression. First introduced is the concept and properties of a symmetric matrix. These properties are exploited to apply to any matrix such that the matrix transformation can be decomposed with SVD into a simple scaling transformation. The simplicity of the scaling transformation offers opportunities to accurately approximate the information of the matrix. Such approximation can be applicable to the compression of images. An example of SVD inspired image compression is presented at the end.

2 Introduction

Linear transformations can be difficult to work with. If the appropriate basis is provided to any given linear transformation, however, the linear transformation becomes a mere scaling transformation. When a linear transformation is reduced to a scaling transformation, the transformation is simpler to understand and the simplicity provides opportunities for modifying the transformation in helpful ways. Fortunately, every linear transformation can be reduced to a scaling transformation with appropriate choice of input and output bases. The mathematical formulation below will describe how every linear transformation can be represented in terms of a scaling transformation with an appropriate set of new bases.

3 Mathematical Formulation

It is first important to note that any linear transformation can be represented by a matrix. That is, a linear transformation from R^n to R^m can take the form of an $m \times n$ matrix. The idea of a linear transformation having the simple function of scaling relates directly to eigenvalues and eigenvectors. If a matrix $A_{m\times n}$ is not square, however, it will not have eigenvalues. In order to work with eigenvalues the matrix A will need to be represented in some square form. A guess for a useful representation is A^TA or AA^T. Both of these resulting matrices are square. It is also important to realize they are both symmetric. Symmetric matrices have particular properties when it comes to eigenvalues and eigenvectors. These properties are now explored.

3.1 Symmetric Matrices

Suppose K is a symmetric matrix of dimensions $n \times n$. It will not be proven here, but K has all real eigenvalues. For a symmetric matrix K the property $(Kv) \cdot w = (Kw) \cdot v$ can be exploited. Given two distinct eigenvalues of K,

$$\begin{aligned}
&Kv = \lambda v,\ Kw = \alpha w \\
&(Kv) \cdot w = \lambda v \cdot w = (Kw) \cdot v = \alpha w \cdot v \implies \\
&\lambda(v \cdot w) = \alpha(v \cdot w) \implies (\lambda - \alpha)(v \cdot w) = 0 \implies v \cdot w = 0
\end{aligned}$$

That is, for any two distinct eigenvalues of K, any corresponding eigenvectors are orthogonal. In other words, all eigenspaces corresponding to distinct eigenvalues are orthogonal eigenspaces. Another crucial property that will not be proven is that K is complete. That means any eigenvalue of multiplicity d corresponds to an eigenspace of dimension d.

These properties can be combined to realize that an orthonormal basis for R^n can be formed from a set of eigenvectors of K. This is because there are a total of n eigenvectors. For an eignevector of mulitplicity d an orthonormal basis can be formed for the corresponding eigenspace of dimesion d using Gram-Schmidt or another method. Each eigenspace is orthogonal to the others. Thus the orthonormal bases for the eigenspaces can be put into a set, forming an orthonormal basis for R^n.

There is one final property of symmetric matrices that will be established. Since $(K - \lambda I)v = 0$, the eigenspace for the zero-eigenvalue is equivalent to the non-trivial kernal of K. With v any element of the zero-eigenspace it is

the case that $(K - 0I)v = 0v \implies Kv = 0$ and since v is an eigenvector it is non-zero so it is an element of the non-trivial kernal of K. This equivalence implies that the zero-eigenspace has the same dimension $(n-r)$ as the kernal of K. Since K is complete, the multiplicity of the zero-eigenvalue is the same as the dimension of the zero-eigenspace, that is $n - r$. With a total of n (all real) eigenvalues there must be r non-zero eigenvalues. Because all eigenspaces are othogonal, the eigenvectors for non-zero eigenvalues must be othogonal to the zero-eigenspace, and together with the basis for the zero-eigenspace they must complete a basis for R^n. As such, these eigenvectors for non-zero eigenvalues must form a basis for the othogonal complement of the zero-eigenspace or equivalently the kernal. The orthogonal complement of the kernal is the corange. Hence the eigenvectors for the r non-zero eigenvalues form a basis for the corange (and range) of K, while the eigenvectors for the $n - r$ zero eigenvalues form a basis for the kernal (and cokernal) of K. Together, the eigenvectors for the n eigenvalues form a basis for R^n.

3.2 Symmetric Properties with A

Now that the properties of symmetric matrices have been established, the focus becomes how to take advantage of these properties with the symmetric matrices A^TA and AA^T to decompose A into the desired scaling transformation. Suppose λ is an eigenvalue and v an eigenvector of A^TA. Then

$$A^TAv = \lambda v \implies AA^T(Av) = \lambda(Av)$$
$$\text{if } \lambda v \neq 0 \implies A^T(Av) \neq 0 \implies Av \neq 0$$

This means every non-zero eigenvalue of A^TA is also one for AA^T, and a corresponding eigenvector for AA^T is Av where v is a corresponding eigenvector for A^TA. The eigenvectors of A^TA for the zero-eigenvalue can also be found as Av. Likewise,

$$AA^Tw = \alpha w \implies A^TA(A^Tw) = \alpha(A^Tw)$$
$$\text{if } \alpha w \neq 0 \implies A(A^Tw) \neq 0 \implies A^Tw \neq 0$$

Every non-zero eigenvalue of AA^T is also one for A^TA, and a corresponding eigenvector for A^TA is A^Tw where w is a corresponding eigenvector for AA^T.

The eigenvectors of AA^T for the zero-eigenvalue can also be found as $A^T w$. Hence $A^T A$ and AA^T have identical non-zero eigenvalues and closely related corresponding eigenvectors.

With matrix input and output subspace analysis, the below relationships can be established.

$$\text{rng}(AA^T) = \text{rng}(A),\ \text{rng}(A^T A) = \text{rng}(A^T)$$
$$\dim(\text{rng}(A)) = \dim(\text{rng}(A^T)) = r \implies \dim(\text{rng}(AA^T)) = \dim(\text{rng}(A^T A)) = r$$

It is evident that the rank of $A^T A$ and AA^T is r. Since symmetrix matrices have rank equal to the number of non-zero eigenvalues, $A^T A$ and AA^T both have r non-zero eigenvalues. The eigenvalue and eigenvector relationships between $A^T A$ and AA^T will soon be useful.

3.3 Decomposing A

As the goal is to express A as a scaling transformation, it may be helpful to first attempt to express $A^T A$ and AA^T as scaling transformations. Beginning with $A^T A$, a new basis for the input and output vector space is needed. This means an appropriate new set of basis vectors for R^n given that $A^T A$ has dimensions $n \times n$. If the columns of a matrix Q are the new basis vectors then the vector Qv is the input vector $v \in R^n$ expressed in terms of the new basis, while Qw is the output vector $w \in R^n$ expressed in the new basis. For a scaling transformation it must be that $A^T Ax = \lambda x,\ x \in R^n$. Thus the input vector must be an eigenvector. With this condition in mind perhaps if the new basis vectors were eigenvectors then any input vector could be expressed as an eigenvector. An eigenvector basis is possible because eigenvectors of the symmetric matrix $A^T A$ are able to form an orthonormal basis for R^n. Hence the linear transformation of $A^T A$ can be expressed as

$$(A^T A)(Qv) = (Qw) \implies (Q^T A^T AQ)v = w$$

where the columns of Q form an orthonormal eigenvector basis for R^n. By choosing an orthonormal basis it is the case that $Q^T Q = I$ and the new scaling transformation is expressed as $C_Q = Q^T A^T AQ$. To prove that this transformation is indeed a scaling transformation meaning that C_Q is a diagonal matrix, the form $QC_Q = A^T AQ$ is examined column wise, with j as

the column index.

$$
\begin{aligned}
&QC_{Qj} = A^T A Q_j,\ 1 \leq j \leq n \\
&QC_{Qj} = C_{Q1j}Q_1 + \cdots + C_{Qnj}Q_n \\
&A^T A Q_j = \lambda_j Q_j \\
&C_{Q1j}Q_1 + \cdots + C_{Qnj}Q_n = \lambda_j Q_j \\
&0 = C_{Q1j}Q_1 + \cdots + (C_{Qjj} - \lambda_j)Q_j + \cdots + C_{Qnj}Q_n
\end{aligned}
$$

Since the columns of Q form a basis they are linearly independent and therefore the only way to for the previous equation to be true is if all coefficients of the Q columns are zero. This means for any column j of C_Q

$$C_{Q1j} = 0, \ldots, C_{Q(j-1)j} = 0,\ C_{Qjj} = \lambda_j,\ C_{Q(j+1)j} = 0, \ldots, C_{Qnj} = 0$$

showing that C_Q is a diagonal matrix with eigenvalue entries.

The linear transformation AA^T can similarly be expressed as a scaling transformation as $C_U = U^T AA^T U$ where the columns of U form an othonormal eigenvector basis for R^m, and the diagonal values of C_U are the eigenvalues of AA^T.

In order to change the input and output bases for the A matrix itself the form $AXv = Yw$ will be necessary where $v \in R^n$ is the input vector, $w \in R^m$ the output vector, $X_{n\times n}$ the new input basis, and $Y_{m\times m}$ the new output basis. The difference between input and output basis is necessary because the transformation A may not be square so the input and output vector spaces may differ. The matrices Q and U are possibilities for X and Y respectively worth exploring, keeping in mind that the goal is to find a matrix $C = Y^{-1}AX$ with diagonal form. Plugging in Q for X and U for Y

$$
\begin{aligned}
&C_Q = Q^T(A^T A)Q,\ C_U = U^T(AA^T)U,\ C = U^{-1}AQ = U^T AQ \\
&AQ = UC \implies A^T AQ = A^T UC \implies \\
&QC_Q = A^T UC \implies AQC_Q = AA^T UC \implies \\
&AQC_Q = UC_U C \implies U^T AQC_Q = C_U C = CC_Q
\end{aligned}
$$

The following two points are relevant.

- C_Q and C_U are both diagonal matrices with identical (eigenvalue) entries. Thus $C_Q = C_Q$.

- The only (not necessarily square) matrix, C, that will commute with a diagonal matrix, $C_Q = C_U$, is another diagonal matrix or the zero matrix.

The two points above lead to the conclusion that C must be either the desired diagonal matrix or the zero matrix. The following analysis applies regardless whether or not C is a diagonal or zero matrix. Looking at $AQ = UC$ column-wise

$$AQ_k = c_{k,k}U_k,\ k = 1, \ldots, \min(m, n)$$

The right hand side of the above comes from the fact that C has zero entries everywhere outside the diagonal. Since the column Q_k is an eigenvector v_k of A^TA, and since the column U_k is an eigenvector Av_k of AA^T corresponding to the same eigenvalue as v_k, the above is reformulated as

$$Av_k = c_{k,k}v_k$$

Taking the norm of both sides

$$c_{k,k} = \frac{||Av_k||}{||v_k||} \implies c_{k,k}^2 = \frac{Av_k \cdot Av_k}{v_k \cdot v_k} = \frac{v_k^T A^T A v_k}{v_k^T v_k} =$$
$$\frac{v_k^T(\lambda_k v_k)}{v_k^T v_k} = \frac{\lambda_k ||v_k||}{||v_k||} = \lambda_k \implies c_{k,k} = \sqrt{\lambda_k}$$

where $||v_k|| = 1$ because v_k is an unit eigenvector. Thus each non-zero diagonal entry of C is the square root of the same diagonal entry of C_Q and C_U. These diagonal entries of C are called the singular values of A.

It appears substituting Q and U for X and Y respectively results in a diagonal linear transformation as desired! With the input change of basis Q and the output change of basis U the form is now

$$AQv = Uw \implies U^TAQv = Cv = w$$
$$U^TAQ = C \implies A = UCQ^T$$

where the columns of Q form an orthonormal basis for R^n and the columns of U form an orthonormal basis for R^m. Since A^TA and AA^T have r non-zero eigenvalues, A has r non-zero singular values. These singular values appear on the top left diagonal entries of the $m \times n$ matrix C in decreasing order. This scaling transformation C and its change of bases Q and U constitute a decomposition of A called the singular value decomposition.

4 Image Compression by SVD

4.1 Data Compression

The decomposition shown above is called the 'thick' SVD, and it yields three matrices with a total of

$$m \times m + m \times n + n \times n$$

entries. Since C is a diagonal matrix with r non-zero diagonals, the $m - r$ columns of U when multiplied by C, are multiplied by zero. These $m - r$ columns from U and the corresponding $m - r$ zero rows of C can thus be removed without affecting the integrity of the decomposition. Likewise, the $n - r$ columns of Q and the $n - r$ zero columns of C can also be removed. In this case the new decomposition called the 'thin' SVD yields three matrices with a total of

$$m \times r + r \times r + r \times n$$

entries. This number of entries is likely smaller than $m \times n$ and further compression is applicable through the SVD.

The SVD is useful in the context of lossy data compression. This is because if the data is a linear transformation in the form of a matrix, an approximate linear transformation can be calculated by transforming A into the diagonal transformation C through the SVD. Then, some of the smaller diagonal entries of C, that is the smaller singular values of A, can be removed from C without significantly altering the linear transformation. The approximation is justified because the small diagonal values of C correspond to small and relatively unimportant scalings in the diagonal transformation. If C only retains the largest singular values then reconstucting A by the SVD UCQ^T will yield an approximate linear transformation for A.

The advantage of approximating the matrix A is that for every singular value of A removed from C, two corresponding vectors can be removed as well, one from Q and one from U. In other words, if the k largest singular values of A are chosen for approximation, only the corresponding two sets of k vectors are necessary to retain. This means that for k singular values kept,

$$k + km + kn$$

data points are retained. This can significantly reduce the number of necessary data points. We define the 'compression ratio' as

$$\text{compression ratio: } \frac{k+km+kn}{mn} = \frac{k(1+m+n)}{mn}$$

Since the singular values are the scalings factors of the transformation, the Euclidian norm of the singular values can be interpreted as a measure of the amount of information contained in the transformation. This particular norm has the name 'frobenius' norm. By taking the frobenius norm of an approximated matrix and dividing it by the frobenius norm of the original matrix, the result can be thought of as the proportion of information of the original matrix still contained in the approximate matrix. We thus define the 'remain ratio' as

$$\text{remain ratio: } \frac{\sqrt{\sigma_1^2+\cdots+\sigma_k^2}}{\sqrt{\sigma_1^2+\cdots+\sigma_r^2}},\ k \leq r$$

4.2 Image Compression

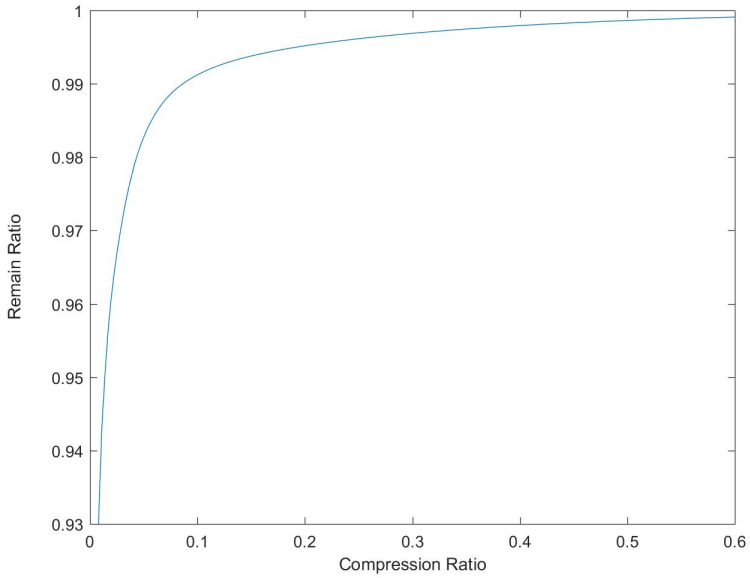
An black and white image can be represented as a matrix with the (i,j)th entry having a value between 0 and 1 representing the darkness of the (i,j)th pixel. Moreover, a colored image can be represented with multiple matrices. The colored image is split into three matrices, the first matrix corresponding to the red layer of the image, the second matrix to the green layer, and the third matrix to the blue layer. Each entry in each matrix is a number between 0 and 1 to represent how much of the particular color is present in the RGB makeup of the corresponding pixel.

Each color layer in the form of a matrix can be viewed as a linear transformation. The SVD data compression technique is thus applicable to compress each layer. Note that the compression ratio is the same for all 3 color layers of the original matrix since the same number of singular values will be kept for each layer. Of course commercial image compression is more complex than this.

4.3 Example

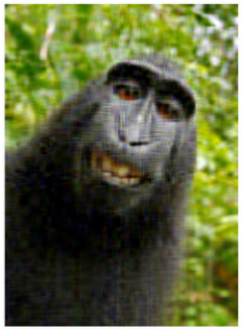
This example approximates a colored image by using the SVD compression technique for four different compression ratios. The four corresponding approximated images are shown. The number of singular values kept is equal

to the rank of the approximated matrix. A plot of the retain ratio versus the compression ratio is also displayed.

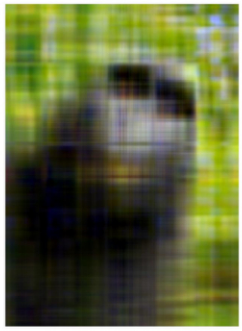


Rank 200

Rank 50



Rank 20



Rank 5

The more singular values kept in the modified matrix, the more information retained in the compressed image. The table below shows the relation between the rank, compression ratio, and remain ratio.

Rank	Compression Ratio	Remain Ratio
200	0.5561	0.9989
50	0.1390	0.9934
20	0.0556	0.9846
5	0.0139	0.9502

As seen in the table, keeping 200 singular values will keep 99.89% of the information of the original image, while the compression ratio is surprisingly low, only about 55.61%, which means the modified image occupies just only half the space of the original one. At the other end of the ranks, rank 5 yields an image retaining 95.02% of the information, but the image is barely recognizable. Comparing the images of for ranks 500 and 5 it becomes evident how much our recognition depends upon the many small image details. Just as the difference in information between the two images is a small $99.89 - 95.02 = 4.87\%$ yet the images appear dramatically different, so too do the genes between chimps and humans differ only by 1.2% yet traits and appearance differ dramatically.

5 Conclusion

The example above should demonstrate some applicability of SVD. The mathematical formulation of the SVD was intended to offer an additional perspective and additional meaning for origin, nature, and purpose of the SVD than is usually offered in introductory textbooks.

```
clear all
Img = imread('Monkey Selfie.jpg');
Img = im2double(Img);
Red = Img(:,:,1);
Green = Img(:,:,2);
Blue = Img(:,:,3);

ranks = [200,50,20,5,2];

for ii = 1:length(ranks)
        figure
```

```
        r = ranks(ii);
        %Red SVD
        [U_red,S_red,V_red] = svd(Red);
        U_red_thin = U_red(:,1:r);
        S_red_mod = Mod_Matrix(S_red,r);
        S_red_thin = S_red_mod(1:r,1:r);
        V_red_thin = V_red(:,1:r);
        Mod_red_thin = U_red_thin * S_red_thin * V_red_thin';
        %Green SVD
        [U_green,S_green,V_green] = svd(Green);
        U_green_thin = U_green(:,1:r);
        S_green_mod = Mod_Matrix(S_green,r);
        S_green_thin = S_green_mod(1:r,1:r);
        V_green_thin = V_green(:,1:r);
        Mod_green_thin = U_green_thin * S_green_thin *
            V_green_thin';
        %Blue SVD
        [U_blue,S_blue,V_blue] = svd(Blue);
        U_blue_thin = U_blue(:,1:r);
        S_blue_mod = Mod_Matrix(S_blue,r);
        S_blue_thin = S_blue_mod(1:r,1:r);
        V_blue_thin = V_blue(:,1:r);
        Mod_blue_thin = U_blue_thin * S_blue_thin * V_blue_thin
            ';
        %Reconstruct Image
        Mod_Img_thin = cat(3,Mod_red_thin,Mod_green_thin,
            Mod_blue_thin);
        imshow(Mod_Img_thin)
end
```

```
clear all
Img = imread('Monkey Selfie.jpg');
Img = im2double(Img);
Red = Img(:,:,1);
Green = Img(:,:,2);
Blue = Img(:,:,3);

[m,n] = size(Red);

ranks = [1:360];
RemainRatio = zeros(1,length(ranks));
CompRatio = ranks * (m+n+1)/(m*n);

for ii = 1:length(ranks)
        s_red = svds(Red,ranks(ii));
        s_green = svds(Green,ranks(ii));
```

```
17          s_blue = svds(Blue,ranks(ii));
18          Remain_Red_Ratio = norm(s_red)/norm(Red,'fro');
19          Remain_Green_Ratio = norm(s_green)/norm(Green,'fro');
20          Remain_Blue_Ratio = norm(s_blue)/norm(Blue,'fro');
21          RemainRatio(1,ii) = (Remain_Red_Ratio +
                Remain_Green_Ratio + Remain_Blue_Ratio)/3;
22          ii
23  end
24  plot(CompRatio,RemainRatio)
```

```
1   function Approx_S = Mod_Matrix(S,r)
2     Sigmas = diag(S);
3     Length_sigmas = length(Sigmas);
4     Approx_sigmas = Sigmas;
5     Approx_sigmas(r+1:end) = 0;
6     [R_s,C_s] = size(S);
7     Approx_S = zeros(R_s,C_s);
8     Approx_S(1:Length_sigmas,1:Length_sigmas) = diag(Approx_sigmas
          );
9   end
```

References

[1] "Relating the Trace and Frobenius Matrix Norms". 2005, August 31, Jason D. M. Rennie. Retrieved from http://qwone.com/ jason/writing/-traceFrobenius.pdf

[2] "DNA: Comparing Humans and Chimps". American Museum of Natural History. Retrieved from http://www.amnh.org/exhibitions/permanent-exhibitions/human-origins-and-cultural-halls/anne-and-bernard-spitzer-hall-of-human-origins/understanding-our-past/dna-comparing-humans-and-chimps/

[3] Cool Linear Algebra: Singular Value Decomposition. 2013, May 29. Retrieved from http://andrew.gibiansky.com/blog/mathematics/cool-linear-algebra-singular-value-decomposition/